Enabling the business-based
Internet of Things and Services

(FP7 257852)

D4.5 Analysis and design of semantic interoperability mechanisms

Published by the ebbits Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2009.1.3: Internet of Things and Enterprise environments

Document control page

Document file: D4.5_Analysis_and_design_of_semantic_interoperability_mechanisms_v1_0.doc
Document version: 1.0
Document owner: Yves Martin (SAP AG)

Work package: WP4 – Semantic Knowledge Infrastructure
Task: Task T4.3 – Mechanisms for semantic interoperability in heterogeneous scenarios
Deliverable type: R

Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Yves Martin (SAP AG)	2012-01-01	Deliverable outline, ToC
0.4	Yves Martin (SAP AG)	2012-02-20	Chapters 1,2,3,4,5,6,9
0.5	Karol Furdik (InterSoft)	2012-02-20	Chapter 7
0.6	Peter Kostelnik (TUK)	2012-02-20	Chapter 8
0.7	Yves Martin (SAP AG)	2012-02-23	Version for internal review
0.8	Karol Furdik (InterSoft)	2012-02-28	Updated after internal reviews
0.9	Peter Kostelnik (TUK)	2012-02-28	Updated after internal reviews
1.0	Yves Martin (SAP AG)	2012-02-29	Updated after internal reviews and final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Peter Rosengren (CNET)	2012-02-28	Approved with comments
Jonathan Simon (FIT)	2012-02-27	Approved with minor revisions

Legal Notice

The information in this document is subject to change without notice.

The Members of the ebbits Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the ebbits Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Possible inaccuracies of information are under the responsibility of the project. This report reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein.

Index:

1. Executive summary

This deliverable reports on the analysis and design of semantic interoperability mechanisms. It belongs to the Task T4.3 "Mechanisms for semantic interoperability in heterogeneous scenarios" of the ebbits project. This task is concerned about the problem that the distributed ebbits information producers / consumers will need a common understanding of data they are processing in order to interact. Often heterogeneous data sources contain information about the same topic or the same item but they use different wording and terminology. By identifying entities (objects, data instances) referring to the same real-world entity, the relationships between information in heterogeneous data sources can be made explicit also. Therefore the task and also this report investigate the appropriateness of different strategies to provide such a common understanding in the ebbits project and also propose some approaches and designs for the semantic interoperability in the ebbits project. After the problem definition we evaluate different research approaches with regard to their potential use in the ebbits project. Furthermore, the special case of web services is assed and a design for using a common vocabulary and common instances throughout the project is proposed.

The report starts with an introduction to research in this area and gives a formal definition of the special instance of the semantic interoperability problem which has to be solved within the ebbits project for such a common understanding. It is called instance matching (also referred to various other names in different communities).

In the next section, an approach from the former EU project OKKAM is considered. It proposes the Entity Name System for providing each requester with an URI for the corresponding resource but it needs additional infrastructure.

The next section reports on matching approaches within the Linked Data field. Here, the Silk tool seems promising to achieve semantic interoperability of the ebbits middleware with data from (legacy) mainstream business systems.

An approach to interlink data from several sources, which can be trusted to various degrees, it presented in Section 6.

The next section addresses especially the semantic interoperability within Web services.

In the last section before the conclusion, a design for using a common vocabulary and common instances for all the components in the ebbits project is presented. With help of examples from the agricultural scenario it is also explained, how the ontology manager can be queried for the right term.

2. Introduction

2.1 Overview of the ebbits project

The ebbits project aims to develop architecture, technologies and processes, which allow businesses to semantically integrate the Internet of Things into mainstream enterprise systems and support interoperable real-world, online end-to-end business applications. It will provide semantic resolution to the Internet of Things and hence present a new bridge between backend enterprise applications, people, services and the physical world, using information generated by tags, sensors, and other devices and performing actions on the real world.

The ebbits platform will support interoperable business applications with context-aware processing of data separated in time and space, information and real-world events (addressing tags, sensors and actuators as services), people and workflows (operator and maintenance crews), optimisation using high-level business rules (energy and cost performance criteria), end-to-end business processes (traceability, lifecycle management), or comprehensive consumer demands (product authentication, trustworthy information, and knowledge sharing).

The ebbits platform will feature a Service-oriented Architecture (SoA) based on open protocols and middleware, effectively transforming every subsystem or device into a web service with semantic resolution. The ebbits platform thus enables the convergence of the Internet of People (IoP), the Internet of Things (IoT) and the Internet of Services (IoS) into the "Internet of People, Things and Services (IoPTS)" for business purposes.

The ebbits platform will be demonstrated in end-to-end business applications featuring connectivity to and online monitoring of a product during its entire lifecycle, i.e. from the early manufacturing stage to its end-of-life. The project will develop, implement and demonstrate two ebbits IoPTS applications. The first application demonstrates real-time optimisation metrics, including energy savings, in manufacturing processes. The other demonstrates online traceability with enhanced information on food.

2.2 Purpose, context and scope of this deliverable

This deliverable belongs to the Work Package 4 of the ebbits project, which aim is to research, develop and implement the infrastructure enabling a unified access to the semantic knowledge of the ebbits platform. The aim is to create a framework for the process of knowledge creation, access, maintenance and storage.

Furthermore, it contributes to the Task T4.3 "Mechanisms for semantic interoperability in heterogeneous scenarios" of the ebbits project. This task is concerned about the problem that the distributed ebbits information producers / consumers will need a common understanding of data they are processing in order to interact. This deliverable investigates the appropriateness of different strategies to provide such a common understanding in the ebbits project and also propose some approaches and designs for the semantic interoperability in the ebbits project.

This deliverable is based on results of the previously published deliverables D4.2 "Knowledge representation formalism analysis" and D6.2 "State-of-the-Art design and implementation of systems with persistent and high volume data" and its results will be used in the upcoming deliverables D4.8.1, D4.8.2, D4.8.3 "Implementation of semantic interoperability mechanisms with the ebbits platform" and D6.4 "Implementation of service description, matchmaking and time scale moderation mechanisms".

2.3 Background

This deliverable contributes to the Task T4.3 and uses outcomes of deliverables D4.2 and D6.2.

3. Research Background and Problem Definition

The description of work of Task T4.3 expects that strategies should be examined which could lead to a common understanding of all the data within the ebbits project by making the relationship between information in heterogeneous data sources explicit. For this objective, a semantic representation of the data stored in multiple sources is essential (Jean-Mary, Shironoshita, & Kabuka, 2009). This representation will facilitate the definition of a correspondence among entities from different sources and the resolution of conflicts among sources and should ultimately automate the integration process.

The main problem in this information integration process is the task of identifying entities (objects, data instances) from different sources refererring to the same real-world entity. If this objective is achieved then the information from these multiple sources can be safely combined and this allows ebbits applications to process queries and do reasoning in a homogeneous way over heterogeneous data sources as reflected in the domain scenarios. This task is not easy and it is time consuming and thus imposes a crucial limit for large-scale, dynamic information integration (Bouquet, Stoermer, & Niederee, 2008).

This problem has already been a topic of research for more than 50 years (Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010). During this time it has been considered under various facets and from different communities, including the Artificial Intelligence research community, the Database research community, industry and lately, the Semantic Web research community. Therefore, it has many names and is among others referred to as object matching, entity matching, duplicate indentification, record linkage, entity resolution, reference reconciliation, data deduplication, instance identification, name matching, data cleaning, ontology matching, instance matching (Köpcke & Rahm, 2009; Veshcheva, 2011; Elmagarmid, Ipeirotis, & Verykios, 2007, Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010).

The majority of the existing approaches to this problem tackle the task of matching database records while recent approaches focus mostly on graph-based data representations extended by additional schema information (Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010). The latter methods refer to the problem as ontology matching / terminological alignment and instance matching. As in the ebbits project, we also rely on semantic technologies like ontologies and graph-based data representations (possibly extended with further schema information) (see also Deliverables D4.2 "Knowledge representation formalism analysis"and D6.2 "State-of-the-Art design and implementation of systems with persistent and high volume data" (Hreno, Nutakki, Knechtel, Furdik, & Sabol, 2011; Martin & Ahlsén, 2011)), in this deliverable we will focus on the modern Ontology / Semantic Web based approaches (for an overview about the database approaches see (Elmagarmid, Ipeirotis, & Verykios, 2007)).

In terms of the Semantic Web community, we can give a general formal definition for the task of making the relationship between entities from different sources explicit:

Definition 1 (Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010)**:** Given ontologies O_1 and O_2, let q be a function that defines sets of matchable entities $q(O_1)$ and $q(O_2)$. A correspondence between O_1 and O_2 is a four tuple $<e_1, e_2, r, n>$ such that $e_1 \in (O_1)$ and $e_2 \in (O_2)$, r is a semantic relation and n is a confidence value. An alignment M between O_1 and O_2 is a set of correspondences between O_1 and O_2.

This definition covers a wide range of correspondences by specifying what are matchable entities, a semantic relation, and a confidence value. One important difference between diverse matching tasks is determined by the relation q on the set of matchable entities (Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010). We can distinguish two cases here:

- **Links between terminological entities:** This task arises due to the heterogeneity of vocabulary between different ontologies: the same concept (e.g. "author") or property (e.g. "last name") may be referenced to through different URI's in linked data or different names in the T-Boxes of the ontologies, and therefore may not be recognized as the same concept

or property in two different vocabularies. The alignment *M* in above definition would be providing equivalence or subsumption links between concepts and properties for this case. The most common strategies to reach such an alignment are structure-based (e.g. looking at the hierarchy between concepts or restrictions between concepts and properties) or methods to compute and aggregate value similarities. In the ebbits project, the ontologies are constructed by the Work Package 4 and a common ontology will be used as described in Section 8. Therefore, in this deliverable we do not focus on the terminological matching as this should always be a small or non-existing problem within the ebbits project and thus, solvable by some manual effort (for details on example ontology matching systems RiMOM and ASMOV please confer (Li, Tang, Li, & Luo, 2009; Jean-Mary, Shironoshita, & Kabuka, 2009)).

