SEVENTH FRAMEWORK PROGRAMME

Networked Media

Specific Targeted Research Project

SMART

(FP7-287583)

Search engine for MultimediA environment generated contenT

D4.1 SMART Distributed Knowledge Base and Open Linked Data Mechanisms

Due date of deliverable: 01-11-2012

Actual submission date: 06-11-2012

Start date of project: 01-11-2011 Duration: 36 months

Summary of the document

Code:	**D4.1 SMART Distributed Knowledge Base and Open Linked Data Mechanisms**
Last modification: **State:**	02/11/2012 Final
Participant Partner(s): **Author(s):** **Fragment:**	AIT, Imperial, ATOS, GLA, Prisa Aristodemos Pnevmatikakis, Nikolaos Katsarakis, Regis Riveret, Dyaa Albakour, Tomas Pariente Lobo, Karim Moumene NO
Audience:	☒ public ☐ restricted ☐ internal
Abstract:	This document contains the first version of the D4.1 SMART Distributed Knowledge Base and Open Linked Data Mechanisms. This document will be updated in month 27 (Jan. 2014).
Keywords:	• Metadata, Linked Data, RDF, XML, JSON • Knowledge-base, non-SQL database
References:	DoW

Table of Contents

1 Executive Summary

1.1 Scope

The SMART system is about offering users information fused from the physical world of sensors and the virtual world of social networks and Linked Data. The local entities that collect and fuse the information from these sources for a given location are the edge nodes. This document was originally designed as a companion to the edge node software release. Instead, we now see a twofold purpose: on the one hand we introduce the edge node architecture, data and capabilities, while on the other hand we provide an installation and usage manual for the edge node software components.

This document will be accompanied by a second version, due on January 2014. Hence some sections of the present document do not contain the information, but pointers to future work.

1.2 Audience

The handling of metadata in a SMART edge node is of great interest to a number of stakeholders including:

- **Developers of the SMART open source project**: SMART open source developers will understand the different metadata feeds, what information they bring into the system and how to interface with them (generate or consume them).
- **The Open Source Community**: This document will serve as a manual for installing and using the edge node.
- **SMART project members (notably members working in WP5 and WP6 of the project)**: The present deliverable will provide valuable insights to all SMART project members as to how to employ the available metadata in the search and application layers.

1.3 Summary

From the functionality perspective the SMART edge nodes serve as the local source of information to the SMART system. The architecture supports metadata collection from diverse sources, fusion and reasoning upon them. The metadata, either raw or processed are stored in the edge node database, which is part of the distributed SMART knowledge base. The metadata are streamed, either within the different edge node modules, or towards the search layer from the database. All this process is demonstrated by an early proof-of-concept.

From a software implementation perspective, the SMART edge node comprises (i) components that process the data streams from the physical world of sensors and the virtual world of social networks and Linked Data, (ii) components that format the metadata feeds and ensure the communication to and from the database and finally (iii) components about reasoning. The two versions of this document are about the formatting and communication components, and the database setup.

1.4 Structure

The chapters in this document serve either as the manual for the software implementation, or as the edge node functionality presentation.

The document is structured as follows:

- Chapter 2 provides an introduction to the document and puts it in the scope of the rest of SMART documentation.
- Chapter 3 discusses the components and the software architecture of the edge node. As such it serves both as the manual for the software implementation, and as the edge node functionality presentation.

- Chapter 4 describes the different metadata types, how they are obtained and formatted into a report. Hence it presents edge node functionality.
- Chapter 5 discusses the external Linked Data the SMART system will be accessing. This part of the edge node functionality is still work in progress and will be enhanced in the second version of the document.
- Chapter 6 describes the database, how feeds are organised in it and how metadata are written into it and are read from it. The chapter contains installation instructions for needed third-party components and function reference for the database interfaces. It is part of the manual for the software implementation.
- Chapter 7 discusses the proof-of-concept. Its purpose is to present an early end-to-end system demonstrating the capabilities and potential of SMART. The presentation is centred on the edge node, hence it is part of the edge node functionality presentation.
- Chapter 8 concludes the deliverable.

2 Introduction

Traditional search engines offer information from documents in the Web. SMART is about real-time information gathered from the neighbourhoods of the cities we live in. This information is obtained by processing feeds from the physical world of sensors and the virtual world of social networks and Linked Data, turning those into metadata streams and fusing them together into interesting high-level events.

To do so the SMART system needs to be able to sense the neighbourhoods of the cities, and to collect information about them from the cloud for the Linked Data and the social networks.

2.1 The SMART edge node

The SMART edge node provides exactly this local functionality: It senses the pulse of the neighbourhoods by collecting and fusing the information from diverse sensors and networks for the given location.

Throughout the project duration edge nodes will be installed by the partners in the city of Santander, forming outdoors edge nodes, and also at Athens Information Technology, forming indoors edge nodes. Two such nodes are already in operation, as discussed in chapters 6 and 7.

A handful of nodes can only serve to demonstrate the potential of SMART. For the system to be functional we need multiple edge nodes, contributed by the community. For this it is important to have a take-up of the SMART open source project. This document serves exactly this purpose: By demonstrating the capabilities of the SMART edge node and providing for a software implementation manual, it is one part of the SMART documentation that will allow people outside of the consortium to setup their functional edge nodes in the way their metadata will be accessible by the SMART system.

2.2 Relation to the rest of the documentation

The edge node, being the source of local information of the SMART system, is partly described in many different SMART documents.

2.2.1 The edge node in the SMART deliverables

The past deliverables related to the edge node and hence to the material found in this document are:

- D2.1: The requirements are collected. In chapter 3 the relevant requirements are shown to be fulfilled by the selected edge node architecture.
- D2.2: The use cases are described. The proof-of-concept scenario presented in chapter 7 is an adaptation of the live news use case adapted for indoors sensors.
- D2.3: The architecture of the complete SMART system is described. Parts of chapter 3 can be found there.
- D3.1: The data knowledge representation is detailed. The input streams are turned into metadata feeds to be handled by the edge node. The representation of the streams in D3.1 is used in chapter 4 to format the edge node metadata reports and in chapter 6 to interface to the database.

The edge node description will also continue in:

- All the versions of deliverables D3.2, 3.3 and 3.4. There, the perceptual processing algorithms yielding the perceived metadata are to be detailed. These can be processing stage components of the edge node, but are not part of the open source software, since the algorithms implemented are proprietary.
- The second version of D4.1. That will expand on this document. Usage of external Linked Data and configuration mechanisms will be discussed there.
- D4.2: The social networks and sensor networks integration. This is about the extraction of the filtered metadata (from social networks) and the fusion of perceived and filtered metadata in the

intelligent fusion manager.
- The two versions of D4.3: There the complete edge node installation manual is to be provided.

2.2.2 The edge node in Track

The SMART open source software uses Trac for its documentation. Please visit our documentation at: http://opensoftware.smartfp7.eu/projects/smart. The parts of this document that comprise the software implementation manual can also be found there, under the edge node section: http://opensoftware.smartfp7.eu/projects/smart/wiki/EdgeToC.

3 Edge node architecture

3.1 System

The edge node is the interface of the SMART system to the physical world of sensors, as well as the conceptual world of the cloud (Linked Data and social networks).

Accordingly to the uses cases and the overall goals of the project, the SMART edge nodes have to process continuously flowing data from a large number of geographically distributed and heterogeneous sources at unpredictable rate, to timely produce new metadata feeds that are to be used in answering complex queries.

Instead of Data Based-Management Systems (DBMS) designed to work on persistent data, or Data Stream Management Systems (DSMS) executing standing queries based on common SQL operators, where little semantics is attached to the data being processed, the SMART edge nodes is rather meant to belong to Complex Event Processing (CEP) class of systems. In such systems where precise semantics is given to the information items being processed, events or situations of the real world are detected, combined using complex patterns in terms of understandable higher-level events and notified to users. CEP systems can be seen as an extension to traditional publish-subscribe systems, which allow subscribers to express their interest in composite events. Though CEP systems are usually meant to be real-time in the sense that continuous streams of data are processed on the fly, the SMART edge nodes contain a database whose data shall be further processed in a batched fashion by the reasoning module to derive high-level information

3.1.1 Requirements

Edges nodes have to fulfil the following requirements that we classify into functional and technical requirements. The functional requirements are:

1. The rules defining the complex event processing can be configured by rule managers.
2. The rules can be automatically learned.

Technical requirements of the SMART edge nodes are basically those of any CEP system, and thus:

3. Edge nodes have to deal with geographically distributed and heterogeneous sources.
4. Edge nodes have to deal with flows or streams of data. The information flows from the sensors to the edge nodes does not mean that it may be persistently stored in into some databases for future uses.
5. Edge nodes have to cater for information arriving at unpredictable rates.
6. Edge nodes have to provide timely responses.
7. Edge nodes have to optimize parameters, like bandwidth utilization and end-to-end latency.

The edge node architecture is presented in the remaining of this chapter, both from a component view and a software stages view. Every one of these seven requirements is facilitated by the chosen edge node architecture as explained in the following sections.

3.2 Component view of the edge node

The architecture of the edge node is given in Figure 3.1. This is the component view of the edge node, grouped together in modules for:

- Information sources: the design allows a diversity of the sources (physical and conceptual from the cloud), fulfilling requirement 3;
- Multimedia content indexing, where the metadata are handled. There is no need to store the actual data for later processing, so requirement 4 is fulfilled;
- Knowledge base & Linked Data;
- Edge content selection & delivery.

The following subsections give an overview of these components.

