Cosmos

Cultivate resilient smart Objects for Sustainable city applicatiOnS

Grant Agreement Nº 609043

D2.2.1 State of the Art Analysis and Requirements Definition (Initial)

WP2 "Requirements and Architecture"

Version: 1
Due Date: 28 February 2014
Delivery Date: 7th April 2014
Nature: Report
Dissemination Level: Public

Lead partner: UNIS
Authors: All Partners
Internal reviewers: Atos, NTUA, Siemens

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
1	31/03/2014	UNIS (editor) & All	All	

Annexes:

Nº	File Name	Title
1	COSMOS_Requirements_v1.xls	COSMOS REQUIREMENT (INITIAL VERSION)

Table of Contents

TABLE OF ACRONYMS

Acronym	Meaning
3DES (TDES)	Triple DES
ACL	Agent Communication Language
AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure
AMQP	Advanced Message Queuing Protocol
AMS	Agent Management System
API	Application Programming Interface
ASM	Agent Security Manager
ATSN	Agent-based Trust Model for Sensor NEtworks
AWS	Amazon Web Service
CCF	Consumable Crypto Firewall
CDMI	Cloud Data Management Interface
CEP	Complex Event Processing
CPU	Central Processing Unit
DDM	Dynamic Data Masking
DDoS	Distributed DoS
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
DSL	Domain Specific Language
DSS	Data Distribution Service
EAL	Evaluation Assurance Levels
ECA	Event Condition Action
ECC	Elliptic Curve Cryptography
EM	Expectation Maximization
EPC	Electronic Product Code
EPCIS	EPC Information Service
ESI	Energy Service Interface
ESP	Event Stream Processing
FIFO	First-In / First-Out
FIPA	Foundation for Physical Intelligent Agents
FLD	Fuzzy-Logic Device
GPS	Global Positioning System
GVE	Group VE
HAN	Home Area Network
HMAC	Key-Hash Message Authentication Code
ID	IDentifier
IETF	Internet Engineering Task Force
IHD	In Home Display

IoT	Internet of Things
IoT-A	Internet of Things - Architecture
Json	Java-Scrip Object Notation
M2M	Machine-to-Machine
MAPE-K	Monitor/Analyse/Plan/Execute - Knowldege
MaL	Maximum Likelihood
MANET	Mobile Ad-hoc NETwork
MAP	Maximum A Posteriori
MDM	Meter Data Management
ML	Machine Learning
MOF	Meta-Object Facility
MQTT(-S)	Message Queue Telemetry Transport (extension to Sensors)
N3	Notation 3
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
OMG	Object Management Group
ONS	Object Naming Service
OWL	Web Ontology Language
P2P	Peer-to-Peer
PDF	Probability Density Function
PIR	Private Information Retrieval
PKI	Public Key Infrastructure
PE	Physical Entity
QoS	Quality of Service
RDF	Resource Description Framework

RDF-S	RDF Schema
RFID	Radio Frequency IDentification
RFSN	Reputation-based Framework for Sensor Network
RSA	Rivest Shamir Adleman
RuleML	Rule Mark-up Language
S3	Simple Storage Service
SaND	Social Network and Discovery Engine
SDM	Static Data Masking
SDN	Storage Delivery Network
SGML	Standard Generalised Mark-up Language
SHA	Secure Hash Algorithm
SNIA	Storage Networking Industry Association
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language (recursive then)
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 Modelling Language
UNIs	(IoT-A) UNIfied requirements

VE	Virtual Entity
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WSN	Wireless Sensor Network
XML	eXtensible Markup Language
XMPP	eXtensible Message and Presence Protocol

1 Introduction

This first version of the State-of-the-Art Analysis and Requirements Definition is the preliminary version about this topic which will be complemented along the project life-span by two more iteration. Its two main objectives are to make an analysis of the literature and current state of the art in a number of domain which the COSMOS project touches and to identify the main requirements that the COSMOS project will use as incentives when designing the COSMOS Architecture (WP2) and working on the technical challenges identified in the Description of Work (work packages (W3, 4, 5, 6)).

Section 2 of this document gives a bit of the COSMOS background, 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 show 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 then provides a SOTA analysis and explains how COSMOS will position itself w.r.t the current SOTA. In this first iteration however there are many remaining open points being discussed –at the time of the writing- in several work package, consequently this positioning (what does COSMOS reuse, adapt, invent) is still under discussion and partially shown here. The second iteration of this chapter in D2.2.2 will reveal the whole positioning.

Section 4 introduces the process COSMOS is following as far as Requirement Collection is concerned. As we follow the IoT-A methodology, this section is a reminder of how we will proceed within COSMOS, and also explains where the Excel template used for Requirement Collection comes from.

The list of requirements is not presented in this document as we are using an Excel Sheet internally for that matter. This excel file is attached to this document as an appendix.

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 which were 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 reused in the context of COSMOS with a different meaning or interpretation.

This section therefore is made of two main parts which are respectively 1/ the Glossary (a table) and 2/ a conceptual model that describes how the COSMOS concepts related to each other (in the spirit of the IoT Domain Model) and shows/explicit 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 definitions of the concepts introduced in the list of COSMOS Requirements, and how they relate to IoT-A (when not strictly identical).