- **Links between instances:** This task requires identification of instances, i.e., the same real world object (e.g. "Dresden") may be assigned different URI's in different RDF repositories or different names in the A-Boxes of ontologies, and therefore may not be recognized as the same instance. The assignment of different URIs to the same resource is thereby called an URI alias. The alignment *M* in above definition would provide pairs of instances that refer to the same real world object and thus, the semantic relation is that of identity for this case. Such an identity relation can be expressed between RDF repositories as an http://www.w3.org/2002/07/owl#sameAs (owl:sameAs) link between the URI aliases (Veshcheva, 2011). It is usually derived by inferring similarities between instance labels and datatype properties. In the ebbits project, this task should be again avoided as much as possible and the identity of instances should be common over all compomenents as described in Section 8. This approach is similar to the Entity Name System which is delineated in Section 4. However, the ebbits middleware with all its components should also be connected to mainstream (legacy) Enterprise systems. Although these systems usually do not contain a lot of terminological knowledge in form of ontologies, instances referring to the same object in these systems and and the ebbits middleware have to be matched. As already specified in deliverable D6.2 "State-of-the-Art design and implementation of systems with persistent and high volume data", the data in these systems should be converted to linked data and therefore the RDF format or alternatively, the query language is mapped (for details see (Martin & Ahlsén, 2011)). As soon as the information is in this graph-based data representation format, special matching algorithms for linked data can be used and these are described in Section 5. The linked data approaches usually work incrementally which eases the handling of large data volumes.

In the above definition, there is also a confidence value *n* for a correspondence given. This value quantifies the degree of trust in the correctness of the proposed correspondence (Noessner, Niepert, Meilicke, & Stuckenschmidt, 2010). If the correspondence is generated automatically by a matching system, this value is usually an average of scores from different matching strategies. In general, due to the large variety of data sources and entities to match there is no single "best" matching strategy. A single strategy which performs very well on one domain can be very bad for a different match problem in another domain. Therefore, it is often useful and necessary to combine several methods for improved matching quality (Köpcke & Rahm, 2009). Furthermore, if one wants to compare entities from a set *A* with entities from a set *B*, then in general */A/*/B/* comparisons are necessary. In this case, each object from set *A* is compared with each object from set *B*. This approach leads for already medium-sized sets to very high computing costs. For larger datasets, it is therefore necessary to apply optimization to avoid unnessary comparisons and to reduce the search space for matching and achieve sufficiently fast execution times. Such optimizations are so-called blocking strategies and they partition the input sets and then, pairs are only compared within a partition. Doing so may lead to poorer results of the matching as correct pairs can now lie in different partitions.

In (Köpcke & Rahm, 2009), several high-level requirements which can be used to compare different matching approaches, are defined:

- **Effectiveness:** The main goal for matching is to achieve a high-quality match result. This criterion is usually measured in recall and precision, i.e., all real corresponding pairs but no other should be included in the result.
- **Efficiency:** The matching approach should be fast even for large datasets.
- **Genericity:** The algorithm should be applicable to different match problems from various domains and support different data models.
- **Online/Offline Matching:** A matching system should be applicable to online and offline match tasks, i.e., on-demand matching for interactive data integration steps such as mediated quieries or data mashups based on specific user input or pro-active homogenization. The offline matching can be further categorized into fully matched beforehand or so-called "pay as you go" approaches. In general, oflline matching is less time critical and therefore allows the usage of more matching strategies and the application to larger datasets. Most of the current approaches do offline matching.
- **Low manual effort:** The manual effort to determine parameters and the combination of matching strategies should be as low as possible. To facilitate this, some of the approaches use machine learning techniques with training data in order to learn automatically how to self-tune the system. With these techniques, the additional question of selection and provenance of correct training data arises which could also lead to further manual effort.

In the ebbits project, the considered approaches should be effective and efficient. The genericity can be restricted for the moment to our application domains, the food tracking and the manufacturing scenario. Furthermore, at the moment, the use cases do not foresee complicated and unanticipated ad-hoc end-user queries such that we can concentrate on the prevailant offline approaches. The manual effort should be low but as we already have a lot of domain knowledge about the ebbits use cases, this knowledge should be integrated to improve the quality of the results.

4. The Entity Name System - the OKKAM Approach

The FP7 EU Large-scale Integrating Project **OKKAM – Enabling a Web of Entities** (contract no. ICT-215032), which ran from 2008 until 2010 and is presented here because it was exactly about instance matching, already recognized that identifying information from different sources which refer to the same (real world) entity is a crucial challenge in instance-level information integration (Bouquet, Stoermer, Niederee, & Mana, 2008). The integration of various information islands into a global Semantic Web should be based on the use of URIs. If two RDF graphs have two nodes labeled with the same URI and additionally there are no URI aliases and no URI collisions (assignment of the same URI to two different resources) in and between these two graphs, the common nodes can be collapsed forming a bigger graph. This way, the information about the resource labeled with the URI and its related nodes is automatically integrated from the two sources. Therefore, both, URI collisions and URI aliases, should be avoided and the latter one is more difficult to achieve.

One possible solution, which aims at abolishing URI aliases and this especially on the instance level, is the so-called Entity Name System (ENS) which was proposed as part of the OKKAM project. The ENS should be a web-scale infrastructure for supporting reliable and trustworthy the reuse of URIs. It should work for any type of entity and across decentralized and independent RDF repositories (Bouquet, Stoermer, Niederee, & Mana, 2008). ENS should be an essential part of the Semantic Web as it would allow information integration by simple RDF graph merging.

The main functionality of the ENS can be generally described as: The input to the system is given by any representation of an entity (e.g. a set of keywords, a paragraph of text, a set of key-value pairs, a graph …). With this input and matching techniques the systems decides if an URI for this entity is already available in the entity repository. If this is the case, the system will return the URI or at least a ranked list of top candidates. Otherwise, the ENS issues and stores a new URI. In both cases, different users are given the same URI for this resource (Bouquet, Stoermer, Niederee, & Mana, 2008).

There are a lot of technical and organizational issues to be solved for such an approach (Bouquet, Stoermer, Niederee, & Mana, 2008):

- **Dealing with entity types:** The question arises how far and in what granularity the system should support types and schemas for the entities. On the one hand, a strong type system can increase effectiveness and efficiency of the internally used matching strategies. On the other hand, for a global approach it is difficult to agree on common types, only for very general top-level types can a consensus be reached across organizations and communities.
- **Repository maintenance:** The technical questions are here about aging information and the need to update information about entities in the repository to adjust to changed entity representations in the real world.
- **Trust, privacy and ownership:** The questions of access control for a distributed system and very different users with various crucial security and privacy requirements have to be addressed.
- **Scalability:** If the system is used by a large amount of users, a lot of matching tasks have to be solved in parallel as ENS should be applied in automated search and reasoning problems.

The ENS is implemented as a federated architecture and depicted in Figure 1. In addition to the distributed system, the architecture also proposes a so-called local node. This node is foreseen for organizations which want to store their entities on their own infrastructure for reasons of security for highly sensitive information or because a large amount of these entities are only of interest inside the corresponding institution (Bouquet, Stoermer, Niederee, & Mana, 2008). The identifiers used inside the architecture are guaranteed to be unique across space and time and thus avoiding URI collision.

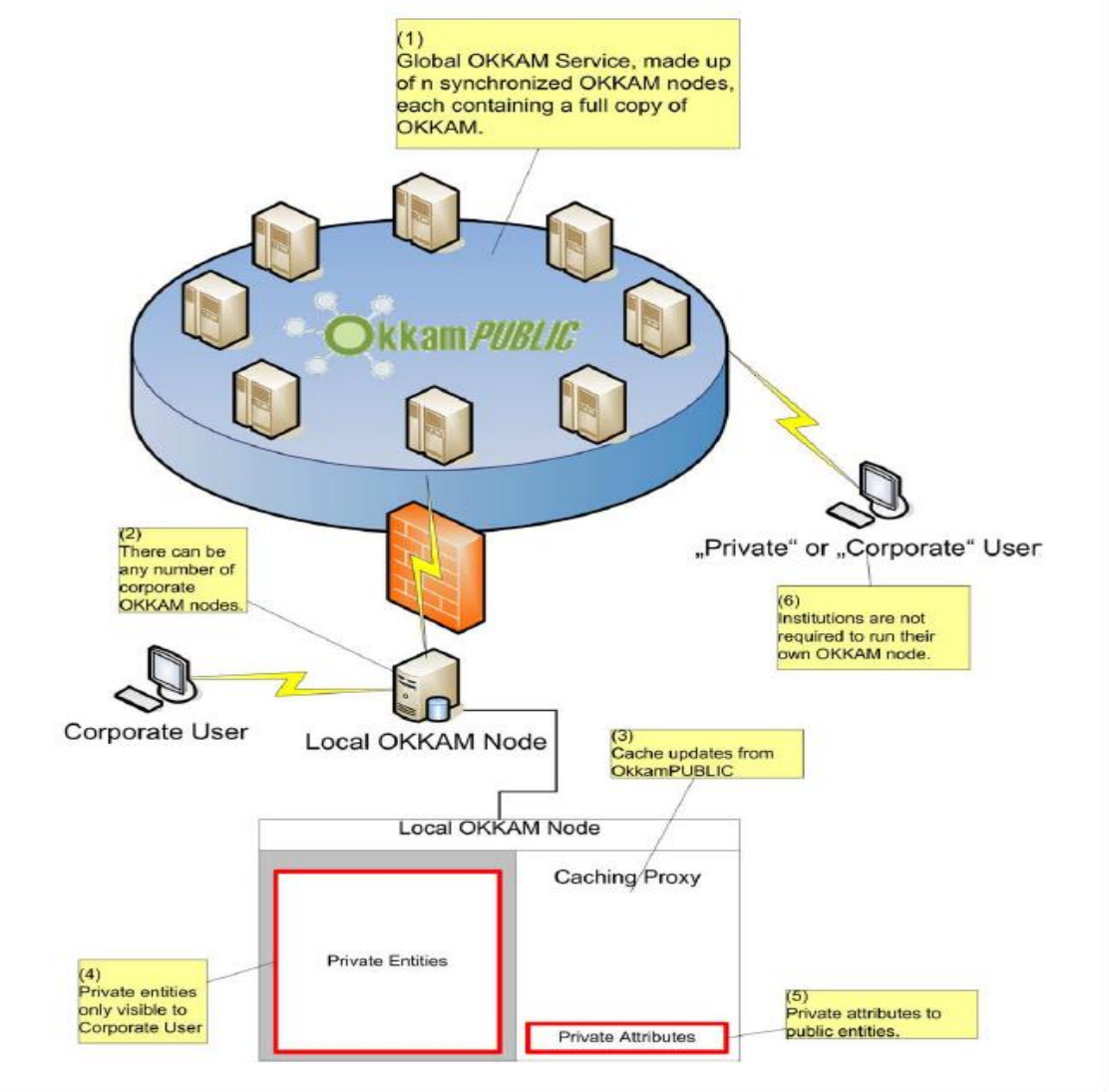

Figure 1: The Architecture of OKKAM ENS, Source: (Bouquet, Stoermer, Niederee, & Mana, 2008)

In the ebbits project, we retain the idea from this approach that we should avoid URI aliases as much as possible but we do not setup such an entity name system as an additional infrastructure. Even so we do not aim for a global service; we also want to avoid setting up a complete local OKKAM node as this would need additional resources and effort. Instead we use a common vocabulary as defined in Section 8 and do pro-active homogenization as specified in the next section.