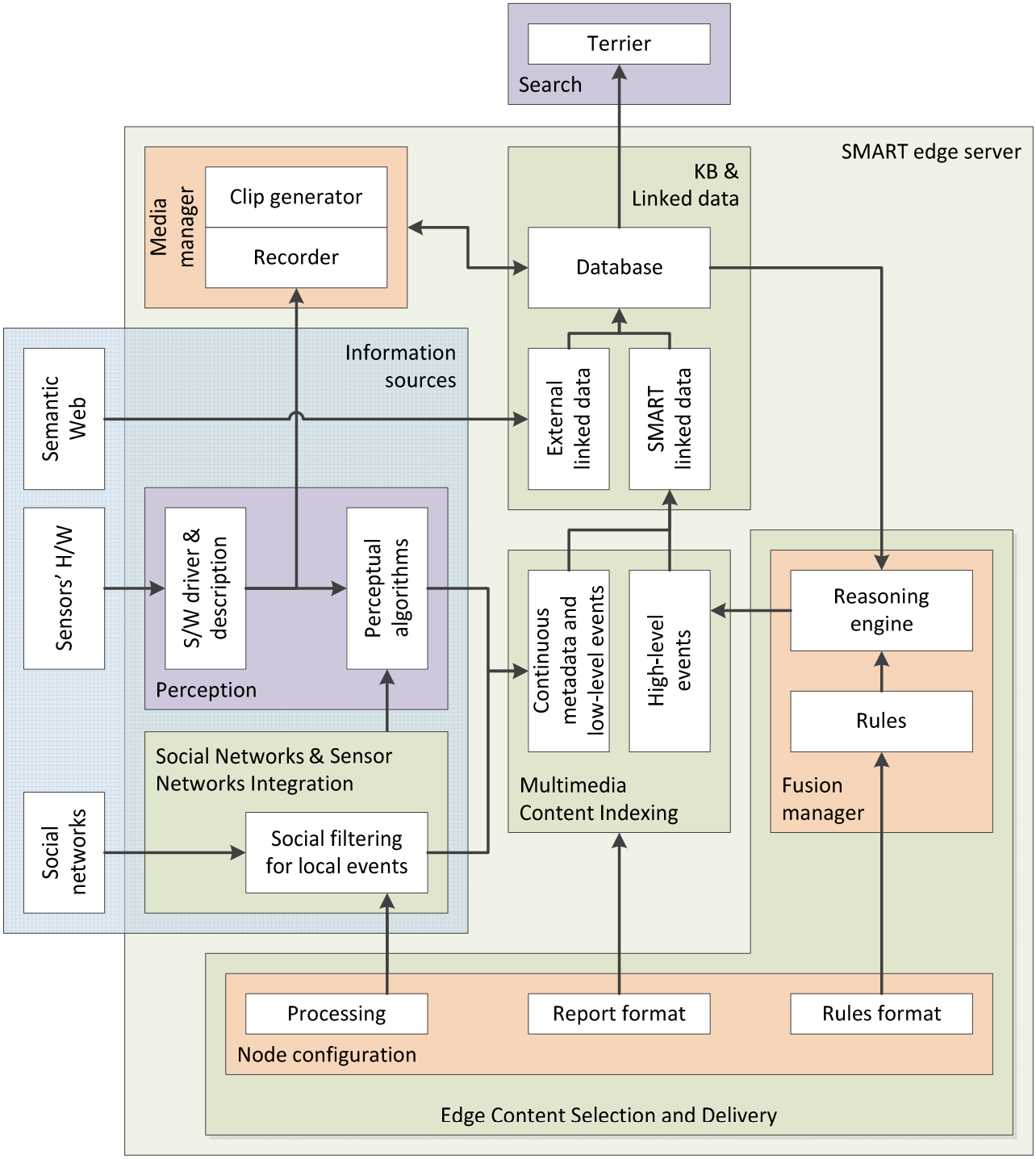

Figure 3.1. Edge Node Architecture

3.2.1 Multimedia Content Indexing (MCI)

The Multimedia Content Indexing module is about ensuring that pieces of information provided by sensors and the Intelligent Fusion Manager are formatted with respect to some common format such as RDF and Linked Data. The metadata can appear at any rate and they populate their respective data feed, hence requirement 5 is fulfilled.

This module is detailed in chapter 4 of this deliverable.

3.2.2 Knowledge Base (KB)

The pieces of information relating low-level and high-level information formatted in the MCI module and the information mined from the Semantic Web are gathered into a knowledge base under the form of Linked Data. Information from the Semantic Web is captured as external Linked Data while low-level and high-level events formatted in the MCI module are gathered as internal SMART Linked Data. Internal and external Linked Data are then stored into a database for future use.

This module is detailed in chapter 6 of this deliverable. The external Linked Data are outlined in chapter 5. More information on these will be given in the second version of this deliverable, due January 2014.

3.2.3 Intelligent Fusion Manager (IFM)

The Intelligent Fusion Manager is meant to (i) infer high-level information/events from low-level information/events via some declarative recognition patterns (ii) learn these meaningful patterns (hence requirement 2 is fulfilled).

External and internal Linked Data from the knowledge base are fused to infer high-level events which are passed to the MCI module to ensure their proper format before being send to the knowledge base..

This module is part of Deliverable 4.2, “Social Networks and Sensor Networks Integration,” due April 2013. Its software implementation will be described there.

3.2.4 Configuration Manager (CM)

The configuration manager allows agents to configure: (i) the reasoning engine, in particular with respect to rule-based patterns (ii) the reports with the metadata to be stored to the edge node KB, and (iii) the perceptual components. These two levels of configuration fulfil requirements 1 and 7.

More information on the configuration manager, and its software implementation, will be given in the second version of this deliverable, due January 2014.

3.2.5 Media Data Manager (MDM)

The main mission of the Media Data Management unit is to be able to provide multimedia (audio, video) contents to the users of SMART. The MDM must (i) store the multimedia contents from the multimedia sensors, (ii) extract clips of interest based on the inferred events, and (iii) adapt the contents to the most common terminals.

More information on the configuration manager, and its software implementation, will be given in the second version of this deliverable, due January 2014.

3.3 Software view of the edge node

The edge node components shown in the architecture block diagram of Figure 3.1 are collected in three software stages (see Figure 3.2).

The “processing stage” components have to ensure that they produce metadata at the rate the sensor data streams are arriving, allowing for requirement 5 to be fulfilled. From the processing stage components, the media manager’s clip generator and the Linked Data manager are to be covered in the next version of this deliverable. The social networks manager is to be covered in Deliverable 4.2, “Social Networks and Sensor Networks Integration.” The perceptual components are not part of the open source of SMART, since they comprise proprietary algorithms of the partners. They are described in the relative deliverables of work-package 3.

The “reporting stage” software components and the metadata storage are the core of this version of the deliverable.

The “reasoning stage” components are also to be covered in Deliverable 4.2, “Social Networks and Sensor Networks Integration.”

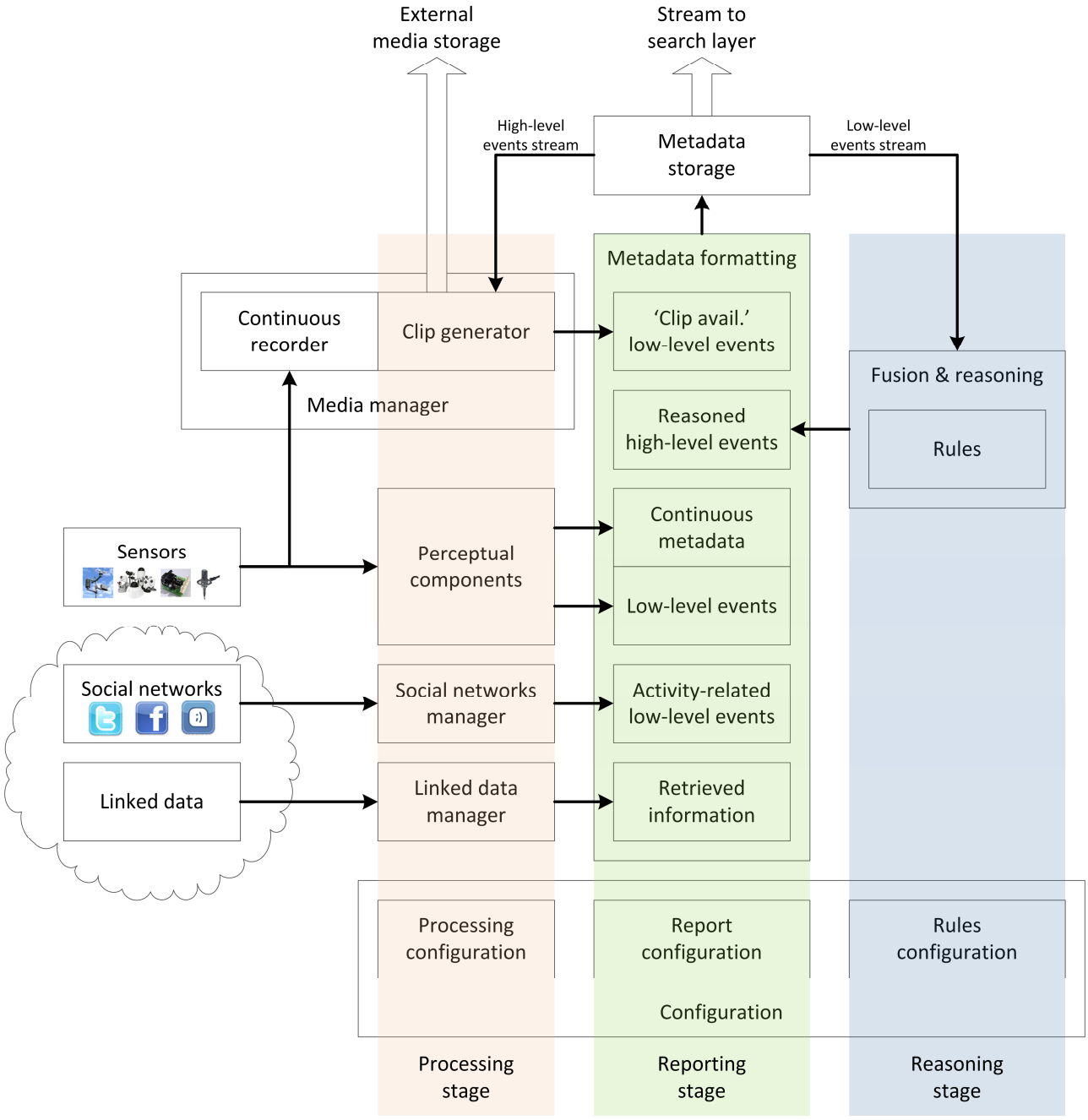

Figure 3.2: The three software stages of the edge node.