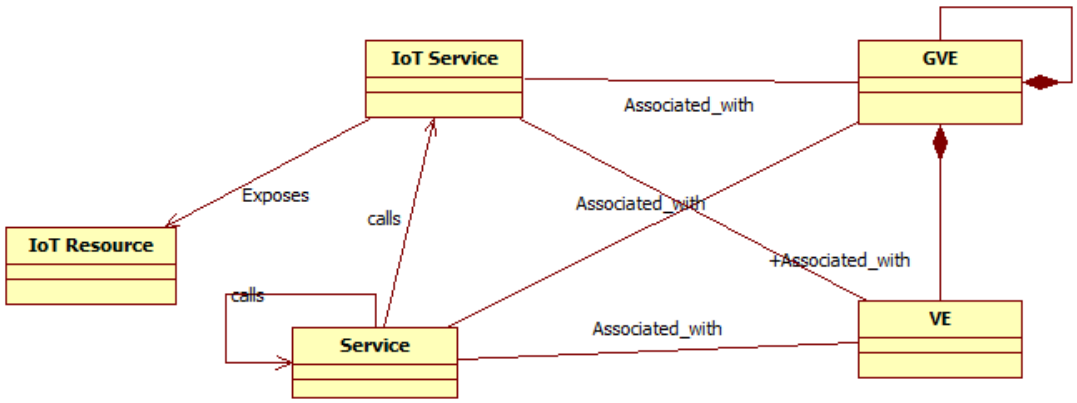
It is worth noting that the Architecture document will include a “customised” version of the IoT Domain Model from IoT-A adapted to the COSMOS terminology.

Concept	Definition	Concept in IoT-A
Object	The entity of interest in an IoT application. Objects can be Buses, room of a dwelling, flat/house, bus line, bus stops etc… all will be represented in the IoT system by VE’s or Group VE’s	Physical Entity (PE)
Experience	Experience are different type of knowledge (in the broad sense) which is exchanged between VE’s. It can be piece of knowledge (following an ontology), model resulting from Machine Learning, contextual information, knowhow etc…	n/a
Virtual Entity (VE)	The counterpart of the object in the Cyberworld	Virtual Entity
Group VE’s	A VE that represents a group of VE’s, e.g. a bus line GVE is made of Bus VE’s	Virtual Entity (IoT-A allows nesting of VE’s)

2.2 Conceptual Model

As already stated in the introduction of this document, the following picture shows only a short fragment of the COSMOS Domain Model restricted to VE’s (individual VE’s and Group

VE's), 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 VE's (either single VEs or Group VE's). VE are the representation in the "Cyber World" of the Real World Objects. VE's have service logics (see the MAPE-K paragraph to get a glimpse of how a VE service logic can be structured) and a set of attributes that describe the characteristics of a VE (or GVE). A VE is somehow "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 set its current position (as one of its attribute). Group VE's 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 GVs and are exposing IoT Resource in a standard way, while Services don't. 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.



3 State of the Art Analysis (SoTA)

This chapter provides an analysis of the State of the Art for the different technical domain we have either identified initially when writing the DoW or discovered when going further in the precise definition of the WP technical 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 coded as 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 most benefit 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 characteristic 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 Byte 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 internet of things 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 work load.

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 analyse 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, Katja & Kai-Uwe Sattler 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, Katja & Kai-Uwe Sattler 2009).

AnduIN provides the following operators:

1. **Anduin** 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-dependant 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 neighbour, 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, Katja & Kai-Uwe Sattler 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 summarised 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 Machine Learning and Analytics

Machine Learning (ML) at a high level working on regression and classification techniques, for our purposes ML covers the area of the use of statistical methods rather than logic driven methods. Areas of interest are in:

- Clustering/Classification;
- Pattern Recognition;
- Sensor Fusion;

- Regression methods – also can implement “optimal” regressions that will perform classical optimisation;
- Logic driven systems can be can be considered as “Expert Systems” with rule chains producing similar effects as ML and are more suitable for Analytics.

3.2.1. Interpolation

In the field of numerical analysis, Interpolation is defined as the method for constructing new data points within the range of a discrete set of known data points. In the context of Internet of things, we can define interpolation as predicting the sensor values in case of discontinuity or unavailability of the sensor readings at certain time or certain position. This concept can be applied to both time and space domain. If we apply this concept to time domain, it is called temporal Interpolation while if we apply this concept to space domain it is called spatial interpolation. The main concept remains the same. In order to elaborate it further considers a simple practical example. We have temperature sensors on three flats located next to each other. At certain time instant, one of the sensor readings becomes unavailable due to some fault. At that specific time instant, we can predict the missing sensor reading in two ways, either by using the previous values of the same sensor or by using the sensor readings of the neighbouring two flats. In former case, it will be called temporal Interpolation while in later case it will be an example of spatial Interpolation. The main concept of Interpolation will remain the same in both cases. And that’s why we will discuss interpolation in general.