5. Matching within Linked Data Approaches

As already pointed out in Section 3, information about instances in legacy systems should be converted into the linked data RDF format (see also (Martin & Ahlsén, 2011)). Once the data is in this graph-based representation, it can be combined with the ontologies and instances already in the ebbits project, which are also in the RDF format and also use URI references. Of course, the information from the two sources has to be matched in order to identify data instances referring to the same real-world entity. In this section, special characteristics of matching in the context of linked data and two suitable approaches from the Semantic Web community are presented.

One of the principles of linked data is that links between different data sources should be established. This principle does not work so well in practice: There are already more than 25 billion RDF triple of data published in different sources but only less than 3% of them are connected via links (Veshcheva, 2011). If such links exist in linked data, there between two URIs and denote the identity via owl:sameAs. Why does this problem of URI aliases arise at all? There are different reasons:

- **Differences in opinions:** The semantic web is open and it therefore is very likely that different providers publish data about the same real world object. Thereby, there exist different opinions and points of view.
- **Traceability:** If different URIs are used from various publishers, then the provenance of the data can be tracked.
- **No single point of failure:** The web is inherently distributed and that makes one institution responsible for all URIs very unlikely.

Thus, for the Linked Data community, there will be always URI aliases, which are created and later have to be linked together by owl:sameAs. In doing so, the creation of links is again distributed, i.e., different users can provide links for the same two data sources. Additionally, the publisher of the data can already provide links of his/her data to other sources. Thus, the effort of linkage of data sources can be shared. This aspect of linked data can also be a disadvantage because the quality of the links can be low. Therefore, the provenance of the links should always be taken into account. Furthermore, it is an evolutionary, "pay as you go" approach, where additionally links can always be added and one does not have to start with a complete set (Veshcheva, 2011).

For small and static data sets, the interlinking can be done manually while larger data sets generally require an automated or semi-automated approach (Health & Bizer, 2011). If the data sets contain accepted naming schemata, for example the Global Trade Item Numbers (GTIN), the EPC product code or the International Standard Book Number (ISBN), such identifiers should be exposed either as part of the URIs or as property values. Such a value of the property uniquely identifies the subject of the RDF triple and thus, enables simple key-based matching approaches where just the key pattern have to be compared. Otherwise, if such common identifiers do not exist across data sets or if the key values are missing, linked data approaches use the same similarity-based strategies as in database or ontology matching: They aggregate the different similarity scores from lexical and value matchers as well as property-based comparisons of related entries. If all (or most) of the comparisons result in high similarity values and the corresponding aggregations are above a given threshold, the compared instances are interlinked by an owl:sameAs statement. A difference to other research fields is that in linked data scenarios entities in one data source often come from different vocabularies and no consistent RDF or OWL schemata spanning this data source exists. In the following, we present two tools that allow matching heuristics to be defined declaratively and to automate the process of generating RDF links based on these definitions.

5.1 The Silk Linking Framework

The general features and the architecture of the Silk framework has already be described in Deliverable 6.2 "State-of-the-Art design and implementation of systems with persistent and high volume data" and the reader is referred to (Martin & Ahlsén, 2011) for details.

The Silk – Link Specification Language (Silk-LSL) allows in a declarative manner to specify the heuristics and the combination of them for deciding whether a semantic relationship exists between two instances (Volz, Bizer, Gaedke, & Kobilarov, 2009). Furthermore, data access parameters and configurations for caching, indexing and preselection features are defined in Silk-LSL. The following is a first abbreviated example proposal for a Silk-LSL specification for the ebbits project, which would be employed to match device and event instances:

```
01 <Silk>
02 <DataSource id="event">
03 <EndpointURI>http://server1.ebbits.eu/Event/sparql</EndpointURI>
05 <DoCache>1</DoCache>
06 <PageSize>1000</PageSize>
07 </DataSource>
08 <DataSource id="device">
09 <EndpointURI>http://server2.ebbits.eu/Device/sparql</EndpointURI>
10 </DataSource>                    ……
22 <LinkType>owl:sameAs</LinkType>
23 <SourceDataset dataSource="event" var="a">
24 <RestrictTo>
25 ?a rdf:type event:WaterflowEvent
26 </RestrictTo>
27 </SourceDataset>                  ……
51 <AVG>
52 <MAX optional="1" weight="5">
62 <Compare metric="stringEquality" optional="1">
63 <Param name="str1" path="?a/event:name" />
64 <Param name="str2" path="?b/device:name" />
65 </Compare>
66 <Compare metric="stringEquality" optional="1">
67 <Param name="str1" path="?a/event:channel" />
68 <Param name="str2" path="?b/device:channel" />
69 </Compare>
70 </MAX>
71 <Compare metric="numSimilarity" optional="1" weight="2">
72 <Param name="num1" path="?a/event:idNumber" />
73 <Param name="num2" path="?b/device:idNumber" />
74 </Compare>
75 </AVG>
77 <Thresholds accept="0.9" verify="0.7" />
78 <Limit max="1" method="metric_value" />
79 <Output acceptedLinks="device_accepted_links.n3"
80 verifyLinks="device_verify_links.n3" format="n3" mode="truncate" />
81 </Interlink>
82 </Silk>
```

In the first 10 lines the endpoints of the data sources are given. Line 22 specifies the link type and the lines afterwards restrict the search for the events to a specific type. From line 51 on the different matching strategies are defined, in our example, two string matchers and one numerical value matcher. Additionally, it is stipulated that maximum value resulting from the string matchers should be taken and if at least one of them defines a match, this result should be weighted by the factor 5. The resulting value is then averaged with the value from the numerical matcher. The last lines define threshold for automated linking and for showing it to the user. Furthermore, the maximum number of outgoing links is also declared. In the end, the output format and the output file is specified. Of course, this initial specification would have to be adapted to the linked data use cases in the ebbits project, tested, iteratively improved and extended. The use case partners in ebbits would also be able to informally give some more domain knowledge which then can be easily expressed in this declarative specification language.

In practical matching problems, data cleanliness and completeness problems are not always visible from the start on. For example, two data sources, that support a common naming schema, seem to look like an easy key-based matching, but if a large number of such key values are missing, additional similarity-based techniques are necessary. Such data quality problems may not be noticed until the actual computation of links (Volz, Bizer, Gaedke, & Kobilarov, 2009). Therefore, the process is usually iterative and the Silk framework supports the users with fine-tuning their linking specification by providing a web interface for evaluating the correctness and completeness of generated links. It is depicted in Figure 2, where the users has entered a set of computed links for inspection and the framework then recomputes these links and displays the resulting similarity scores for each link.

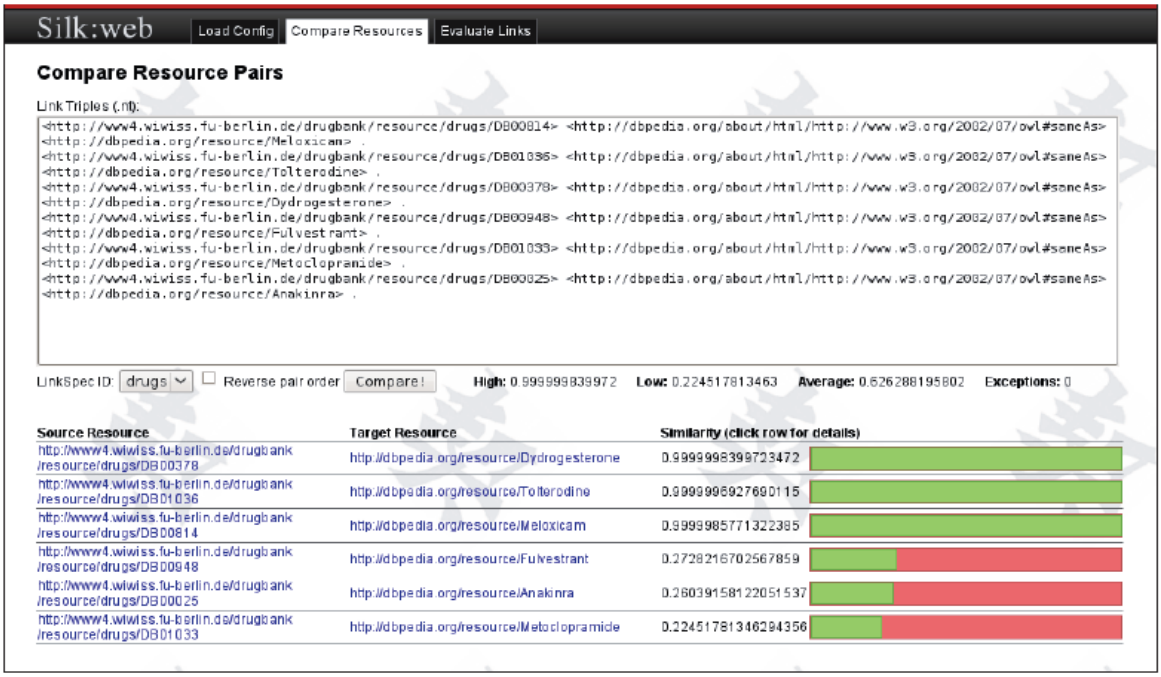

Figure 2: Comparing resource pairs with the Silk web interface, Source: (Volz, Bizer, Gaedke, & Kobilarov, 2009)

It is also possible to do a drill-down by clicking on one of the links in the overview table. Another view is displayed which is depicted in Figure 3. One can see in this figure in detail how the different matching values for set, string and numbers are computed and aggregated for this drug comparison example. This information allows the users to spot bad behaving parts of their linking declarations.