Note that the processing-intensive stage of the edge node is the processing one. If the perceptual algorithms can process the sensor streams in real-time, then the rest of the edge node design allows for the metadata to be streamed to the search engine in a timely fashion, fulfilling requirement 6.

4 Metadata (SMART data)

4.1 State-of-the-art

4.1.1 Semantic web & Linked Data

The Semantic Web is a collaborative program led by the World Wide Web Consortium (W3C) that promotes common formats for data/information/knowledge on the World Wide Web.

While the initial Web was about the layout of documents and the links between them, the Semantic Web is oriented towards the semantic description of the contents. The idea is based on the semantic tagging of contents to give them well-defined meaning, so that machines can process those tagged data to infer new explicit information [1].

The semantic tagging of data is performed using languages as Resource Description Framework (RDF), Resource Description Framework Schemas (RDFS), Web Ontology Language (OWL) which are both XML-based. Notice that those languages are based on subsets of first-order logic, notably Description logic whose computational properties are the main assets.

Although the Semantic Web has proven to be valuable in many implementations and projects, its vision remains largely unrealized. A major issue stems from the difficult connections between content. To tackle this issue, Linked Data are meant to overcome this barrier: Linked data is essential to actually connect the semantic web, as explained by Lee [2]:

"The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With Linked Data, when you have some of it, you can find other, related, data."

The procedure to publish and connect data is simple: things are identified with Uniform Resources Identifiers (URIs) and HTTP URIs are used to locate and access those URIs. When an URI is accessed, more useful information using the Semantic Web standards and other HTTP URIs are given.

In conclusion, Linked Data regroup techniques applied to the RDF data model that identify things as URIs and makes them accessible via the HTTP.

4.1.2 Ontologies for IoT

In recent years a lot of attention has been paid to encode sensor descriptions and sensor observation data using ontologies in order to enable a formal representation, easier and programmatic access to sensor resources. Notice that annotating sensor data implies usually the usage of spatial, temporal and domain-specific metadata.

Some of the most widely used ontologies in the scope of IoT are the following:

- SWEET ontology (http://sweet.jpl.nasa.gov/ontology), an upper level ontology focusing on modelling of observed properties, observation based ontologies
- W3C Sensors and observations ontology (SSN): (http://www.w3.org/2005/Incubator/ssn/wiki/Sensor_Discovery_on_Linked_Data The SSN ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment. SSN is been widely adopted by the sensors community.
- Marine Metadata Interoperability project (http://marinemetadata.org/), is an example of a domain-specific ontology designed for oceanographic sensors.
- WGS84 Vocabulary (http://www.w3.org/2003/01/geo/): A W3C effort for describing the location of spatial things in coordinates (latitude, longitude and height) and places. Used in SSN and GeoNames ontologies.
- Time ontology (http://www.w3.org/TR/owl-time/): Another W3C standard ontology to describe time events. It is used in SSN.
- GeoNames ontology (http://www.geonames.org/ontology/documentation.html): This ontology makes possible to add geospatial semantic information to the Web. It is also available as a Linked Data dataset as explained in section 5.

4.2 Edge node processing for metadata extraction

The edge node is the interface of the SMART system to the physical world of sensor and social networks, as well as the conceptual world of the Linked Data cloud. As such, the edge node brings to the SMART system information:

- Perceived from its physical surroundings as they are sensed by a sensor network,
- Filtered from social networks,
- Retrieved from the Linked Data cloud, and,
- Inferred by combining all the above from diverse sources.

4.2.1 Perceived information

Perceived information comes from the environment in the form of continuous metadata streams that result from perceptual processing of the sensor feeds such as cameras, microphones, or environmental sensors. Examples of continuous metadata are the crowd density or air temperature.

Regarding the A/V signals:

- Video signal processing in SMART involves face and people tracking and visual scene analysis. For the latter, algorithms are being implemented for traffic and crowd analysis, both static (density and colour analysis) and dynamic (motion analysis and change detection).
- Audio signal processing in SMART considers audio event classification and speech processing.

The continuous metadata feeds are processed using a threshold to generate low-level events such as the occurrence of a crowded space. The low-level events are represented as binary signals indicating the onset of the event.

Example based on crowd analysis

The crowd analysis perceptual component is issuing the continuous metadata feed of crowd density. The moment this density exceeds a threshold, the low level event ‘place crowded’ is signalled. Similarly, the moment the crowd density falls below another threshold, the low-level event ‘place not crowded’ is signalled.

An example is illustrated in Figure 4.1, which shows the crowd density as a function of time. This has three large peaks, for which the corresponding processed frames indicating the foreground blobs are given. A fourth frame is an example of minor activity. The extracted low-level metadata indicating crowded intervals are also shown as labelled horizontal lines.

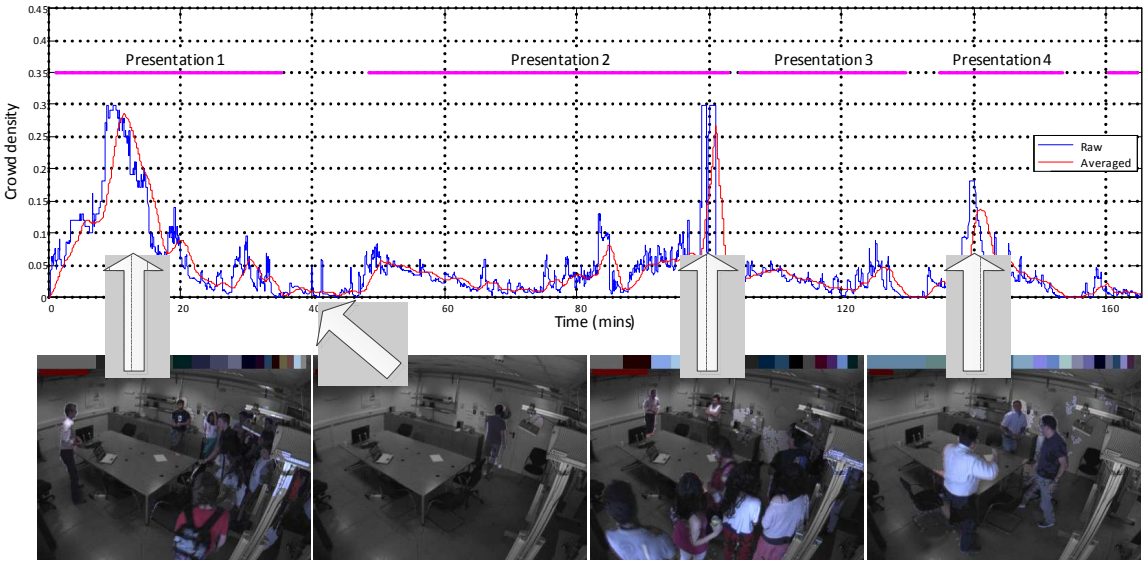

Figure 4.1: Crowd-related continuous metadata and low-level events as a function of time. Some

typical processed frames are also shown.

4.2.2 Filtered information

Filtered information comes from the social networks directly in the form of low-level events. The social network manager monitors social networks like Twitter for mentions of the different aliases of the particular edge node. It then searches for activities associated with it and subsequently monitors these activities in the social network.

The collected text is analysed to understand the association of the activity to the edge node and the details of the activity, like when or what is happening. This social information about activities is stored in the edge node database, to be expressed as low-level events when their designated start and end times come.

Example based on Twitter

There is an edge node at Athens Information Technology (AIT). This node has several aliases, provided at the setup phase. One of them is the Twitter name of AIT, namely @AITathens.

The social network manager searches the social networks for mentions of AIT using all the aliases, and for Twitter the tweets originating from or mentioning of @AITathens are collected.

The activities associated with AIT are then extracted. The simplest way is to collect the hash-tags, and concentrate to the most popular ones. Assume that there is a SMART Demo Event being scheduled and the hash-tag #SMART_demo is used for it. This will be found in some of the tweets, looking for example like:
#SMART_demo is organized by @AITathens on 1/6/2012
@AITathens is hosting #SMART_demo on Friday 1st of June

Such tweets immediately indicate that @AITathens is related to #SMART_demo. Further analysis can reveal that @AITathens is hosting SMART_demo. For this text analysis should be looking for evidence like ‘organising’ or ‘hosting’. This indicates that #SMART_demo is actually happening at the space monitored by the AIT edge node, i.e. the activity should be perceived by the sensors.

After making the connection between the node and the activity, the social network manager will be looking for #SMART_demo independent from @AITathens, in tweets that look like:
#SMART_demo is starting in a week!
Tomorrow is #SMART_demo day!

Analysing this information can reveal the time schedule of the activity. When the right time comes, low-level events will be issued, of the form ‘#SMART_demo is happening’, or ‘#SMART_demo has ended’

4.2.3 Retrieved Linked Data

Information is also retrieved from the Linked Data cloud. The geographical information of the node and its aliases are used to find relevant data. Unlike the social network filtered low-level events, the retrieved Linked Data are not time-critical. Instead they offer information that can help ranking query results.

The external Linked Data and their usage within SMART are detailed in Chapter 5.