3.2.1.1. Piecewise Constant Interpolation

It is the simplest Interpolation technique and it assumes that the system does not change over the last sampling instant and it assigns the nearest data value. In context of temporal Interpolation, this will be last sampled value while in context of spatial interpolation, it will be the reading of nearest neighbour sensor. In practical application, if the sampling rate is high or sensors exist in close proximity, piecewise constant interpolation will give accurate results and will be very useful for its simplicity.

3.2.1.2. Linear Interpolation

Linear Interpolation is another very simple technique used extensively in practical applications. Resource constrained sensor nodes prefer simple algorithms to avoid computation overheads. In linear Interpolation, system predicts that change between two consecutive readings is linear and can easily predict the missing value by mapping it onto linear line joining the two data points.

3.2.1.3. Polynomial Interpolation

In polynomial Interpolation, interpolant is replaced by a polynomial of higher degree to fit the sensor readings more accurately. It gives a better approximate of missing values at the expense of more memory and power usage.

3.2.1.4. Spline Interpolation

Linear Interpolation uses a linear function between two consecutive sample readings whereas Spline interpolation uses low degree polynomial between two consecutive sample readings but it chooses polynomial pieces in such a way between all intervals so that they fit smoothly together. The resulting function is called a spline. It is usually computationally more efficient

then polynomial interpolation and hence preferred on simple Polynomial interpolation in sensor networks.

3.2.1.5. Interpolation via Gaussian Processes:

Gaussian Process can be used for Inference of continuous values with a Gaussian process prior and is known as Gaussian process regression. It forms a powerful non-linear interpolation tool.

3.2.2. Extrapolation

In the field of numerical analysis, extrapolation can be defined as the process of estimating the value of a variable on the basis of its relationship with another variable beyond the observation interval. In internet of things domain, extrapolation refers to predicting the values of sensors in case of temporary withheld of a system. The sensor readings can become temporary unavailable and in that case previous sensor readings will be used to predict the future readings of the sensors.

It is similar to interpolation in many ways but it poses higher risk and greater uncertainty. A prior knowledge of a system is required to choose a suitable choice of extrapolation method for the system. Time series analysis of a data is required to predict if the data is continuous, smooth, possibly periodic etc.

3.2.2.1. Linear Extrapolation:

Linear extrapolation can be used in cases where graph of sensor readings form approximately a linear function. In most practical cases, various data points are used to average the slope of linear interpolant and used that slope to extend the graph further. This technique is also called linear prediction.

3.2.2.2. Polynomial Extrapolation:

A polynomial curve can be created using few or all previous data values and can be extended to predict the sensor readings in case of temporary disconnection. Extrapolation result can be improved by increasing the window size of previous data values but it puts extra burden on node computations. So we have to choose optimum window size considering power and memory constraints. Polynomial extrapolation can be done by using Lagrange interpolation method or by using Newton's method of finite differences.

3.2.3. State Estimation/Prediction

Estimation Methods are based on laws of probability and derived from control Theory. Estimation is used to compute a process state vector from a measurement vector or a sequence of measurement vectors (Bracio, Horn & Moller 1997). It is used to compute or estimate the state of a system at particular instant. In the context of Internet of Things, State Estimation Methods will be used when we want to estimate or predict the state of a system using current or previous inputs.

One technique used extensively in literature for state estimation is based on *Maximum Likelihood* (MaL). According to Brown et al. 1992(BROWN, C., DURRANT-WHYTE, H., LEONARD, J., RAO, B., AND STEER, B 1992), state estimation methods based on likelihood are more suitable when the system is not the outcome of a Random variable. In other words, we know the likelihood or probability density functions of different possible states.

Few examples of the distributed maximum likelihood estimators used are Decentralized *Expectation Maximization* (EM) algorithm (Nowak 2003) and the Local Maximum Likelihood Estimator (Blatt, Hero 2004) that relax the requirement of sharing all data as nodes computes local unbiased estimates that converges towards the global maximum likelihood situation.

Maximum Likelihood Estimator is not viable option if the state of a system to be measured is the outcome of a Random Variable. In such cases, *Maximum A Posteriori* (MAP) Estimator which is based on Bayesian theory is better option. The difference between MaL and MAP is that MaL assumes that state *x* is a fixed through unknown point of the parameter space, while MAP takes *x* as the outcome of a random variable with prior PDF known. In (Schmitt et al. 2002) MAP estimator is used to find the positions of mobile robots in a known environment and track the positions of autonomously moving objects.

In systems, where the states are fixed and are represented by deterministic expressions, MAP and MaL cannot be used. In such scenarios, one suitable technique is called Least Square Estimation Method. Least Square Method is a mathematical optimization technique that minimizes the difference between observed and predicted values of a state. The least square method searches for the value of *x* that minimizes the sum of the squared errors between actual and predicted observations.

One of the most popular techniques used for state estimation in real time systems is based on Kalman Filter. The Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. In internet of things domain, Kalman filter can be used for source localization and tracking applications (Li, Ekpenyong & Huang 2006, Chen et al. 2005) . The algorithm works in two steps, prediction step and updating step. In a former step, it produces the estimates of the current state variables with uncertainties associated with the estimated values. In the later step, these estimates are updated using weighted average, with more weight given to estimates with high certainty.

3.2.4. Analytics close to the Data Store

In order to get true value out of our data, we propose modelling 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 analyse 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 analyse 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.