As the world is dynamic, datasets change and are extended over time. With the changing sources, also the links between them should be updated by introducing new necessary RDF links and removing now superfluous links. For this objective, in (Volz, Bizer, Gaedke, & Kobilarov, 2009), the Web of Data – Link Maintenance Protocol (WOD-LMP) is proposed which automates the communication between two data sources, the link source and the link target. The protocol supports:

- **Initial linking:** A notification can be send from the link source to the target source that the link source has initially published links pointing to the target source. The target source may then consider setting back-links.
- **Periodic update:** The link source requests a list of changes from the target source and uses this information to recompute existing and new links.
- **Continuous monitoring:** The link source subscribes to changes of specific resources in the target source by sending a link notification message.

The Silk framework seems suitable to be used in the ebbits project. Thanks to its declarative nature and many contained matching strategies it should be possible to easily encode the domain knowledge about our application scenarios in this matching framework. The Web interface will be helpful to do this in an iterative fashion. Furthermore, it may be worthwhile to use WOD-LMP for continuous updates.

```
AVG(
  MAX(
    maxSimilarityInSets(
      submetric = "jaroSets",
      set1 = path("?a/rdfs:label") = ['Lorazepam', 'Myrciaria', 'Lorazepam', 'Loratsepaami', 'Lorazépam',
      set2 = path("?b/rdfs:label") = ['Lorazepam']
    ) = 1.0 [W:1.0 O:False D:None],
    maxSimilarityInSets(
      submetric = "jaroSets",
      set1 = path("?a/rdfs:label") = ['Lorazepam', 'Myrciaria', 'Lorazepam', 'Loratsepaami', 'Lorazépam',
      set2 = path("?b/drugbank:synonym") = ['(+/-)-Lorazepam', 'L-Lorazepam Acetate', '0-Chlorooxazepam',
    ) = 0.866666666667 [W:1.0 O:True D:None],
    maxSimilarityInSets(
      submetric = "jaroSets",
      set1 = path("?a/rdfs:label") = ['Lorazepam', 'Myrciaria', 'Lorazepam', 'Loratsepaami', 'Lorazépam',
      set2 = path("?b/drugbank:genericName") = ['Lorazepam']
    ) = 1.0 [W:1.0 O:True D:None]
  ) = 1.0 [W:1.0 O:False D:None],
  MAX(
    stringEquality(
      str2 = path("?b/drugbank:atcCode") = ['N05BA06'],
      concat(
        str2 = path("?a/dbpedia-owl:atcsuffix") = ['BA06'],
        str1 = path("?a/dbpedia-owl:atcprefix") = ['N05']
      ) = ['N05BA06'] [W:- O:- D:-]
    ) = 1 [W:1.0 O:True D:None],
    stringEquality(
      str2 = path("?b/drugbank:casRegistryNumber") = ['http://bio2rdf.org/cas:846-49-1'],
      str1 = path("?a/dbpedia-owl:casNumberLink") = ['http://bio2rdf.org/cas:0846-49-01']
    ) = 0 [W:1.0 O:True D:None],
    stringEquality(
      str2 = path("?b/drugbank:pubchemCompoundId") = ['3958'],
      str1 = path("?a/dbpedia-owl:pubchem") = ['3958']
    ) = 1 [W:1.0 O:True D:None]
  ) = 1.0 [W:5.0 O:True D:None],
  numSimilarity(
    num1 = path("?a/dbpedia-owl:molecularWeight") = ['321.2'],
    num2 = path("?b/drugbank:molecularWeightAverage") = ['321.158']
  ) = 0.999869240349 [W:2.0 O:True D:None]
) = 0.999967310087 [W:- O:- D:-]
```

Figure 3: Detailed drill-down into a comparison of a pair of drug instances, Source: (Volz, Bizer, Gaedke, & Kobilarov, 2009)

5.2 The LIMES Framework

This framework was especially developed as a time-efficient approach for the large-scale matching of instances in metric spaces. It uses the triangle inequality in metric spaces to filter out a large number of those instance pairs which approximated similarity value can't be above the given threshold (Ngonda Ngomo & Sören, 2011). Therefore, it realizes a blocking strategy as already described in Section 3. After the filtering process the real similarity values of the remaining instance pairs are then computed and the matching instances returned. In contrast to other blocking strategies, the LIMES approach does not sacrifice precision and therefore, it does not loose correct pairs.

The idea is the following (Ngonda Ngomo & Sören, 2011): Given three points x, y and z and a metric m, the triangle inequality entails that $m(x, y) \leq m(x, z) + m(z, y)$. After some transformation we get the following boundary condition in metric spaces:

$$m(x, y) - m(y, z) \leq m(x, z) \leq m(x, y) + m(y, z)$$

From this condition it follows that the distance between a point x to any point z can be approximated if one knows the distance from x to a reference point y and from y to z. Such a reference point can represent a portion of the metric space and is then called an exemplar. This partitioning is graphically depicted once more in Figure 4. From the condition it follows furthermore, that the real distance from x to z can only be smaller than a given similarity threshold θ, if the lower bound is smaller than θ. The contrapositive of this observation is the following:

$$m(x, y) - m(y, z) > \theta \Rightarrow m(x, z) > \theta$$

If the distances from all instances t of the target source to exemplars are known, one can compute an approximation of the distance from all instances s from the source to all t and filter out all those pairs for which the approximation is already greater than the threshold. Afterwards, only for the remaining pairs the real distance has to be calculated.

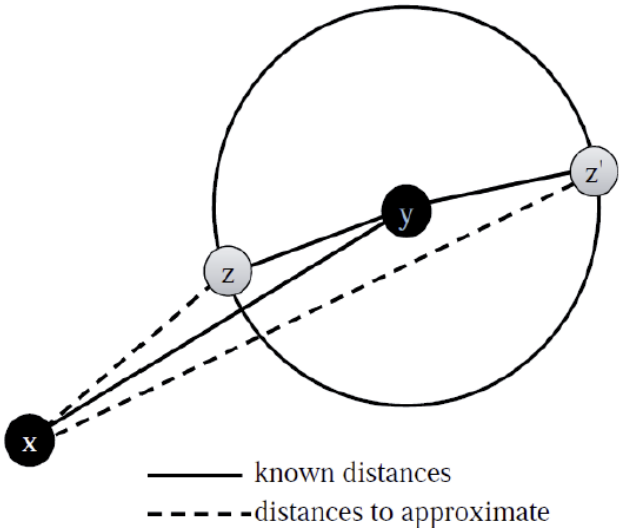

Figure 4: Approximation in the LIMES framework, Source: (Veshcheva, 2011)

In general, the workflow in the LIMES framework, which is also illustrated in Figure 5, consists of four steps (Ngonda Ngomo & Sören, 2011), where the source data set S, the target data set T and the threshold θ are given as input:

1. A set E of exemplars is determined. Exemplars are instances in the metric space and they should be distributed in a uniform way in that space. To achieve this goal, they are chosen to be very dissimilar. Furthermore, the distances of the exemplars to all $t \in T$ are calculated and each $t \in T$ is assigned to its nearest exemplar. The number of exemplars can be set to $\sqrt{T}$.
2. For each $s \in S$ and each $e \in E$, the distance $m(s,e)$ is figured out, which is used in turn to approximate the distance from s to every $t \in T$. The filtering takes place here and can reduce a large number of comparisons compared with the brute force approach.
3. For the remaining pairs (s,t) the real distance $m(s,t)$ is computed.
4. Finally, for the pairs with a high enough similarity the matching $(s,t,m(s,t))$ is stored in a user-specified format.

The LIMES framework could be used ebbits project as a preprocessing step to reduce the number of full comparisons needed. Of course, it is restricted at the moment to metric spaces, thus, for example in the manufacturing scenario the position of the items/machines should be used as metric.

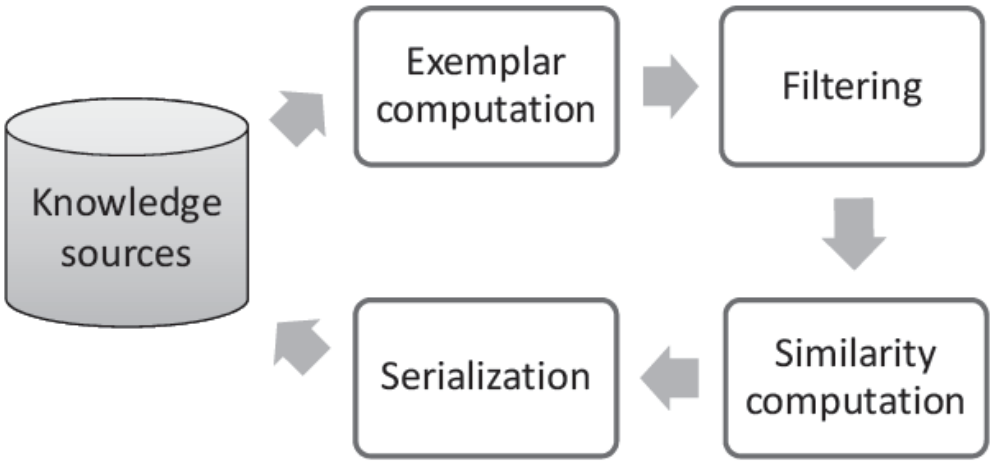

Figure 5: General workflow in the LIMES framework, Source: (Ngonda Ngomo & Sören, 2011)

6. Matching Data from several, differently trusted Sources

From the approaches in Section 5 it is expected that they solve many local, pairwise data integration problems involving two different sources. The quality of such a local solution will differ because of diverse employed matching strategies and also the amount of effort that was undertaken. Thus, we can have a different trust level in each of these pairwise data integration results. As already described in Section 5, everybody can publish links to every topic and therefore, it is probable that we also have conflicting links from several sources with different trust levels. For such a situation, which calls for a large-scale, distributed and uncertain data management, the framework idMesh is proposed. It uses an analysis based on graphs of instances and their uncertain links. Furthermore, it also addresses temporal discrimination which distinguishes entities pertaining to the same referent but taken at different points in time. The approach will only be sketched here with an example, which is depicted in Figure 6; the reader is referred to (Cudre-Mauroux, Haghani, Jost, Aberer, & de Meer, 2009) for details.