Example: SMART as a real-time restaurant guide

Think of the query "restaurant in Santander." To answer this query, SMART needs to have information about which restaurants exist in Santander. Such information can be retrieved from the Linked Data cloud. Answering using such information is just the traditional search engine. The way the results are ranked actually demonstrates the use of SMART:

- SMART can have information about how good the various restaurants are: This needs third party data, either user generated content (by tagging or liking good places) or restaurant critics. Such information can be found in repositories of Linked Data that are external to SMART. As SMART knows the user, the needed information can also be found by filtering feeds from social media of the user’s friends, like Facebook. So ranking can be done by reputation.
- SMART knows the user location. So ranking can be done by proximity.

- SMART knows the user preferences, for example about visiting crowded/lively places. SMART looks for the low level event ‘place crowded’, which is derived by processing of the audio-visual sensor streams. So ranking can be done by taking into account live information.

Apart from just ranking the results, a SMART application can contain all this information in a mashup, using a map and markers for the restaurants, whose size can denote reputation and colour can denote real-time activity at the streets nearby.

4.2.4 Inferred information

Inferred information comes from reasoning over the low-level events (both those from different algorithms perceiving the environment and those from filtering the social networks) and the retrieved Linked Data. These are the high-level events generated by running the reasoning rules.

Example: SMART for presenting what happened

Think of the query "Events at AIT." To answer this query, SMART has to look for information from the @AITAthens edge node. Filtered information about events hosted at AIT is collected there, and is offered to the user ranked by:

- Temporal proximity
- Success of the event. This is actually inferred from the visual and audio low-level events stemming from audio-visual processing of the feeds at the time of the event. Such low-level events are related to visual crowd density and acoustic recognition of applause (positive) or noise (negative).

4.3 Edge node metadata structure

4.3.1 Metadata in RDF

The edge node data are represented as RDF triples of the form [subject, predicate, object]. The RDF is formatted using XML as follows:

```
<subject>
      <predicate1>object1</predicate1>
      <predicate2>object2</predicate2>
      ...
      <predicateN>objectN</predicateN>
</subject>
```

For example the subject can be the crowd; a predicate can be the density with the associated value as an object. Other predicates can be the time of measurement, the colour or the horizontal/vertical speeds. The different subjects and their possible predicates are shown in Table 4.1.

Assume that an edge node has multiple temperature sensors attached. This is not uncommon, for instance a weather station comes with one indoors and another outdoors sensor for temperature (and other measurements). Or assume we have a couple of microphones, or a camera viewing a wide area, that can be logically split into zones. For instance consider a camera viewing the town hall square of Santander, where a stripe at the top of the frame can be assigned to the road (Calle de Jesus del Monasterio), while the rest of the frame to the square itself (Plaza).

In such cases we have different measurements for the same physical quantity, only taken at different locations in the same edge node. We need to report multiple temperatures, crowd densities, or sound intensities, all these coming from different locations in the same wider area. The benefit of reporting all the measurements and not for example a single averaged one is in the increased reasoning detail:

- A road might have traffic all the time, but the plaza next to it is having only the occasional bursts of pedestrians based on the traffic light cycles. The road is crowded but by no means is the plaza.
- Large temperature variations in the same geographic area can indicate hazard, like a fire.

Table 4.1: SMART metadata in RDF triples. The subjects of crowd, any gas from a chemical sensor, environment (from sensor networks) and activity (from social networks) are shown.

Subject	Predicate	Object	Comment
Crowd	Time	Date-Time	Measurement time
	Density	Numeric	Continuous metadata
	Colour	Colour spec (integer part) + frequency (decimal part)	
	Primary horizontal motion	Numeric	
	Primary vertical motion	Numeric	
	Motion spread	Numeric	
	Status	Logical	Low-level event
	Uniform colour	Colour specification	
Audio	Level	Numeric	Continuous metadata
	Applause	Logical	Low-level event
	Crowd		
	Music		
	Traffic		
Gas	Concentration	Numeric	Continuous metadata
	Limit	Exceeding/Normal	Low-level event
Environment	Humidity	Numeric	Continuous metadata
	Temperature	Numeric	
	Dew point	Numeric	
	Wind chill	Numeric	
	Pressure	Numeric	
	Wind average	Numeric	
	Wind gust	Numeric	
	Wind direction	Direction string	
	Rain since last reading	Numeric	
	Rain last hour	Numeric	
	Rain last 24 hours	Numeric	
	Rain last 7 days	Numeric	
	Rain last 30 days	Numeric	
	Is raining	Logical	Low-level event
	Is hot	Logical	
	Is cold	Logical	
Activity	Name	String	Filtered low-level event from social networks
	isActive	Logical	
	Date	Date this event refers to	
	temporalHint	Time this event refers to	

To facilitate reporting multiple similar measurements, we can indicate that they are produced from similar components with different IDs, as defined in the feed description. In RDF representation they can be placed in groups with different IDs as tag attributes. For example:

```
<measurement>
      <crowd ID="calle_monasterio">
            <density>.03</density>
      </crowd>
      <crowd ID="plaza_ayuntamiento">
            <density>.28</density>
      </crowd>
</measurement>
```

The actual sensor locations for the different IDs are defined at feed setup, in order not to be transferring location information at all times. In this particular example, location with ID=”plaza_ayuntamiento” refers to Plaza del Ayuntamiento and with ID=”calle_monasterio” to Calle de Jesus del Monasterio. An example of this approach for crowd density is shown in Figure 4.2.

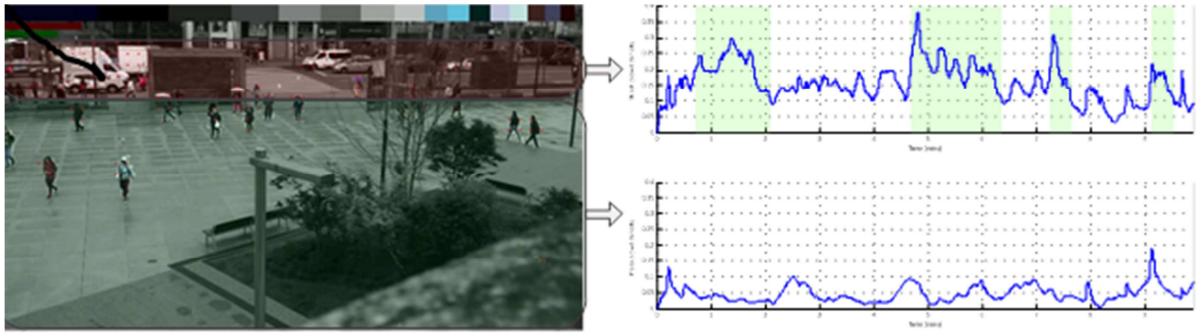

Figure 4.2: Crowd analysis in zones. The upper zone corresponds to a frequently crowded street, while the lower to a plaza that has just the occasional streams of pedestrians.

In the feeds database, the name of the <subject> should correspond to the <Type> tag of the relevant component in the feed description. The <DescriptionTags> tag allows easier semantic representation and should contain a URI signifying the component and one or more URIs describing semantic details like the region of application. Both of these are shown in the example below.

In this example, the two virtual components of the feed necessary for reporting the crowd density at two zones are presented. Both are of type “Crowd”, which is one of the possible subjects listed in Table 4.1. Also both have a common entry in the list of <DescriptionTags>, the “smartfp7:crowd”. The second element in the list is “smartfp7:square” for the virtual component reporting the crowd density at Plaza del Ayuntamiento, indicating that the region being monitored is a square , while it is “smartfp7:road” for the one reporting for Calle de Jesus del Monasterio, indicating that it refers to a road view. Using these URIs, a higher layer can contrast crowds gathering in squares to crowds swarming in the roads.

```
. . .
<Virtual>
      <Name>calle_monasterio</Name>
      <Description>Crowd analysis for camera region focusing on road
(Calle de Jesus del Monasterio)</Description>
      <DescriptionTags>
            <URITags>smartfp7:crowd smartfp7:road</URITags>
      </DescriptionTags>
      <Type>crowd</Type>
</Virtual>
<Virtual>
      <Name>plaza_ayuntamiento</Name>
      <Description>Crowd analysis for camera region focusing on public
square (Plaza del Ayuntamiento)</Description>
      <DescriptionTags>
            <URITags>smartfp7:crowd smartfp7:square</URITags>
      </DescriptionTags>
      <Type>crowd</Type>
</Virtual>
. . .
```

The above listing shows how the users can define the two different zones for the crowd analysis when creating a new feed. As mentioned before, the type “crowd” is the subject in the RDF representation, while the names are the two IDs will correspond to the <crowd> tag attributes.

4.3.2 Elements from established ontologies

For the characterization, linking and usage of Linked Data related to sensors and IoT it is crucial to identify sources for further information and picking the appropriate vocabularies. A sample of candidate suggested vocabularies is the following:

- Generic vocabularies to be used range from the popular Dublin Core (DC) for metadata, Friend Of A Friend (FOAF) for relationships among people, or the Simple Knowledge Organisation System (SKOS) for categorizations and concept maps.
- Relation between resources should also be established using the popular RDFS and OWL properties such as owl:sameAs and rdf:About, amply used in Linked Data.
- For domain specific sensor data, the SSN ontology mentioned in 4.1.2.is a good candidate to characterize the resources.
- For geographical data, the recommendation is to use the WGS84 Vocabulary mentioned in 4.1.2., linked to GeoNames and DBPedia locations.
- For characterization of time events, the best candidate vocabulary is the W3C Time ontology also mentioned in 4.1.2.

In the next version of this deliverable, the suitable elements of these and other vocabularies will be selected and reused in the description of the SMART metadata. Currently this is not done, hence the report presented in the next section uses custom names, which is fine for the internal operation of the SMART system, but is not optimum for the reuse of the SMART metadata in the Linked Data cloud, as discussed in the Linked Data principles of section 5.1.