3.3 Metering and Telemetry

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 analyse 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. 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. Telemetry has CANBUS.

3.4 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.

3.4.1. 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.

3.4.1.1. 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.

3.4.1.2. 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.

3.4.1.3. 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.

3.4.1.4. 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.

3.4.1.5. ZigBee

Unlike WiFi that has been developed with focus on high speed data transfer, ZigBee aims at control networks, where application require only low data rate, long battery life and secure networking. These features, together with easy scaling up to 1024 nodes, communication range up to 200 meters and low cost open standard makes it ideal choice for many IoT applications.

3.4.2. 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.

3.4.2.1. 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.

3.4.2.2. 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.

3.4.2.3. 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. The most 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.

3.4.2.4. 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.

3.4.2.5. 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.

3.4.2.6. State machines

State machines represent a mechanism to model and track complex behaviour 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 behaviour of different actors/devices that can be described as a series of transactions between states.

3.4.3. 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.

3.4.4. 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:

3.4.4.1. 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.

3.4.4.2. 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.

3.4.5. 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

3.4.6. CEP data persistence (post processing to detect behaviour 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, analysing 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.

3.4.7. 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. En 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.

3.4.8. 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:
		  ○ Bandwidth
		  ○ Latency
		  ○ Memory
		  ○ Processing power
		• Data out of order

3.5 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 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 its experience and in case of faulty or effected node reading, it will gain bad reputation and its effect on the final decision will be slowly reduced until it gains its 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.

3.5.1. Trust and Reputation Techniques

Trust and Reputation (T&R) plays a key role in the social relationships. In fact, our society it 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 Trust and Reputation. 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 Internet of Things, 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…, it is necessary to adopt another mechanisms. In this sense, Trust and Reputation technics offer one of the best ways to keep security levels and let the system works autonomously, isolating those nodes which present misbehaviour. The misbehaviour 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.

3.5.2. 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 misbehaviour 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 neighbour 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.

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

RFSN has two main components called Watchdog and Reputation system. Watchdog monitors the actions of neighbouring 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.

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

In ATSN, agent nodes monitor the behaviour of sensor nodes and classify their behaviour as bad behaviour or good behaviour. 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.

3.5.5. 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 neighbours. 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.

3.5.6. 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.

3.5.7. 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 neighbours.

- 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 misbehaviour, selfish and malicious behaviour although it focuses in selfish behaviour, and it is applied to Route Discovering and Packet Forwarding functions.

3.5.8. 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 behaviour 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 analyse 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 in table 1.

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

3.5.9. 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.

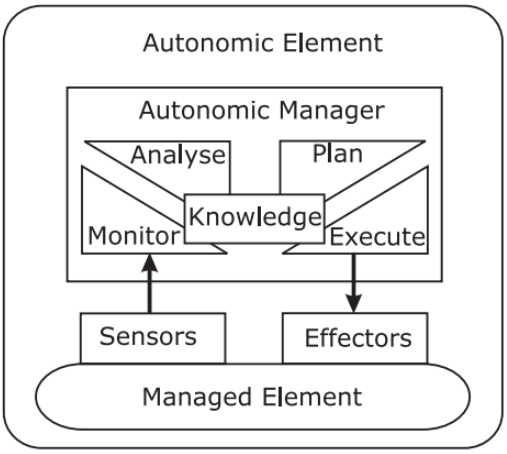
3.6 Autonomic Computing and Distributed Artificial Intelligence

3.6.1. 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 bellow:



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.

3.6.2. 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/Analyse/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.

3.6.3. 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 characterised by:

- A set of Believes: that represent the knowledge that the agent has about its environment, i.e. the knowledge that potentially influence its behaviour;
- A set of Desire: 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.

3.6.4. 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
- (DF) which is an agent registry
- Yellow Page service (for look up) etc…

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

3.6.5. 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]

I have seen 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 focussing within the clusters only, coming with a variety of solutions. Mpitziopoulos *et al*. provides 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.

3.6.6. 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[1] 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

3.6.7. 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 analyse 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 explicite.

3.7 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 behaviour 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

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

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 behaviour of systems.

3.8 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:

3.8.1. Graph based modelling

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.

3.8.2. 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.

3.8.3. 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.

3.8.4. 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.

3.9 Modelling Languages

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

3.9.1.1. 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

3.9.1.2. 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

3.9.1.3. 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.

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

3.9.2.1. 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 serialised in XML and also in Json (see https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-json/index.html)

3.9.2.2. 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.

3.9.2.3. 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 an higher expressive power than pure RDF, consequently not any N3 statement can be serialized in RDF/XML.

3.9.2.4. 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.

3.9.2.5. 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.

3.9.2.6. 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 serialised 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/)

3.9.2.7. 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 optimised 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.

3.9.2.8. 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	Yes/n	No	No	No	Yes	Yes	Yes	Yes	No	No	No	Yes

source	o											
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

3.9.3. Definition and Management of ontology rules

3.9.3.1. 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 behaviours 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.

3.9.3.2. 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) it 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 OW L 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.

3.10 Cloud storage and Meta-data

Cloud Storage refers to a virtualised 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 centres 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.