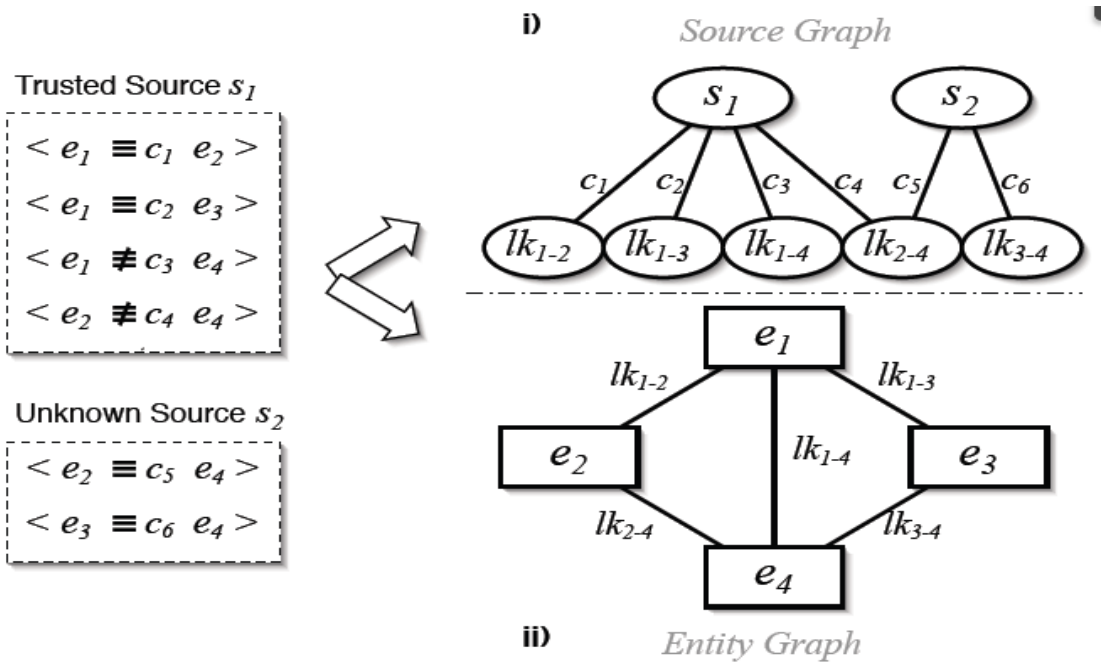

Figure 6: Two sources relate four entities, Source: (Cudre-Mauroux, Haghani, Jost, Aberer, & de Meer, 2009)

Figure 6 illustrates a simple example, where one trusted source s_1 and one unknown source s_2 relate four entities e_1 to e_4 by defining equivalence links lk between them with various confidence values c. Two graphs are created from the information on the left side of the figure: a bipartite source graph relating sources to the links they define and an entity graph specifying how entities are related through the links. As s_2 is the unknown source, its proposed links should be considered with care. By observing the source graph and furthermore, taking into account that an equalivalence relation is symmetric and transitive, conflicts for various links are detected: The link between e_2 and e_4 is defined as non-equivalent by s_1 and equivalent by s_2. Furthermore, from $e_1 \equiv e_3 \wedge e_3 \equiv e_4$ should follow $e_1 \equiv e_4$, which is in conflict with the trusted source s_1. In the end, the algorithm in (Cudre-Mauroux, Haghani, Jost, Aberer, & de Meer, 2009) lowers the confidence of the links declared by s_2 and concludes that e_1, e_2, and e_3 are equivalent and different from e_4.

From the source graph and the entity graph, the approach in (Cudre-Mauroux, Haghani, Jost, Aberer, & de Meer, 2009) finally constructs a so-called constraint satisfaction-based factor-graph, which is solved by sum-product operations. Thereby, this factor-graph takes into account the following conditions:

- Initial links with their confidence values which originate from different sources
- The trust values for these sources

- Symmetry and transitivity of the equivalence relation

For the simple example above, the resulting graph is depicted in Figure 7.

If it appears necessary that information about the same entities from several sources with different trust levels have to be integrated in the ebbits project, the idMesh system could be applied.

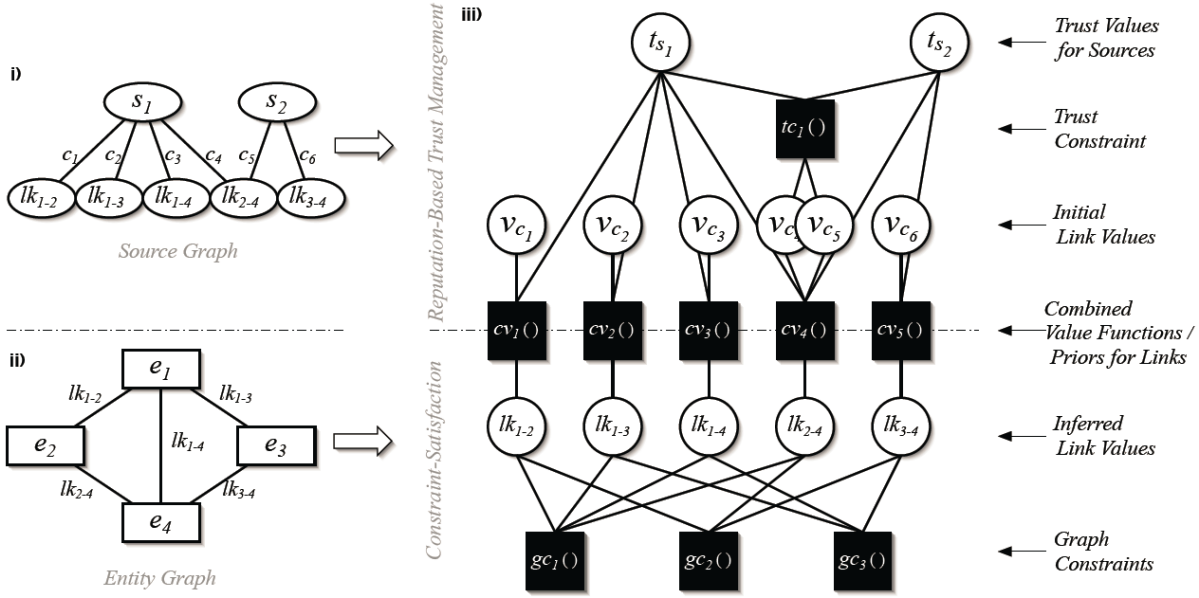

Figure 7: The resulting factor-graph with link values, trust values and constraints, Source: (Cudre-Mauroux, Haghani, Jost, Aberer, & de Meer, 2009)

7. Semantic Interoperability of Web Services

Web services technology is in focus of the ebbits project as a fundamental mechanism enabling flexible communication between physical devices and inner platform components in a highly modular system architecture. An interoperable interfacing and seamless exchange of data between various potentially heterogeneous web services is required in such a service-based system. The interoperability of web services is provided by a common underlying communication protocol and messaging system (SOAP, XML-serialized HTML, JSON) as well as by a standardized service interface, usually described in a machine-processable format of WSDL or in a lighter description of RESTful web API.

However, web services are often employed in applications requiring a composition of web services into more complex workflow structures and process chains. The interoperability specified on the level of protocols and interfaces then needs to be extended towards a more advanced description of the meaning of service inputs and outputs. This extension is typically handled by an approach of semantic web services (McIlraith, Cao Son, & Zeng, 2001), which enables referencing the IOPE parameters (i.e. inputs, outputs, preconditions, and effects) of particular services to semantic structures such as shared ontologies, conceptual service models, or Linked Data constructions (Payam Barnaghi, Bauer, & Meissner, 2011). Semantically described IOPE characteristics then enable a data exchange and workflow composition, i.e., orchestration and choreography, of internally heterogeneous web services that are generally grounded in possibly incompatible service implementations. The data mapping and transformation principle, enabled by semantic web service annotations and ontological domain models, is presented in Figure 8. The output message of WS1 is transformed to a concept C1, which can be transformed directly to an input message of WS2, or may be mediated to the concept C2 and then transformed to the WS2 input. This way, WS1 and WS2 are dynamically composed into a workflow, using the semantic data mediation and transformation.

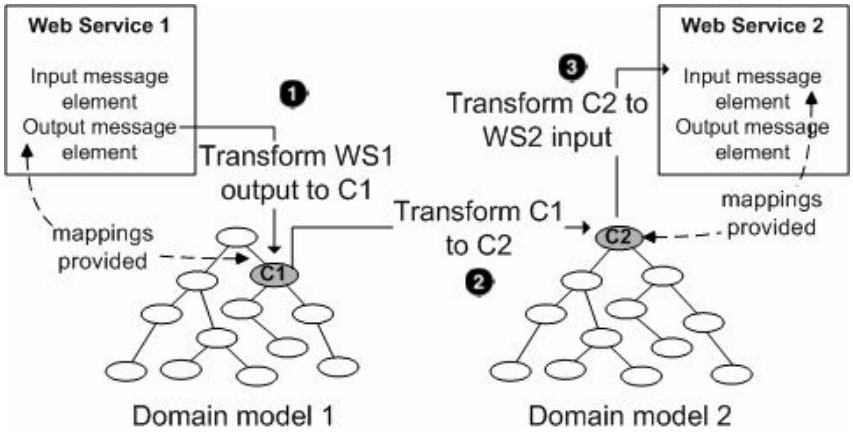

Figure 8: Semantic data mapping and transformation, Source: (Nagarajan, Verma, Sheth, Miller, & Lathem, 2006)

An outline of the most frequently utilized formalisms and frameworks enabling semantic interoperability of web services was provided in the Deliverable D4.2. It namely includes:

- **SAWSDL:** Semantic Annotations for WSDL and XML Schema (Farrell & Lausen, 2007), an extension of basic WSDL service descriptions. This formalism is supported by the SAWSDL4J API interface, Woden4SAWSDL parser, and Radiant eclipse plugin for creating and publishing SAWSDL and WSDL-S service interfaces.
- **OWL-S:** the Semantic Markup for Web Services (Martin D. et al., 2004), an ontology and/or formal language for semantically describing properties and capabilities of web services. By providing the generic concepts of Service Profile, Service Model, and Service Grounding, OWL-S supports web service manipulations such as the capability-based discovery, automatic composition, invocation, and monitoring of the service execution. Moreover, the OWL-S is compatible with SAWSDL descriptions. It uses WSDL as its grounding mechanism;

however, it is rather flexible and is not restricted to WSDL as the only service technology. A relatively wide range of tools enabling OWL-S development include editors, service composers, matchmakers, etc.