4.3.3 Report structure

In this section all the metadata structure discussed in the previous sections is put together to show how a report from the edge node looks like.

Assuming a query about a time period during which a feed has provided two measurements, the returned report in XML looks like:

```
<?xml version="1.0"?>
<rdf>
<!-- Namespace Definitions should be included in the RDF tag -->
      <measurement>
            <time>2012-07-26T10:14:10.467Z</time>
            <crowd ID="plaza_ayuntamiento">
                  <density>0.170000</density>
                  <motion_horizontal>0</motion_horizontal>
                  <motion_vertical>0</motion_vertical>
                  <motion_spread>3.600000</motion_spread>
                  <colour>32.125100</colour>
                  <colour>2113632.106200</colour>
                  <colour>2113600.092100</colour>
                  <colour>4218976.066300</colour>
                  <colour>4210752.061800</colour>
                  <colour>2105376.042200</colour>
                  <colour>8256.036100</colour>
                  <colour>2105408.033300</colour>
                  <colour>8413280.030500</colour>
                  <colour>8224.024800</colour>
                  <colour>0.023300</colour>
                  <colour>10535136.022300</colour>
                  <colour>6316160.021400</colour>
                  <colour>4210784.017800</colour>
                  <colour>2113664.017800</colour>
                  <colour>12640480.017000</colour>
            </crowd>
```

```
            <crowd ID="calle_monasterio">
                  <density>0.220000</density>
            </crowd>
      </measurement>
      <measurement>
            <time>2012-07-26T10:14:10.795Z</time>
            <activity>
                  <name>#SMART_Demo</name>
                  <isActive>true</isActive>
                  <date>26-07-2012</date>
                  <temporalHint>10:14</temporalHint>
            </activity>
            <crowd ID="plaza_ayuntamiento">
                  <density>0.170000</density>
                  <motion_horizontal>0</motion_horizontal>
                  <motion_vertical>0</motion_vertical>
                  <motion_spread>2.200000</motion_spread>
                  <colour>32.125100</colour>
                  <colour>2113632.106200</colour>
                  <colour>2113600.092100</colour>
                  <colour>4218976.066300</colour>
                  <colour>4210752.061800</colour>
                  <colour>2105376.042200</colour>
                  <colour>8256.036100</colour>
                  <colour>2105408.033300</colour>
                  <colour>8413280.030500</colour>
                  <colour>8224.024800</colour>
                  <colour>0.023300</colour>
                  <colour>10535136.022300</colour>
                  <colour>6316160.021400</colour>
                  <colour>4210784.017800</colour>
                  <colour>2113664.017800</colour>
                  <colour>12640480.017000</colour>
            </crowd>
            <crowd ID="calle_monasterio">
                  <density>0.250000</density>
            </crowd>
      </measurement>
</rdf>
```

For the first measurement, the feed contains data for the <crowd> virtual component only, while for the second measurement the feed contains data for both the <crowd> and the <activity> virtual components. The various tags in the virtual components are discussed in Table 4.1.

For examples of reports in JSON format, both sent to the database and returned from it, see Section 6.3.

5 External Linked Data

5.1 Linked Data principles

The traditional Web is based on hyperlinks to connect documents into a single global information space. Linked Data mimics this hyper-linking approach from the web by enabling links between items in different data sources. This allows the creation of a single global data space.

Linked Data imposes very few constraints to reach this single global data space. In fact, Linked Data provides a set of guidelines or best practices for publishing, interlinking and consuming data, widely known as the Linked Data principles:

1. Use URI as names for things: Data item that can be linked is any uniquely identified entity regardless it is a real world thing or a digital item.
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards.
4. Include links to other URIs, so that they can discover more things.

There can be several underlying representations in this global data space; however the most widely used and is the RDF data model. Two main types of RDF triples are Literal Triples and RDF Links. While the Literal Triples are used to describe the properties of resources, the RDF Links describe the relationship between two resources.

Therefore the most common way of fulfilling the Linked Data principles uses the following technologies:

1. Use of URIs to identify resources
2. Use http protocol to dereference URIs
3. Resources are represented in RDF and queries expressed using SPARQL
4. Include RDF links to represent relationships

On this basis, Linked Data offers a unifying data model (RDF), a standardised data access mechanism using the http protocol, a data discovery in a single data space based on typed hyperlinks, and an easier integration of data based on existing and shared vocabularies. All these properties make Linked Data quite an appropriate option for publishing, retrieving and enhancing data.

On the other hand, there are three main processes when dealing with Linked Data: Publication, Linking and consumption:

- **Publishing**, data providers add their data to a global data space, which allows data to be discovered and used by various applications. Publishing a data set as Linked Data on the Web involves the following three basic steps:
  - Naming things with URIs and making URIs dereferenceable.
  - Providing useful RDF information (choosing and using vocabularies to describe data: SKOS, RDFS and OWL, DC, FOAF, SIOC, Good Relations Ontology, WGS84...)
  - Publishing data about data, provide metadata about data set i.e.: authorship, provenance metadata, Licenses, Waivers and Norms for Data
- **Linking** data set with other Linked data sets.
- **Consuming**, data that is published on the web following the Linked Data principles: an application should be able to request, retrieve and process the accessed data
  - Discover further information by following the links between different data sources: the fourth principle enables this.
  - Combine the consumed Linked Data with data from sources (not necessarily Linked Data).
  - Expose the combined data back to the web following the Linked Data principles.

SMART aims at taking advantage of existing Linked Data datasets in the domain. Therefore the process of consuming data, by selecting datasets and the appropriate linking and consumption tools are of paramount importance to the project.

5.2 Existing datasets

Since its first steps in 2006, the LOD Cloud (data set published as Linked Data) has grown from 500 million triples in 2007 to around 27 billion in 2010. But the growth is not only about number of triples but also in the range of different topical domain e.g. 26 datasets of media, 16 in Geographic, 26 in Government, or 42 in Life sciences domain [1]. Next figure shows the latest version of the Linked Data cloud.

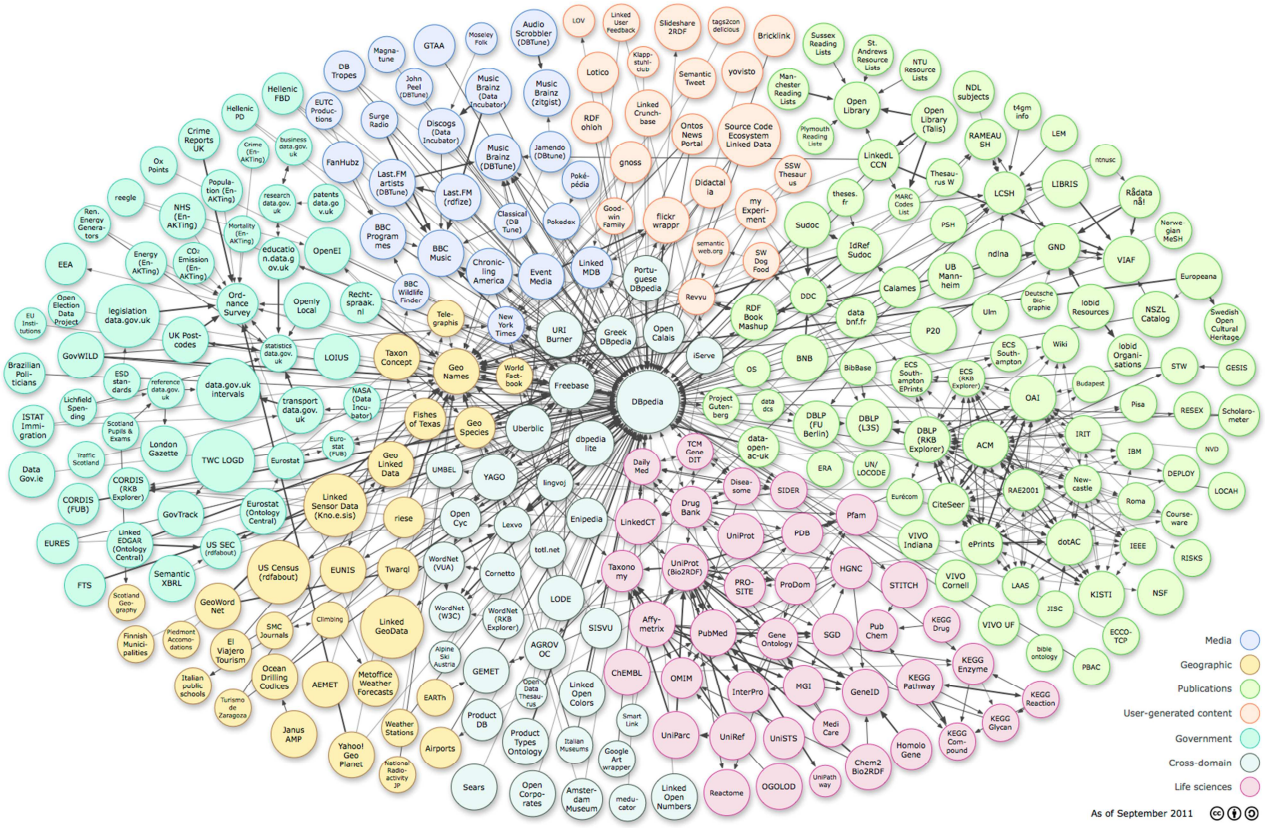

Figure 5.1: The Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

A comprehensive set of the datasets available in the Linked Data cloud can be found in the Linked Open Data cloud list of datasets maintained by the LOD community[2].

5.3 Reusing Linked Data in SMART

SMART could exploit information from existing Linked Data datasets. Examples of datasets are the following:

- General datasets
  - DBPedia: DBPedia provides a RDF view of the Wikipedia content. It is located in the center of the Linked Data cloud, as it provides links to almost all the rest of available datasets. DBPedia can be used via SPARQL using several existing end points, and is also available for download (http://datahub.io/dataset/dbpedia).