3.11 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.

3.12 Security

3.12.1. Security principles

Security refers to the degree of resistance against or protection against harm. In modern days 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.

3.12.2. Hardware security

Today security is an add-on to existing hardware architectures. This approach has led to some effective protection mechanisms against some more common security attacks. New advanced 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. 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 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 [11] 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 [12] 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.

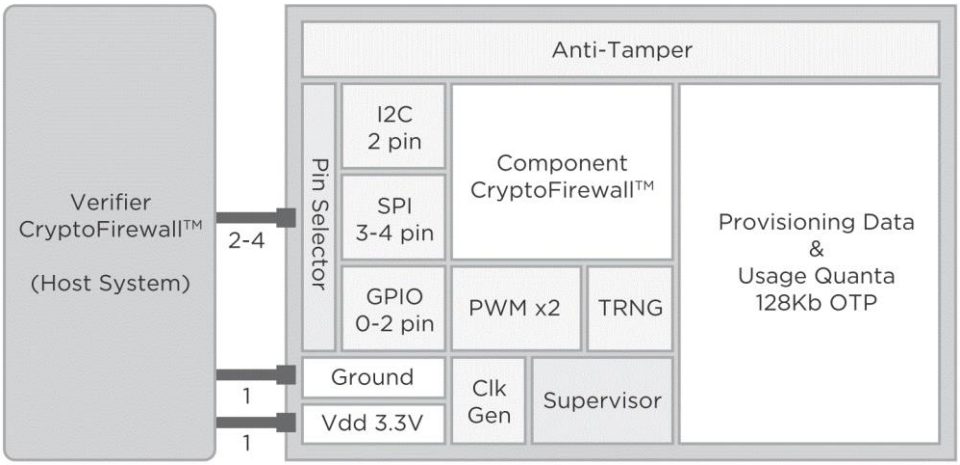


Figure 1. Marvell PA800 block diagram

Barco Silex [13], 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.

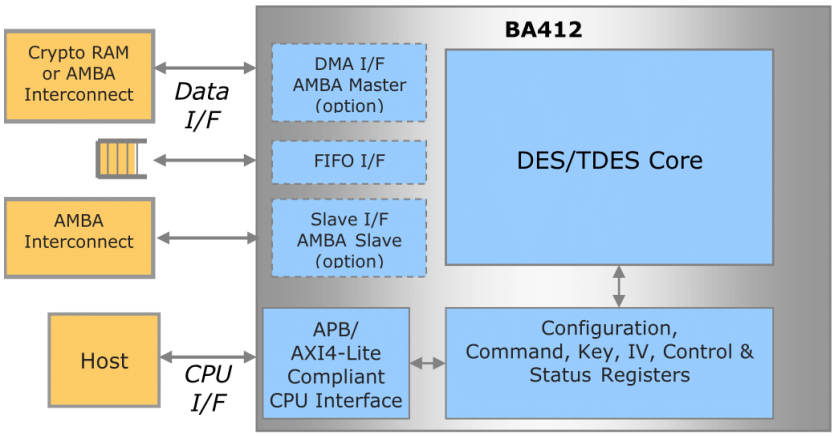


Figure 2 General description of BA412 IP

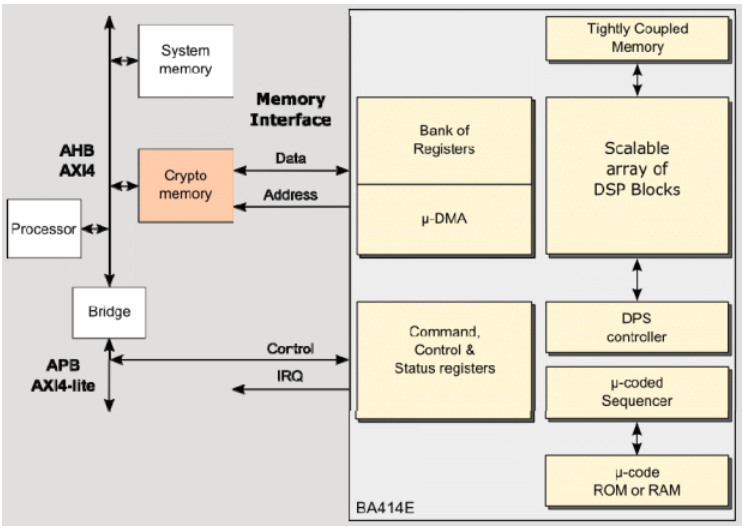


Figure 3 BA412 IP block diagram

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.