- **WSMO:** the Web Service Modelling Ontology (de Bruijn et al., 2005), a conceptual model specifically developed for describing semantic web services. WSMO framework is supported by a set of tools (WSMO Tools), including the WSMO Studio toolkit for editing and maintenance of all major WSMO components - ontologies, goals, web services, and mediators. However, despite its complexity, the development of WSMO was stopped in year 2005 and the framework is nowadays quite obsolete.

A more detailed comparison and benchmark of these frameworks can be found, for example, in (Nagarajan, Verma, Sheth, Miller, & Lathem, 2006).

With respect to the existing status of the design and implementation of the ebbits system, together with the underlying LinkSmart middleware, the OWL-S framework seems to be the best candidate for solving the semantic interoperability of web services in ebbits. This recommendation takes into account the OWL/RDF as the adopted formalism for ebbits ontologies, as well as the OWL-S-compatible OSGi-based web service implementation in LinkSmart. However, if testing on real-world ebbits data will indicate performance problems caused by the complexity of OWL-S, the SAWSDL framework can be employed as an alternative.

An example of employing the SAWSDL approach to support the semantic interoperability of OSGi web services in IoT applications is presented in (Gouvas, Bouras, & Mentzas, 2007). Semantic web service descriptions were particularly employed for handling data and message heterogeneities between interoperating services, search and discovery capabilities, and process composition.

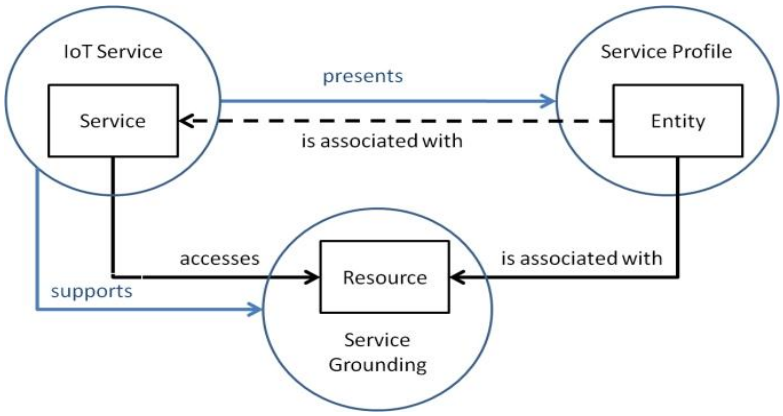

Figure 9: The OWL-S ontology adapted for the IoT domain, Source: (Payam Barnaghi, Bauer, & Meissner, 2011)

A broader semantic modelling approach for different components in an IoT framework is presented in (Payam Barnaghi, Bauer, & Meissner, 2011). The OWL-S is declared here as a minimalistic approach for describing semantic web services, which provides both rich expressive descriptions and well-defined semantics. Thanks to its flexibility, OWL-S was employed as an upper ontology for the IoT-adapted service model, depicted in Figure 9. The Service element is a web service interface that enables standardized interactions with other services in a process flow. The Resource represents a physical entity (i.e., a sensor, device, etc.) in the digital world, including the information on location, invocation, execution, etc., of the corresponding service. The Entity element is attached to a concrete device and contains a set of semantic attributes that characterize the service, including the IOPE parameters. In this model, physical entities and respective services are not connected through fixed links that are a part of the entity or resource models, but instead are linked through separately modelled associations. This way, a single service may be associated with multiple entities at the same time (e.g., a temperature sensor may provide the indoor temperature of a room and at the same time the ambient temperature of all the people who are currently in the room). The proposed OWL-S extension enables reasoning, discovery, and search facilities on all three elements, assuming that the services, entities, and/or resources are further linked to semantic representations of respective domain ontologies.

8. Semantic Interoperability within the Ebbits Architecture

The core ebbits components enable application developers to implement the logic of the applications mainly in the way of defining rules. This approach is very flexible, as it provides the possibility of creation of specific logic declaratively, what leads to the reduction of the amount of the implementation code. The rule-based systems are provided by:

- eventing subsystem using the rules for filtering, aggregation and storing the events
- context-awareness (and self-*) subsystem specifying the rules for the basic application behaviour
- business rules specifying the higher level application logic defined in terms of business goals which must be accomplished

Formally, rules are a declarative knowledge representation, which uses symbols or procedures in the assumption and consequence parts of the rules. These symbols usually represent the properties of the application entities, application states, actions to take, etc. As the ebbits components using the rules are implemented separately, in one concrete application they have to share the knowledge about the application entities in some common way. Otherwise, each ebbits component will have to implement the same data structures representing the application specific entities by itself. This often leads to redundant data structures representing the same application entities (defined for each component) and to possible errors caused by possible differences in the data structures representing the same entities by different components.

To ensure that all ebbits components will share the same knowledge representation of the application entities, a common knowledge representation must be provided. All knowledge is represented in the form of ontologies handled by the ebbits application ontology manager. The application ontology manager serves as the unique endpoint to all knowledge by providing dedicated knowledge retrieval services common to all ebbits components.

Semantic interoperability between the ebbits components can be ensured only in the case when this component shares the knowledge model and its interpretation (to work with information about the same entities in the same way).

This section aims to provide the conceptual design of the internal semantic interoperability mechanisms by defining the semantic models and the uniform access to the knowledge in various ways. To be able to describe the interoperability techniques as simple and clearly as possible, the conceptual design will be described using a concrete simple use-case. For this use case, the step-by-step processes and aspects of the interoperability mechanisms will be described by concrete examples. In the examples, the description of the ontology models and application rules will be described in the form of simple illustration pseudo-code.

8.1 The example use-case scenario

On the farm, there are pens and pigs moving in the farm area. Pigs may enter and leave pens. In each pen, there is a door RFID tag, which is triggered when a pig enters the pen. Each pen also contains a thermometer sensor measuring the temperature inside of the pen. An important business goal is to alert the maintenance crew, when there is a pig inside of the pen and the temperature in the pen is higher than a specified value. In this case, the maintainer responsible for this pen must receive a notification about this situation via SMS on his mobile phone. More formally, in a very simple readable way, the application logic for this simple scenario can be defined as follows:

```
BUSINESS RULE:
------------------
ON EVENT temperature-alert-in(pen)
      ACTION send-SMS-to-person-responsible-for(pen)
CONTEXT RULES:
```

```
--------------------
ON EVENT pen-door-tag-triggered(pen)
      RAISE EVENT pig-inside-the-pen(pen)
ON EVENT pig-inside-the-pen(pen) AND temperature-in(pen)> threshold
      RAISE EVENT temperature-alert-in(pen)
```

In this simple example, there is one business rule handled by the business rules component and two rules handled by the context manager. What is important to see is that the event raised by the context manager `temperature-alert-in(location)` is used by the business rules component. Thus, the term definition of the event must be shared by two separate ebbits component. Another important thing is that the information on the location, where the situation happens, in this case the concrete `pen`, must be attached to the events which are raised and pushed forward to the architecture to be able to track this information by other rules. The next important aspect is that the context information attached to the events, in this case the location, must be retrieved from somewhere and attached to the events somehow.

In the next sections it will be explained, how the ontologies are used to access and retrieve the knowledge about the application entities, how this information is attached to the events and used by the rules.

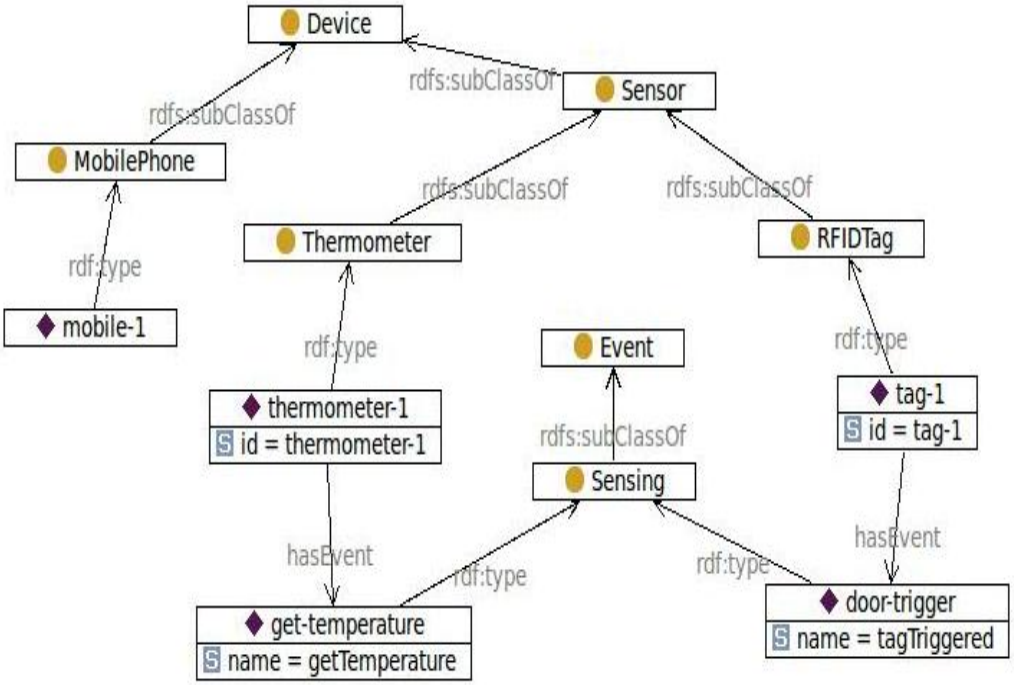

Figure 10: The use-case device ontology

8.2 The semantic knowledge model

Device ontology

First of all, it is necessary to know, what devices are used by the application and which events they provide. This knowledge is represented by the device ontology. In our use case, we'll need to represent three devices: the pen door RFID tag reader, the pen thermometer and the mobile phone of the maintenance person. The door reader and the thermometers provide the events to the ebbits

architecture. The ebbits eventing system is based on a publisher-subscriber model, so each event is publishing the values to the concrete channel, which must be also defined in the ontology. The identifier of the event channel serves as the main mapping point between the data infrastructure of the eventing system and the semantic model. When we know the publication channel of the event, we can use this information to access any information related to it in the semantic model. The device ontology for our use case is illustrated in Figure 10.