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/state/

[2] http://www4.wiwiss.fu-berlin.de/lodcloud/

    - Freebase: Similar to DBPedia, Freebase offers cross-domain community-generated content complementary to DBPedia (http://datahub.io/dataset/freebase).
- Geographic datasets:
    - Geonames (http://www.geonames.org), which is an open-license geographical database that publishes Linked Data about 8 million locations.
    - LinkedGeoData [Stadler12], provides a Linked Data conversion of data from the OpenStreetMap project, which includes information about more than 350 million spatial features. Locations in Geonames and LinkedGeoData are interlinked with corresponding locations in DBpedia, ensuring there is a core of interLinked Data about geographical locations.
    - Several
- Sensor data
    - Linked Sensor Data (http://datahub.io/dataset/knoesis-linked-sensor-data) offers a dataset Datasets for sensors and sensor observations, created at Kno.e.sis Center containing descriptions of 20.000 weather stations and 160 million observations. There are many national agencies providing Linked Data weather time series besides Linked Sensor Data.

The Linked Data functionality is still work in progress. Therefore the list of datasets to be used to enrich the edge nodes will be presented in detail in the second version of the document.

6 Data repository

An integral part of SMART is reasoning, that combines together metadata stemming from diverse types of sensor feeds, and even social networks feeds, effectively integrating them.

Reasoning is based on the history of metadata, over some temporal window. Regarding the sensor streams, this window can be rather short, spanning a few minutes. This is for example the interval needed to understand that some crowd has been forming up in a square of the city of Santander, based on visual and audio feeds.

On the other hand, this window can be very long for metadata coming from social networks. People might be tweeting about gathering to that square over a period of several days, and the system should be able to judge the social importance of the event, based on the analysis of this Twitter feed.

For this reason it is impractical and unsafe to just keep the metadata in the reasoning module memory. We need to store them to address three issues:

- The length of the temporal window makes memory storage impractical.
- The metadata should not be lost in the case of service outages, since they cannot be recomputed. The sensor streams are not saved at all, while the social streams are kept with only limited days of history. For example, the tweets of up to 10 days old can be retrieved.
- The metadata should be organized in a form to be easily retrieved by the reasoning module.

All these three issues are addressed by the use of a database that stores the metadata in an indexed form.

Since the sources of the metadata are diverse, and we also expect third-party users to be able to connect their sensor/processing pairs to an open SMART system, it is unreasonable to expect that a common structure will exist. For this reason the edge node uses Apache CouchDB (http://couchdb.apache.org/), a relatively new no-SQL database.

What no-SQL means is that a CouchDB database lacks a schema [3], i.e. lacks rigid pre-defined data structures (such as tables). Data is stored in a CouchDB database as JSON documents. The structure of these JSON documents can change to accommodate evolving needs.

6.1 Structure of the database

The edge node database actually consists of multiple databases, from which the edge node components and the search layer can ask for metadata:

- There is a single database that holds the initialization information for all the feeds that are registered with the edge node.
- Then, there is one database dedicated to each feed. These feed databases hold the metadata.

6.2 Setting up the system

The detailed manual for setting up a server to hold an edge node is to be provided in D4.3, “Integrated Edge Server.” Here just a brief overview is given.

6.2.1 CouchDB installation

Apache CouchDB offers the database storage and the direct access REST interface functionalities. The former have been described in section 6.1, while the latter is discussed throughout section 6.3.

To install CouchDB, download the executable from http://couchdb.apache.org/.

6.2.2 Apache and Tomcat servers

Apache HTTP server is used for reverse proxy, allowing the other servers (CouchDB & Tomcat) to share the common port 80 for easy access. Tomcat is the Java application server, hosting the edge

node J2EE application. An easy way to install both in Windows is to download XAMPP from Apache Friends: http://www.apachefriends.org/en/xampp.html.

6.2.3 Edge node J2EE application

This WAR file contains the servlets implementing the common interface as discussed throughout section 6.3. The installation simply involves copying the WAR file to the Tomcat web applications' directory.

The users wishing to setup an edge node can either use the WAR file, if they do not plan in changing the source code, or they can download and compile the complete project.

The edge node source code is an Eclipse project organized in three packages:

- **eu.smartfp7.EdgeNode**: This contains the servlets implementing the common interface.
- **eu.smartfp7.utils**: This contains utilities such as converters between XML and JSON, and simple parsers.
- **org.lightcouch**: This contains a modified version of LightCouch, a CouchDB Java API. The modifications were made to facilitate data access by avoiding conversions of JSON text to Java objects.

6.3 Interfacing to the system

Two interfaces have been built. The **direct access interface** provides direct access to the CouchDB and used the API of the database. It offers information in a more compact form (JSON). As it does not have intermediate processing stages, it is the fastest interface.

The **common interface** is not database-specific, effectively hiding the database from the user. It allows easy replacement of the database (should someone wish to implement an edge node with an SQL database). It is based on Java servlets, allowing different document types (either XML or JSON) and human-readable timecodes in the queries and data. The source code for this interface is contained in the software distribution of SMART, residing in the project for the edge node (SMARTEdgeNode directory). More specifically it is implemented in the eu.smartfp7.EdgeNode package with some utilities in the eu.smartfp7.utils package.

6.3.1 Feed management

Feed management has been covered in D3.1, "Sensors and Multimedia Data Knowledge Representation." This section documents the related Java servlets that implement it in the common interface.

Create data feed

This is the request to register a new data feed with the database. The XML description of the new feed is in the body of the HTTP request. The response can be any of the following:

- An HTTP response Status Code 500 (internal server error) is issued if no response can be sent or if the data cannot be read.
- Status Code 400 (bad request) indicates that the provided XML cannot be validated against the XSD. The validator error message is included in the body of the response.
- An HTTP response Status Code 201 (Created) indicates success, whereupon the feed is registered with the database.

A sample HTTP POST request for this could be:

http://dusk.ait.gr/SMARTEdgeNode/createFeed with **"Content-Type: application/xml"** in the Header and in the Body the XML description of the new feed, according to D3.1, "Sensors and Multimedia Data Knowledge Representation." Examples can be found in Sections 4.3 to 4.5 of that deliverable.

Retrieve feeds

This returns an XML description of registered feeds. If no tags are provided, then all feeds are returned. Using tags only the matching feeds are returned.

A sample HTTP GET request for this could be:

http://dusk.ait.gr/SMARTEdgeNode/listFeeds?tags=smartfp7:crowd

This example will return all feeds registered in the edge node database, which are tagged as analyzing crowd.

6.3.2 Populating with metadata

Direct access interface

Writing to the database can be achieved via the native calls to it. However this requires the data to be timestamped and in JSON format. Hence, it is not recommended to use this interface at all. Instead, the common interface should be utilized at all times.

The user wanting to ignore the above advice and use the CouchDB functionality for writing data should make sure that the data is in JSON format and includes timestamps and unique IDs. The data should then be written in the database corresponding to the feed.

For example he should issue an HTTP POST request to http://dusk.ait.gr/couchdb/aitathens with **"Content-Type: application/json"** in the Header and the following data in the Body:


```
{
	"_id" : "2012-07-26T10:14:10.795Z",
	"timestamp" : 1343297650795,
	"data" : {
		"time" : "2012-07-26T10:14:10.795Z",
		"activity" :
		{
			"name" : "#SMART_Demo",
			"isActive" : "true",
			"date" : "26-07-2012",
			"temporalHint" : "10:14"
		},
		"crowd" :
		[
			{
				"@ID" : "plaza_ayuntamiento",
				"density" : 0.170000,
				"motion_horizontal" : 0,
				"motion_vertical" : 0,
				"motion_spread" : 2.200000,
				"colour" : [32.125100, 2113632.106200,
2113600.092100, 4218976.066300, 4210752.061800, 2105376.042200,
8256.036100, 2105408.033300, 8413280.030500, 8224.024800, 0.023300,
10535136.022300, 6316160.021400, 4210784.017800, 2113664.017800,
12640480.017000]
			},
			{
				"@ID" : "calle_monasterio",
				"density" : "0.250000"
			}
		]
	}
}
```

Note that timestamp is the measurement time in milliseconds from 1 January 1970. Furthermore, the "_id" should either be unique, or omitted completely from the document.

If successful, it will return a status code “201 Created”

There exist various utilities to issue the above request; a very useful one is the “**RestClient**” plugin for Firefox and another commonly used is “**CURL**”, which can be invoked either from command line or as a C/C++ library.

Common interface

The Insert data to database servlet sends the request to write some data in the database. This common interface variant for writing data in the database is expected to be more widely used than its direct access counterpart for the following reasons:

- The Java servlet can validate the tags and datatypes, according to the description of the metadata at feed creation.
- Apart from JSON, XML formatting of the metadata being sent can also be supported.
- Automatic timestamping can be achieved. This is important since some sensors may not provide their own timestamps. In this case the time of data reception is used for timestamp.
- Furthermore, timestamps inside the data can be automatically converted from human-readable strings to machine-preferable integers and vice-versa.