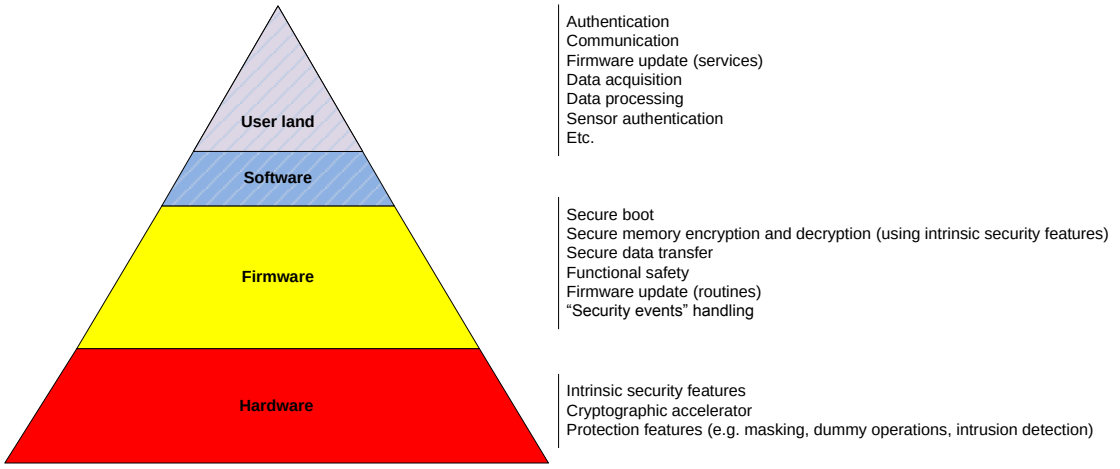


Figure 4 Root of trust

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” [10], 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.

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

3.12.3. 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 [4], 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 [5] 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 of Albaroodi et al. [6], investigated the security issues in PaaS and IaaS, 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 [7].

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 [8]. 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 [9]. 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.

3.12.4. Privacy in IoT

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

3.12.4.1. Platform for Privacy Preferences Project (P3P)

Privacy policies [14] 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.

3.12.4.2. Privacy Proxies and Privacy-aware database

A different approximation to the privacy problem solution is made by privacy proxies [15].

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.

3.12.4.3. Virtual Private Networks

A *Virtual Private Network* (VPN) [16] [17] 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.

3.12.4.4. Transport Layer Security

Transport Layer Security (TLS) [16] [18] [19] 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.

3.12.4.5. DNS Security Extensions

This mechanism implemented in the application layer works by digitally signing records for DNS lookup [16] [20] [21] 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.

3.12.4.6. Onion Routing

Onion Routing [16] [22] [23] 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.

3.12.4.7. 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 [16] [24] 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.

3.12.4.8. Peer-to-Peer systems

A further method to increase security and privacy are *Peer-to-Peer* (P2P) [16] [25] 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.

3.12.4.9. 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. [26]

3.12.4.10. k-Anonymity

k-Anonymity [27] 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.

3.12.4.11. 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 [28].

3.12.4.12. 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 [29] 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.

3.12.4.13. The butterfly method [30] [31]

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.

3.13 Intelligent Traffic Model

Predicting Bus arrival time and modeling Bus route using GPS data from the bus is not a new research topic and researchers have been working on it for few years now. But with the advent of technology and sensor networks everywhere, it opens new areas of research. Exploring traffic models in the context of big data is rather new field. Data from several sensors on bus, data from passengers cell, GPS data of other cars, traffic lights information, population characteristics and weather sensors can all contribute to an intelligent Traffic model which can predict the passengers arrival time more accurately. In (Sun et al. 2007) authors discussed that three types of predicting models can be used to predict the location of bus at any time instant. These are 1) Model based on Historical data 2) Multi linear Regression Models 3) Artificial Neural Network Models

In their work, they derived a model for a single bus route using historical data. They divided the route into small segments containing nodes with known GPS coordinates. Bus GPS data was sampled after regular intervals and using map matching algorithms were matched to particular road segments. Data was collected over two weeks time to make the model accurate. It certainly helped to improve the prediction time for the arrival of buses. But there model was only based on GPS readings of the bus. If any incident occurs down the route, it puts the bus into unknown state. It also did not consider the effect of weather and time of day on traffic models.

In (Rzeszotko, Nguyen 2012), the authors applied the machine learning concepts to predict the route and velocity of cars through the streets of Warsaw. Data from the company TomTom was used which gives the location of the cars and instantaneous velocities of the cars at specific times. The authors used regression techniques to predict the route of particular car and to locate its position in time. After that the authors used propagation neural networks for approximating the average car velocities driving through particular streets. The authors did this work for a competition organized by TomTom for promoting research in traffic prediction models.

In (Zhang, Xu & Liao 2013), authors discussed two most common approaches for traffic estimation based on GPS data using historical data analysis. These are a) The Naïve Method in which all records are aggregated with equal methods. b) And the sliding window sampling method in which only most recent records are preserved. After it, they proposed a novel weighted approach with increasing the weights of most recent records and compared with the already discussed approaches. They demonstrated the effectiveness and feasibility of their approach compared to existing techniques on a field experiment data set.

In (Kong et al. 2013), authors improved the efficiency of traffic state estimation using GPS data by improving the map matching algorithms for plotting GPS data on a digital map. They proposed a new approach for constructing exact digital GIS-T map. They used following two methods for traffic state estimation on the basis of their digital map. 1) The curve based fitting Method. 2) The vehicle-Tracking-based Method.

In (Mak, Fan 2014), the authors used algorithm fusion method to detect an incident on Melbourne freeways automatically. They obtained relative high detection rate of above 80% in

their algorithm when implemented practically. Such automatic detection of an incident by examining the flow of traffic can contribute to more intelligent and accurate traffic model.