Event ontology

The semantic model describing the events contains the basic taxonomy and the representation of all events used in the application. It also includes the definition of related event properties, such as capabilities or payloads returned by the events. The event taxonomy is mostly used by the ebbits eventing subsystem requiring the information necessary for filtering, aggregation or storing the event data. The events defined in the model specify also the terms, such as application states, used in the assumption and consequence parts of the rules. This is required to share the event terms in a uniform way by all components. The detailed description of the event ontology and the semantic support of eventing subsystem can be found in the upcoming Deliverable D7.3.2 "Technical description of the implementation of Data and Event Management models 2". The event model used in our use case is illustrated in Figure 11.

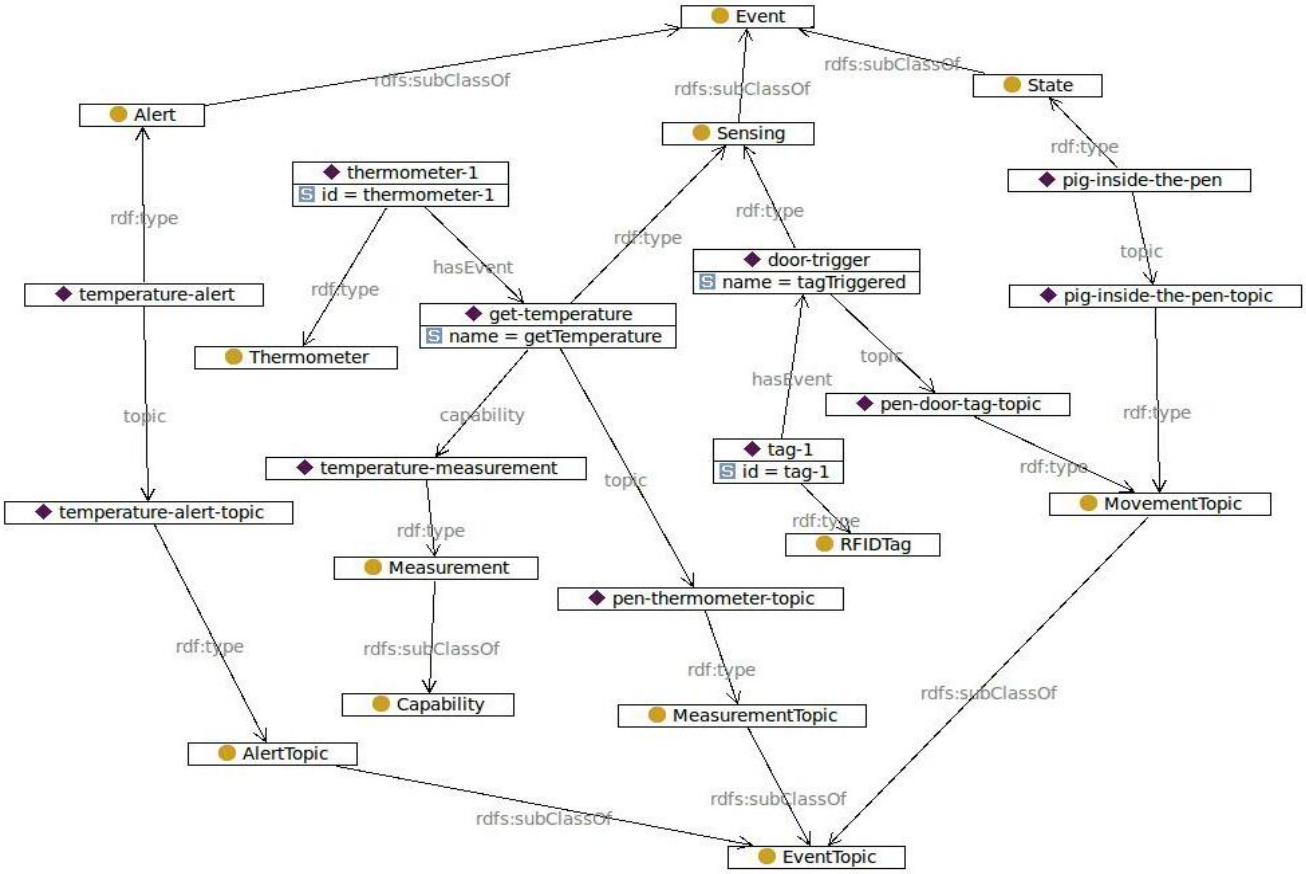

Figure 11: The use-case event ontology

Application (domain) ontology

The application ontology specifies the application specific domain model including all actors, context entities, their mutual relations and their relations to the device and event models. The application knowledge is mostly used by the rules and serves as the decision support information. In our use case it is important to represent, where the devices are located, the maintenance crew responsibility for locations and the ownership of the mobile phone device. The application ontology is illustrated in Figure 12.

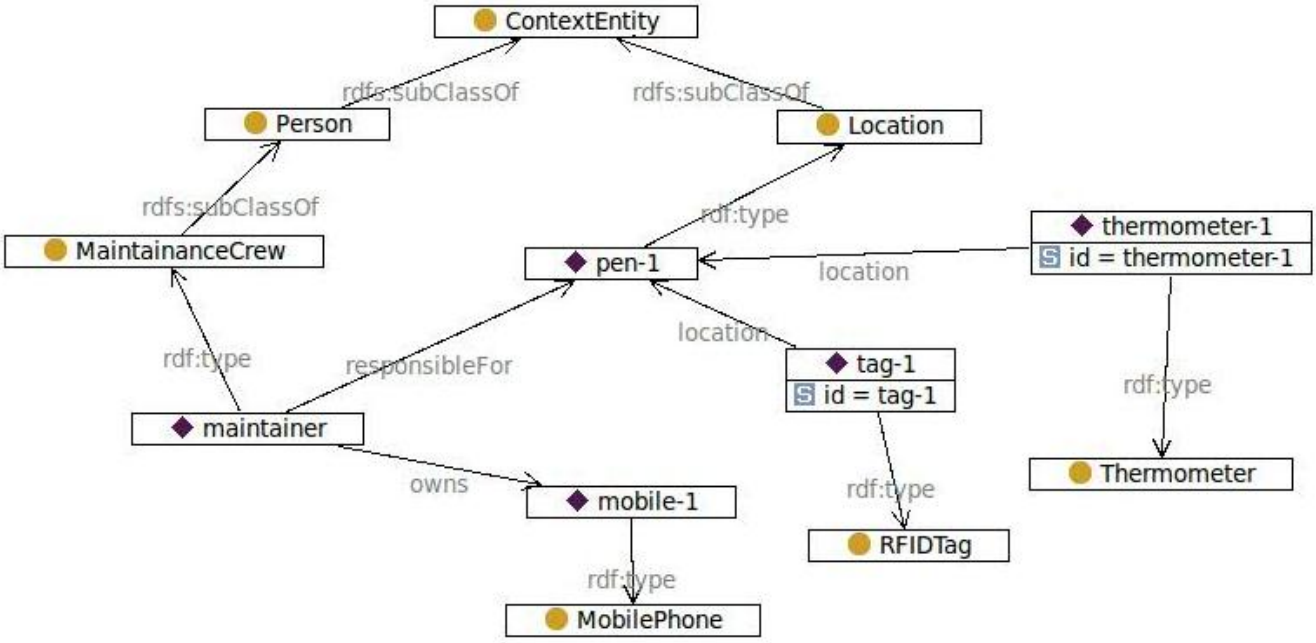

Figure 12: The use-case application ontology

8.3 The common access to semantic knowledge

In order to simplify the whole metadata retrieval process, the ebbits application ontology manager provides services enabling flexible access to semantic knowledge. Most of the scenarios require the retrieval of properties related to a concrete entity, such as device, service, event or context entity. The general signature of this service is specified as follows:

```
Ontology.getData(
      match: {property list},
      retrieve: {property list}           );
```

The `match` parameter specifies the properties, which must be matched for the searched entity. The `retrieve` parameter specifies the properties, which will be retrieved for entities satisfying the match parameter. The role of the `retrieve` parameter is to specify only relevant properties to be retrieved instead of the full model. This service aims to make the development of the rules easier. The developer will use the graphical tool, which will help to compose the query, instead of learning the whole SPARQL language. The property lists are specified as the path of the properties from the entity to the concrete property value (see examples below). The composition of the service call may be easily guided by the ontology using graphical tools, where the developer simply selects the relevant properties to match and to retrieve directly in the model browser.

The usage of this service will be explained using a simple example: The ebbits device proxies above the PWAL component listen to the publisher-subscriber channels, which are mapped to the concrete events of the concrete devices. For example, the thermometer device publishes the event to the channel `'pen-thermometer-topic'`. The ontology model of this thermometer is defined as follows:

```
thermometer-1
      id 'thermometer-1,
      rdf:type Thermometer,
```

```
      info
            manufacturer 'Thermometers, ltd.',
            device-model 'T X1',
      hasEvent
            name 'getTemperature',
            capability temperature-measurement
            channel 'pen-thermometer-topic'
      location pen-1
```

When the proxy receives the event from the concrete subscribed channel, it can retrieve all ontology metadata mapped to the name of the channel. Then the event information can be extended with the metadata, such as location, and the proxy may create the ebbits event, which is pushed forward to the infrastructure.

The example of getting the required information on the entity directly can be e.g. retrieving the location of the device having the event subscribed to the channel `'pen-thermometer-topic'`. The service call will look as follows:

```
Ontology.getData(
      match: {entity.hasEvent.channel: 'pen-thermometer-topic'},
      retrieve: {entity.location}         );
```

This service call will be translated to the following SPARQL query and executed against the model:

```
SELECT ?entityLocation WHERE {
      ?entity hasEvent ?hasEvent.
      ?hasEvent channel 'pen-thermometer-topic'.
      ?entity location ?entityLocation.         }
```

The service will retrieve the result:

```
 [{entity.location: 'pen-1'}]
```

If there would be more matching entities, the resulting list would have more elements.