A sample HTTP POST request for this could be:

http://dusk.ait.gr/SMARTEdgeNode/insertData?feed=aitathens with **“Content-Type: application/json”** in the Header and the following data in the Body:

```
{
	"time" : "2012-07-26T10:14:10.795Z",
	"activity" :
	{
		"name" : "#SMART_Demo",
		"isActive" : "true",
		"date" : "26-07-2012",
		"temporalHint" : "10:14"
	},
	"calle_monasterio" : {
		"density" : 0.170000,
		"motion_horizontal" : 0,
		"motion_vertical" : 0,
		"motion_spread" : 2.200000,
		"colour" : [32.125100, 2113632.106200, 2113600.092100]
	},
	"plaza_ayuntamiento" : {
		"density" : "0.250000"
	},
}
```

6.3.3 Querying for metadata

Direct access interface

The URL of all the web services of the direct access interface accessing CouchDB for the feed named *feedName*, residing in an edge node with URL *edgeNodeURL* is:

edgeNodeURL/couchdb/*feedName*/

The most important operation is streaming data from CouchDB:

http://dusk.ait.gr/couchdb/aitsmartlab/_changes?feed=continuous&include_docs=true&since=9980

This is the live mode: the returned information is updated by any new entry as it arrives in CouchDB. Note that currently the edge node *dusk.ait.gr* hosts two feeds, namely *aitathens* and *aitsmartlab* (see

the setup for the proof of concept at chapter 7). The query string (what follows the "?") encodes the following fields (streaming parameters):

- *since* field: This is the starting sequence number. It is useful for recovering from interrupted streaming connections, in which case we do not want all the data to be streamed again.
- *include_docs* field: This must always be set to *true* to actually stream the metadata instead of just a changelog.
- *feed* field: This also must always be set to *continuous*; otherwise only the metadata up to the current one will be streamed and the connection will be closed without waiting for new metadata.

During streaming, every new entry in the database for the requested feed is turned into a JSON document with increasing sequence number that looks as follows:


```
{
	"seq" : 8371,
	"id" : "c0fc77370f5348029e854e2b64ebe2fa",
	"changes" :
	[
		{
			"rev" : "1-5dfbb50a6a67601faf19295228be7259"
		}
	],
	"doc" :
	{
		"_id" : "c0fc77370f5348029e854e2b64ebe2fa",
		"_rev" : "1-5dfbb50a6a67601faf19295228be7259",
		"timestamp" : 1343297650795,
		"data" :
		{
			"time" : "2012-07-26T10:14:10.795Z",
			"activity" :
			{
				"name" : "#SMART_Demo",
				"isActive" : "true",
				"date" : "26-07-2012",
				"temporalHint" : "10:14"
			},
			"crowd" :
			[
				{
					"@ID" : "plaza_ayuntamiento",
					"density" : 0.170000,
					"motion_horizontal" : 0,
					"motion_vertical" : 0,
					"motion_spread" : 2.200000,
					"colour" : [32.125100, 2113632.106200,
2113600.092100, 4218976.066300, 4210752.061800, 2105376.042200,
8256.036100, 2105408.033300, 8413280.030500, 8224.024800, 0.023300,
10535136.022300, 6316160.021400, 4210784.017800, 2113664.017800,
12640480.017000]
				},
				{
					"@ID" : "calle_monasterio",
					"density" : "0.250000"
				}
			]
```

```
            }
        }
    }
```

The important part of this JSON document is the “doc” tag, in particular the “timestamp” and “data” tags.

The direct access interface also allows queries for parts of the data:

http://dusk.ait.gr/couchdb/aitathens/_design/get_data/_view/by_date?startkey=1343299431397&endkey=1343299434000&limit=10

This is the off-line mode. The command instructs to get metadata from some time (startkey), to some other time (endkey), both expressed as UNIX timecodes * 1000. It also allows limiting the number of returned entries (limit).

The URL for another example is:

http://dusk.ait.gr/couchdb/aitathens/_design/get_data/_view/by_date?limit=10&descending=true

This returns the ten latest entries.

Common interface

The common interface returns the RDF metadata in a XML collection. When querying for metadata, the main use of the common interface is for debugging purposes, since this interface does not offer a streaming mode.

The URL now accepts both human-readable or UNIX timecodes (in milliseconds, i.e. unix time * 1000) for start and end dates:

http://dusk.ait.gr/SMARTEdgeNode/retrieveXML?feed=aitathens&start_date=2012-07-26T10:43:22.318Z&end_date=2012-07-26T10:44:22.318Z

The entries are sorted oldest first.

A second example asks for the ten latest entries:

http://dusk.ait.gr/SMARTEdgeNode/retrieveXML?feed=aitathens&limit=10

In this case the entries are sorted most recent first.

7 Proof of concept: A day in AIT

The purpose of this proof-of-concept setup is to demonstrate all three layers of SMART functionality. Two different edge nodes are generating metadata, the search layer is indexing them and at the application layer mashups are provided and queries are answered. Instead of using two different physical servers, we simulated the two different edge nodes by using two different feeds in the same edge node.

7.1 Scenario: What is happening

At Thursday 26/7/2012, we have been simulating a conference at Athens Information technology, while at the same time work is as usual in the various labs.

Before that date, the conference has attracted some attention in the social media, and during that day there have been some real-time posts about it. The laboratory also has some minimal tweets associated to it, related to the meetings and demonstrations that have been expected to take place that day.

There have been two edge nodes involved in this scenario, one at the main amphitheatre of Athens Information Technology (@AITathens) where the conference is taking place, and another at the Smart Lab of Athens Information Technology (@AITSmartLab).

7.1.1 @AITathens

At the SMART Demonstration Day conference, identified in Twitter by the hash-tag #SMART_demo, there have been four presentations taking place during the afternoon session. They had been scheduled as in Table 7.1.

Table 7.1: Schedule of presentations at the SMART Demonstration Day conference.

Time	Speaker	Title
10:00	@jsoldatos	SMART architecture
10:45	@APnevmatikakis	SMART edge node
11:45	@aste80	Querying SMART
12:15	@menelaosbgr	SMART Applications

All these have been tweeted about in the previous days, and there have been some real-time tweets by the audience.

The @AITathens edge node has a single sensor and perceptual component attached to it: a camera and the crowd analysis algorithm that yields amongst other measurements the crowd density continuous metadata.

The actual times of the four presentations varies a bit, but can be derived by the activity in the conference amphitheatre. They are obtained by a simple thresholding of a filtered version of the crowd density continuous metadata feed, as shown in Table 7.2.

Table 7.2: Scheduled vs. actual time of presentations at the SMART Demonstration Day conference.

Scheduled time	Actual timespan
10:00	10:00-10:35
10:45	10:48-11:43
11:45	11:45-12:10
12:15	12:15-12:33

7.1.2 @AITSmartLab

Table 7.3 depicts the scheduled activities at @AITSmartLab. Some of them had been announced by tweets.

The second edge node, @AITSmartLab, monitoring the Smart Lab of Athens Information Technology, is equipped with a camera and a crowd analysis algorithm as the @AITathens node. Using the crowd density continuous metadata the actual timespan of the lab activities is estimated. There have been other unscheduled events (ad hoc group meetings) taking place as well.

Table 7.3: scheduled events at @AITSmartLab.

Scheduled time	Event	Actual timespan
7:10	MeetYourFuture visit group A	7:14-7:26
7:25	MeetYourFuture visit group B	7:28-7:51
8:00	Visit by @PetrosKokk	8:05-8:17
10:45	Conference call	10:52-11:41

7.2 Social streams and metadata

There has been some presence of the activities of the conference and the lab in Twitter during the days prior the proof-of-concept and on that day. The hash-tag of the activity (#SMART_demo) has been associated by the Social Network Manager to the place (edge node) it is happening (@AITathens). The tweets posted these days even go down to the presentation level of detail, announcing the time schedule:

- *The #SMART_demo of the @smartfp7 project will be hosted by @AITathens at 26/7/2012, 10:00GMT*
- *A presentation on @smartfp7 architecture is scheduled at the 10:00 session of #SMART_demo*
- *A presentation on @smartfp7 edge node is scheduled at the 10:45 session of #SMART_demo*
- *A presentation on @smartfp7 search is scheduled at the 11:45 session of #SMART_demo*
- *A presentation on @smartfp7 applications is scheduled at the 12:15 session of #SMART_demo*

Variations always include #SMART_demo. The time format did change. The edge node name @AITathens did or did not appear.

Live tweets about the presentations of #SMART_demo have been issued at the time of the event:

- *Now attending a presentation on @smartfp7 search at #SMART_demo*

For the second edge node, @AITSmartLab, the personnel of the lab have been tweeting about the scheduled visits:

- *The students of MeetYourFuture will be visiting @AITSmartLab in two groups at 26/7/2012, 7:00GMT*
- *Will be demonstrating @AITSmartLab to the MeetYourFuture students at 26/7/2012, 7:00GMT*
- *An official visit of @AITSmartLab is scheduled for 26/7/2012, 8:00GMT*

The filters in the Social Network Manager attempt to extract information about the activities associated with the edge nodes and format it in RDF triples of the form:

```
<activity>
    <name>#SMART_Demo<name>
    <isActive>false</isActive>
    <date>26-07-2012</date>
    <temporalHint>10:00</temporalHint>
    <temporalHint>10:45</temporalHint>
    <temporalHint>11:45</temporalHint>
    <temporalHint>12:15</temporalHint>
    <temporalHint>12:45</temporalHint>
</activity>
```

The triples in the example above are accumulated from all tweets about the #SMART_demo activity during the previous days. Since no RDFs have been collected in the previous days, this summary is included in the report from the @AITathens node issued at the timestamp 2012-07-26T09:59:51.147Z. Reports including <activity> tags from the real-time tweets are found at timestamps 2012-07-26T10:14:10.795Z, 2012-07-26T10:42:08.193Z, 2012-07-26T11:49:22.232Z, 2012-07-26T12:17:20.459Z and 2012-07-26T12:43:25.842Z.

A finer level of analysis at the Social Network Manager could result to understand finer details of the activity, but building such filters is beyond the scope of this deliverable. This will be covered in D4.2, Social Networks and Sensor Networks Integration.

7.3 Visual metadata: crowd analysis

The only perceptual component in operation at the time of the proof-of-concept has been the crowd analysis system that provides the crowd density metric as continuous visual metadata (i.e. a function of time, one sample per each processing epoch). Figure 7.1 depicts these continuous metadata, as they have been recorded by the @AITathens and the @AITSmartLab nodes.