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 flavour. 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 5 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.

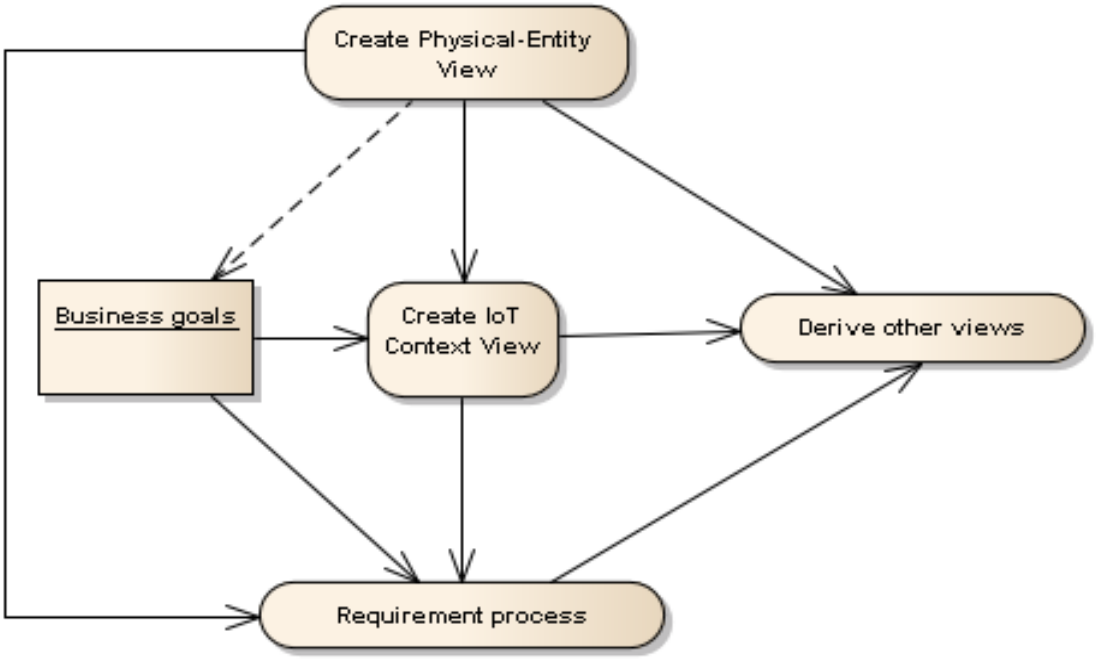


Figure 5: Simplified view of the architecting process

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 flavour 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:** emphasise 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[2]
- 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 ignored in a first step
- 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 (v1) is the first iteration of the Requirement list. It will be refined continuously during the project life spans and the missing column will be populated as the Architecture work makes progress. This list will also be checked regularly in order to ensure that no requirement is left behind.

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

5 References

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.

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

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

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/

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

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

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

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

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.

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

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

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

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

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

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.

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/

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

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

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

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

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.

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

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

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

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

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

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/

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/

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

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.

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

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.

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.

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

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

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).

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.

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.

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

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

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

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

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

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

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

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

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).

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/

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

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)

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

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

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

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

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

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

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

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)

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

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

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

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)

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.

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, H.S. Seo 2008)

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

Blatt, D. & Hero, A. 2004, *Distributed maximum likelihood estimation for sensor networks,* IEEE, NEW YORK; 345 E 47TH ST, NEW YORK, NY 10017 USA.

Bracio, B.R., Horn, W. & Moller, D.P.F. 1997, "Sensor fusion in biomedical systems", *Proceedings of the 19th Annual International Conference of the Ieee Engineering in Medicine and Biology Society, Vol 19, Pts 1-6: Magnificent Milestones and Emerging Opportunities in Medical Engineering,* vol. 19, pp. 1387-1390.

BROWN, C., DURRANT-WHYTE, H., LEONARD, J., RAO, B., AND STEER, B 1992, "Distributed data fusion using Kalman filtering: A robotics application. In Data Fusion in Robotics and Machine Intelligence", .

Carney, D., Cherniack, M., Tatbul, N. & Zdonik, s. 2002, "Monitoring Streams - A New Class of Data Management Applications", .

Chandrasekaran, S., Cooper, O., Reiss, F. & Shah, M. 2003, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World", *Proceedings of the 2003 CIDR Conference*.

Chen, H., Deng, P., Xu, Y. & Li, X. 2005, *A robust location algorithm with biased extended Kalman filtering of TDOA data for wireless sensor networks,* IEEE, NEW YORK; 345 E 47TH ST, NEW YORK, NY 10017 USA.

Klan, D., Katja, H. & Kai-Uwe Sattler 2009, "Developing and Deploying Sensor Network Applications with AnduIN", *DMSN' 09, August 24, 2009*.

Klan, D., Karnstedt, M., Hose, K., Ribe-Baumann, L. & Sattler, K. 2011, "Stream engines meet wireless sensor networks: cost-based planning and processing of complex queries in AnduIN", vol. 29, no. 1-2.