In more complicated cases, the other mostly used scenario is to answer questions requiring a mutual relation of two entities. For this case, a more complex ontology service is provided:

```
Ontology.getData(
      entity-match: {property list},
      subentity-match: {property list},
      entity-retrieve: {property list},
      subentity-retrieve: {property list}       );
```

For example, the developer wants to retrieve the identifier of mobile phone device, which is owned by the maintenance crew person responsible for the location named `'pen-1'`. The service call will look as follows:

```
Ontology.getData(
      entity-match: {
            entity.type: MaintainanceCrew,
            entity.responsibleFor: 'pen-1',
            entity.owns: subentity        },
      subentity-match: {subentity.type: MobilePhone},
```

```
      entity-retrieve: {},
      subentity-retrieve: {subentity.id}              );
```

In this case, the most important fact is that the relation between the two entities is specified using registered keywords `entity` and `subentity`. Also, one of the clauses `entity-match` or `subentity-match` must contain the definition of inter-entity relation, e.g. `entity.isInRelationTo:subentity` or vice versa. This call is translated to the SPARQL query:

```
SELECT ?subentityId WHERE {
      ?entity type MaintinanceCrew.
      ?entity responsibleFor 'pen-1'.
      ?entity owns ?subentity.
      ?subentity type MobilePhone.
      ?subentity id ?subentityId.         }
```

It is assumed, that these two dedicated ontology manager services would cover most cases of ontology queries. Anyway, for more complex queries, there is still the generic SPARQL endpoint available:

```
Ontology.sparql(SPARQL query)
```

8.4 The ontology support for the rule-based systems

Meta-data retrieval and propagation

The assumption part of the rules mostly works with the definition of events, which trigger the rules. The events within the ebbits architecture are defined as a list of key-value pairs specifying the properties of the event. The event example including information about the event itself, the device, which has generated the event, and the additional context information (e.g. location) can be formalized as follows:

```
e = {       device-id: 'thermometer-1',
            channel: 'pen-thermometer-topic',
            location: pen-1,
            capability: 'temperature-measurement'     }
```

The rules may access or add any metadata of the events, to use them for the decision making or to push them forward to the eventing infrastructure. How to use the common knowledge model in the rule-based systems will be explained through the rules for the use-case scenario described in the introduction section. Let's start with the most simple context rule.

RULE 1: If the tag at the pen door was triggered, raise event that the pig is inside of the pen.

```
ON EVENT e[channel] = 'pen-door-tag-topic'
      RAISE EVENT {
            channel: 'pig-inside-the-pen-topic',
            location: e[location]               }
```

As the events are extended with the relevant context information retrieved from the semantic model, such as the location, the rules must be able to track this context information. In this case, the location of the device using the addressing: `e[location]` is attached to the newly generated event. This way, the context information added to devices or events can be automatically propagated through the whole rule system.

RULE 2: If a pig is inside of the pen and the pen temperature is higher than 25 degrees, raise the temperature alert for this pen.

This is a much more complicated example, where the rule system must check the temperature in a certain location. To achieve this, it must be able to find the device having the event that is capable of measuring the temperature and is located in required place. Then it must read the last actual values measured by this device for this related event. Thus the rule must use the ontology to find the device and then the component able to retrieve the measured values for the device event. The rule can be specified as follows:

```
ON EVENT e[channel] = 'pig-inside-the-pen-topic'
      EVALUATE
            model = Ontology.getData(
                        entity-match: {
                              entity.location: e[location],
                              entity.hasEvent: subentity    },
                        subentity-match: {
                        subentity.capability: 'temperature-measurement' },
                        entity-retrieve: {entity.id},
                        subentity-retrieve: {subentity.channel}         );


IF Data.getValue(
      model[entity-retrieve][entity.id],
      model[subentity-retrieve][subentity.channel],
      ) > 25 THEN
      RAISE EVENT {channel: 'temperature-alert-topic', location:
e[location]}
```

The device id and the channel of the relevant temperature measurement event are retrieved from the ontology using the location and capability properties. This information is used to retrieve the last measured values by reading the channel or by accessing the history data. The way of accessing information can be dependent on the event type. This information can be also defined in the ontology. Then the new `'temperature-alert'` is generated with the attached location information which was propagated from the lower level rules.

BUSINNESS RULE 1: If there is the temperature alert in some location, call the maintenance responsible for this location.

In this case it is required to use the ontology to retrieve the identifier of the existing mobile phone device owned by the maintenance crew member, who is responsible for concrete location. Then this identifier is used to call the application service for sending the SMS.

```
ON EVENT e[channel] = 'temperature-alert-topic'
      EVALUATE
            model = Ontology.getData(
                        entity-match: {
                              entity.type: MaintainanceCrew,
                              entity.responsibleFor: e[location],
                              entity.owns: subentity              },
                        subentity-match: {
                              subentity.type: MobilePhone
```

```
                                    subentity.hasService.type: SMSService},
                              entity-retrieve: {},
                              subentity-retrieve: {subentity.id}              );

IF isDefined(model[subentity-retrieve][subentity.id]) THEN
      ACTION Application.sendSMS({
            deviceId: model[subentity-retrieve][subentity.id],
            sms: 'Someting devilish happends at e[location]'      });
```

8.5 Ontology support for rule composition

To establish and ensure the interoperability between the different ebbits components, the first step is the definition of the commonly used vocabularies. These are formalized in the form of a semantic model which is shared and accessed by all ebbits components in a uniform way. The terms used in the assumption and consequence parts of the rules are employed following the ontology definitions. To enable this, the semantic infrastructure provides graphical tools helping the model-guided composition of the rules. When the developer creates a new rule, he or she will use the graphical event browser which will provide a complete guide by listing the events and their definitions following the developer's needs. These event definitions are then used in the rules. The ontology might also specify information, with which the concrete events must be extended in the consequence part of the rule. For example, the definition of the `temperature-alert` event may contain the requirement that this event must be extended with the `location` information, when it is created. The same graphical tools will enable the ontology-guided creation of the properties, which are required when using the `Ontology.getData()` service. This way, all terms are used in a uniform way in the whole ebbits architecture.

9. Conclusions

Semantic interoperability for the ebbits project can be achieved by using a common terminology throughout the whole project. Therefore, a design for using common vocabulary and instances for all the components in the ebbits project is presented, which has to be extended in the future by Work Package 4 with the help of the ebbits application partners. This is similar in spirit like the Entity Name System of the OKKAM project, but without building up an additional infrastructure.

For the special case of web services, the OWL-S language is proposed.

For these remaining cases, where a matching algorithm has to be applied, the ebbits project does not want to develop a completely new system, but instead apply the established results from the literature of years of research. For the case of matching data in the RDF format, we mainly propose the SILK system which is openly available under an Apache 2 license. It is declarative, supports both, lexical and structural matching and assists in the evaluation of links through a web interface. Depending on the obtained outcomes, especially regarding efficiency, further results from the LIMES system can be considered. Furthermore, if it appears necessary that information from several sources with different trust levels have to be integrated, the idMesh system could be applied.

10. References

Bouquet, P., Stoermer, H., Niederee, C., & Mana, A. (2008). Entity Name System: The Backbone of an Open and Scalable Web of Data. *Proceedings of the IEEE International Conference on Semantic Computing*, (S. 554-561).

Cudre-Mauroux, P., Haghani, P., Jost, M., Aberer, K., & de Meer, H. (2009). idMesh: Graph-Based Disambiguation of Linked Data. *Proceedings of the World Wide Web Conference*, (S. 591-600). Madrid.

de Bruijn, J. e. (2005). *Web Service Modeling Ontology (WSMO)*. Von http://www.w3.org/Submission/WSMO/ abgerufen

Elmagarmid, A., Ipeirotis, P., & Verykios, V. (2007). Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* .

Farrell, J., & Lausen, H. (2007). *Semantic Annotations for WSDL and XML Schema*. Von http://www.w3.org/TR/sawsdl/ abgerufen

Gouvas, P., Bouras, T., & Mentzas, G. (2007). An OSGi-Based Semantic Service-Oriented Device. *In Proceedings of the 2007 OTM Confederated international conference on the move to meaningful internet systems*, (S. 773-782).

Health, T., & Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space.* Morgan & Claypool.

Hreno, J., Nutakki, P., Knechtel, M., Furdik, K., & Sabol, T. (2011). *Knowledge representation formalism analysis.* ebbits Project Deliverable D4.2, FP7 project 257852.

Jean-Mary, Y., Shironoshita, P., & Kabuka, M. (2009). Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 235-251.

Köpcke, H., & Rahm, E. (2009). Frameworks for entity matching: A comparison. *Data Knowledge Engineering*.

Li, J., Tang, J., Li, Y., & Luo, Q. (2009). RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering*, 1218-1232.

Martin, D. e. (2004). *OWL-S: Semantic Markup for Web Services* . Von http://www.w3.org/Submission/OWL-S/ abgerufen

Martin, Y., & Ahlsén, M. (2011). *State-of-the-Art design and implementation of systems with persistent and high volume data.* ebbits Project Deliverable D6.2, FP7 project 257852.

McIlraith, S., Cao Son, T., & Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46-53.

Nagarajan, M., Verma, K., Sheth, A., Miller, J., & Lathem, J. (2006). Semantic Interoperability of Web Services – Challenges and Experiences. *In Proceedings of the IEEE International Conference on Web Services ICWS '06*, (S. 373-382).

Ngonda Ngomo, A.-C., & Sören, A. (2011). LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, (S. 2312-2317).

Noessner, J., Niepert, M., Meilicke, C., & Stuckenschmidt, H. (2010). Leveraging Terminological Structure for Object Reconciliation. *Proceedings of the ISWC Workshop 2010*, (S. 142-149).

Payam Barnaghi, S. D., Bauer, M., & Meissner, S. (2011). Service Modelling for the Internet of Things. *In Proceedings of Computer Science and Information Systems*, (S. 949-955).

Veshcheva, Y. (2011). *Object Matching für Linked Data.* Master Thesis at Universität Leipzig (In German).

Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). Discovering and Maintaining Links on the Web of Data. *Proceedings of the International Semantic Web Conference*, (S. 650-665).