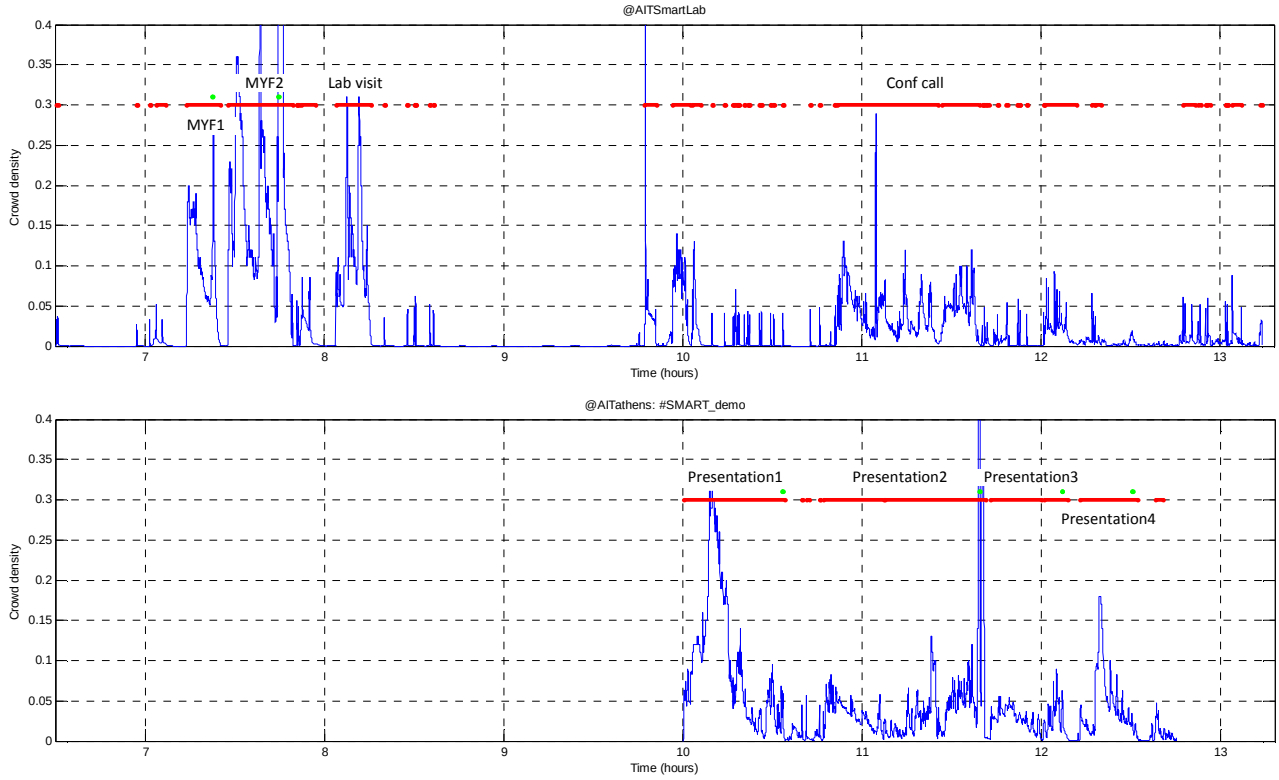

Figure 7.1: Crowd-related metadata comprise visual crowd density continuous metadata (blue waveforms) and low-level events (dense crowd as red lines and applause audio event as green dots) as a function of time for the two nodes involved in the proof-of-concept.

7.4 Low level events

The continuous metadata are mainly for within the edge node consumption, but at least for this proof-of-concept they have also been indexed by the search engine.

The low-level events are the signals used within the reasoning engine to provide higher-level reasoning. Each one is extracted by simple thresholding of a continuous metadata feed and is added in RDF

description to the same documents having the RDF descriptions of the continuous metadata.

7.4.1 Visual events

The visual low-level events are the crowd events resulting from thresholding the crowd density continuous metadata feed. The moment the threshold is exceeded, the crowd event is happening, while the moment the feed receives a smaller value, the crowd event is not happening.

The low-level crowd events are also shown in Figure 7.1: the periods the crowd is dense (i.e. in-between a crowd dense and a crowd not dense events) are depicted by red lines.

7.4.2 Audio events

To ensure richness of the metadata, we have also been simulating applause events, i.e. an audio low-level event. They are shown as green dots in Figure 7.1, representing the periods between two consecutive applause active and applause inactive events.

7.5 A preview on SMART applications

In this section we outline the usage of the SMART system at the application layer, by describing a web application that allows users to submit queries and retrieve results, which are then visualised on a mashup. This is inspired by a demonstration application that we have shown to the BBC World Service on 13th July 2012.

The interface of the web application is depicted in Figure 7.2. The interface follows a common pattern of traditional Web search engines. A text box allows the user to submit a query and the SMART application answers the query by retrieving relevant events to the query that happened recently and relevant posts from the social media.

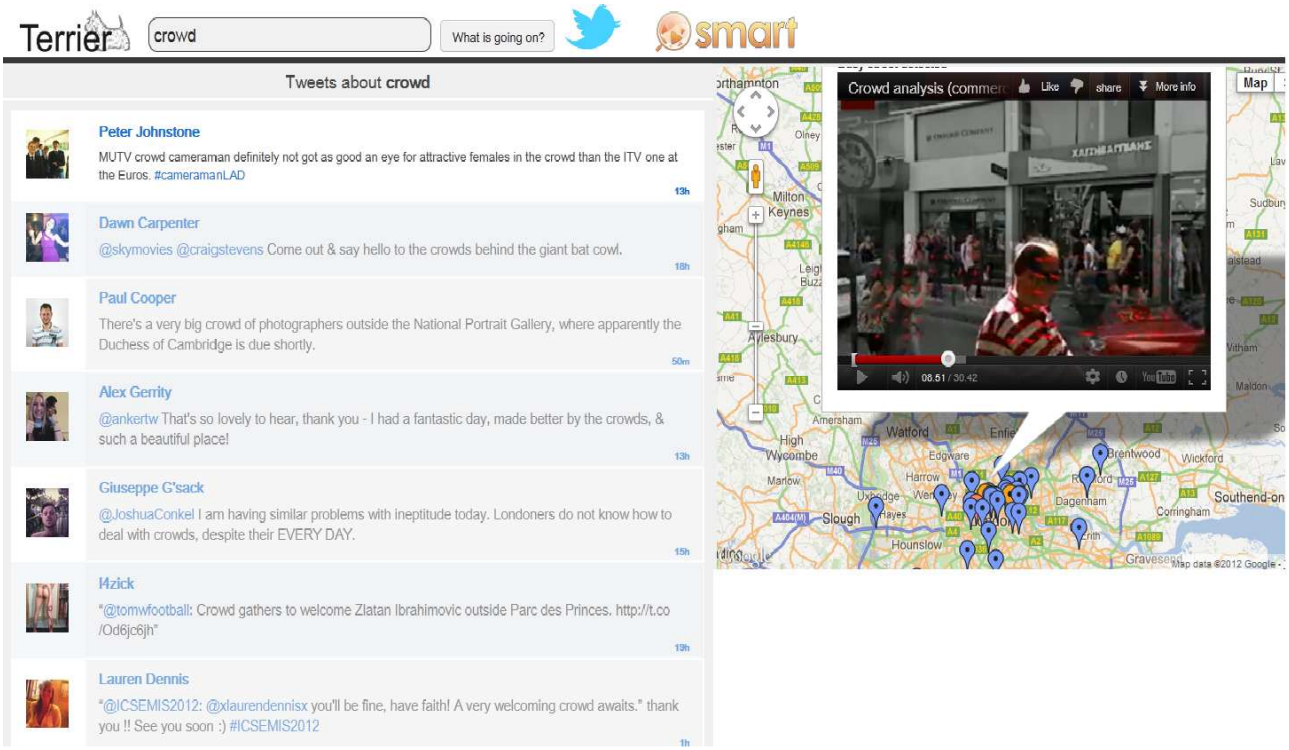

Figure 7.2: Screenshot of the Web interface.

Some example queries that are relevant to the proof-of-concept scenarios of the POC are:

- “SMART presentation”
- “conference”
- “School visit”

A mapping mashup widget is used to allow the user to interactively browse the results. The edge nodes from which the events were retrieved are displayed on the map (the orange markers). A balloon pops up upon clicking on these markers to play videos of the events and their time intervals. Relevant posts (tweets) from social media are also displayed as a ranked list and as blue icons on the map.

Moreover, as the event develops after submitting a query, the user is notified about new relevant events or social media posts. This is done by flashing new markers on the map and highlighting new social media posts (tweets) on top of the displayed list. For example, if the query ‘AIT conference’ was submitted during the second presentation, the user will be *notified* in *real-time* about the following presentations and the Tweets posted about those presentations.

8 Conclusions

In this deliverable we have described the way edge nodes manipulate metadata. Metadata generated from the processing stage software components are fed into the database and from there they are served to other edge node components, or to the search layer. The interface to the database is handled by the reporting stage software components.

The document serves two goals. On the one hand it presents the edge node functionality, which is demonstrated by the proof-of-concept. On the other hand it serves as a manual for the software implementation of the reporting components. These components are detailed in both the Apache CouchDB-driven direct access and the Java-based common interfaces are detailed.

The metadata are stored in the database as reports of linked data in RDF, using a JSON format. The predicates used in the RDF triples should reuse elements from established ontologies, but this is deferred to the second version of the document in January 2014.

Apart from the processed, filtered and reasoned metadata, the edge node will be able to gather Linked Data about its surroundings. This work is also deferred to the second version of the document in January 2014.

9 References

[1] Berners-Lee, Tim; James Hendler and Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved March 26, 2008.

[2] Tim Berners-Lee, Design Issues: Linked Data note (http://www.w3.org/DesignIssues/LinkedData).

[3] CouchDB: http://wiki.apache.org/couchdb/Introduction.