Kong, Q., Zhao, Q., Wei, C. & Liu, Y. 2013, "Efficient Traffic State Estimation for Large-Scale Urban Road Networks", *Ieee Transactions on Intelligent Transportation Systems,* vol. 14, no. 1, pp. 398-407.

Li, T., Ekpenyong, A. & Huang, Y. 2006, "Source localization and tracking using distributed a synchronous sensors", *Ieee Transactions on Signal Processing,* vol. 54, no. 10, pp. 3991-4003.

Mak, C.L. & Fan, H.S.L. 2014, "Algorithm fusion method to enhance automatic incident detection on Melbourne freeways", *Transportation Planning and Technology,* vol. 37, no. 2, pp. 169-185.

Nowak, R. 2003, "Distributed EM algorithms for density estimation and clustering in sensor networks", *Ieee Transactions on Signal Processing,* vol. 51, no. 8, pp. 2245-2253.

Rzeszotko, J. & Nguyen, S.H. 2012, "Machine Learning for Traffic Prediction", *Fundamenta Informaticae,* vol. 119, no. 3-4, pp. 407-420.

Schmitt, T., Hanek, R., Beetz, M., Buck, S. & Radig, B. 2002, "Cooperative probabilistic state estimation for vision-based autonomous mobile robots", *IEEE Transactions on Robotics and Automation,* vol. 18, no. 5, pp. 670-684.

Sun, D., Luo, H., Fu, L., Liu, W., Liao, X. & Zhao, M. 2007, "Predicting bus arrival time an the basis of global positioning system data", *Transportation Research Record,* , no. 2034, pp. 62-72.

T.K. Kim & H.S. Seo 2008, "A trust model using fuzzy logic in wireless sensor network", *Proceedings of World academy of Science Engineering and Technology*.

Tatbul, N., Ahmad, Y., Cetintemel, U., Hwang, J., Xing, Y. & Zdonik, S. 2008, "Load management and high availability in the Borealis distributed stream processing engine", *Geosensor Networks,* vol. 4540, pp. 66-85.

Zhang, J., Xu, J. & Liao, S.S. 2013, "Aggregating and Sampling Methods for Processing GPS Data Streams for Traffic State Estimation", *Ieee Transactions on Intelligent Transportation Systems,* vol. 14, no. 4, pp. 1629-1641.

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

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

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

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

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

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

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

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

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

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

http://www8.cs.umu.se/research/ifor/dl/Control/Fuzzy%20Logic%20in%20Control%20Systems%20Part%20I.pdf

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

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.

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.

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.

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

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.

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.

Security references

[1] Wilson, T. (2009). Security is chief obstacle to cloud computing adoption, study says,. Retrieved from http://www.darkreading.com/ securityservices/security/perimeter/showArticle .jhtml?articleID=221901195

[2] Messmer, E. (2009). Are security issues delaying adoption of cloud computing. Retrieved from http://www.networkworld.com/news/ 2009/042709-burning-securitycloud-computing.html

[3] Mitchel, R. L. (2009). Cloud storage triggers security worries,. Retrieved from http://www.computerworld.com/s/article/340438

[4] 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

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

[6] Albaroodi, Hala, Selvakumar Manickam, and Parminder Singh. "CRITICAL REVIEW OF OPENSTACK SECURITY: ISSUES AND WEAKNESSES."Journal of Computer Science 10.1 (2013): 23.

[7] Server side encryption blueprint for OpenStack Swift, [Online], Available: https://wiki.openstack.org/wiki/Swift/server-side-enc

[8] “AWS Compliance”, [Online], Available: http://aws.amazon.com/compliance/.

[9] “Amazon Web Services: Risk and Compliance”, [Online], Available: http://media.amazonwebservices.com/AWS_Risk_and_Compliance_Whitepaper.pdf

[10] “Common Criteria Portal”, [Online], Available: http://www.commoncriteriaportal.org/

[11] “Weightless – a hardware platform for IoT”, [Online], Available: http://www.weightless.org/

[12] “Marvell”, [Online], Available: http://www.marvell.com/

[13] “Barco-Silex”, [Online], Available: http://www.barco-silex.com/

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

[15] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing," *Institute of Information Systems, ETH Zurich.*

[16] R. H. Weber, "Internet of Things – New security and privacy challenges," *ScienceDirect,* 2010.

[17] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Virtual_Private_Networks.

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

[19] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Transport_Layer_Security.

[20] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "DNS Security Introduction and Requirements," 2005.

[21] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/DNS_Security_Extensions.

[22] D. Goldschlag, M. Reed and P. Syverson, "Onion Routing for Anonymous and Private," 1999.

[23] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Onion_Routing.

[24] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Private_Information_Retrieval.

[25] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Peer-to-peer.

[26] M. Langheinrich, "Privacy by Design - Principles of Privacy-Aware," *Distributed Systems Group Institute of Information Systems, IFW Swiss Federal Institute of Technology, ETH Zurich.*

[27] V. Shmatikov and A. Narayanan, "Privacy and Security: Myths and Fallacies of “Personally Identifiable Information”," *viewpoints,* 2010.

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

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

[30] J. Pei, Y. Tao, J. Li and X. Xiao, "Privacy Preserving Publishing on Multiple".

[31] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Quasi-identifier.