SEMbySEM: a Framework for Sensors Management

Jean-Sébastien Brunner, Jean-François Goudou, Patrick Gatellier, Jérôme Beck, Charles-Eric Laporte

Theresis Innovation Center - Thales Security Solutions & Services
Campus Polytechnique - 1, avenue Augustin Fresnel
91767 Palaiseau cedex - France
jean-sebastien.brunner@thalesgroup.com
jean-francois.goudou@thalesgroup.com
patrick.gatellier@thalesgroup.com
jerome.beck@thalesgroup.com
charles-eric.laporte@thalesgroup.com

Abstract. This document presents the SEMbySEM project aiming to provide a framework for universal sensors management by semantics. The entire scope from the sensors description to the End-users display is addressed, including sensors connection and events handling, system ontology, business rules design, graphical models and End-users display. Within the course of the project a new semantic standard dedicated to system management is defined according to business requirements and addressing the semantic description of managed objects as well as the means to bind the actual entities to their conceptual counterparts.

Keywords: Semantic Web Technologies, Sensor Web, Ontologies, Rules, Sensors, Internet of things.

1 Introduction

With the advent of what is commonly described as the “Internet of things”, the trend toward a world of sensors is becoming everyday more obvious as many current life objects become equipped with embedded data and communication capabilities (like RFID tags). In this “world of sensors”, the semantic sensor web is a framework aiming to provide ways to process the huge amount of data they will produce.

Our work targets the end-user point of view. From an end-user point of view, the information provided by a set of sensors is only meaningful within the scope of some end-user activity, targeting a defined goal achievable via a dedicated scenario.

The SEMbySEM project aims at defining tools and standards for the management of systems defined as coherent set of objects and grounded on a semantic abstract representation of the system to be supervised or managed.

This abstract representation has two purposes. The first one is to isolate the technical issues related to the communications with the various sensors, in what we call a Façade Layer. This Facade layer transforms the data coming from these sensors into semantic information and allows end-users to focus only on their activity while ignoring the technical details of each sensor. The second purpose is to be able to work directly on a semantic model of the system consisting of dynamically updated ontology plus related business rules (i.e. production rules). In this way, the multiple sensors data is linked to concepts of the system using a well-defined level of granularity. For instance, sensors will be grouped together if they belong to the same object, or if they are in the same location.

In order to define the ontology and the business rules a need for a new semantic representation appears, as the systems to be managed are intrinsically dynamic. A main need in the semantic model is the possible actions on real-life objects, as sensors may also be linked to actuators.

2 Related work

Sensor Web has gained interest due to hardware and communication advances (generalization of technologies such as RFID, geo-localisation, extension of internet-connected devices) and needs for standards to allow more interoperability between the various types of sensors. The Open Geospatial Consortium[1] developed a framework of standards for Sensor Web Enablement (SWE). This standardization effort enables the use of a neutral format to define the various sensors and systems, their interfaces, the type of information they convey and their communications. However, SWE standards are syntactic and do not embed logical expressivity for inference. Therefore the logic of the managed system, defining how the various sensors combine their information together to represent complex objects, needs to be embedded in the core of applications.

On another hand, Semantic Web standards, developed by the World Wide Web consortium[2], are able to represent complex knowledge, including logic associated to the data. RDF [5], as a neutral format for data representation, enables communication and storage in a neutral format. Based on this format, OWL [4] permits to define ontologies, i.e. the conceptualization of a given domain. While this format allows the definition of a model, it also enables the use of Description Logic (DL) to partly defines the behaviour of the system. For instance, Description Logic defines the notion of *Restriction*, allowing the definition of dynamic classification; instances are classified in a class as soon as they match given criteria (e.g. a given *train* is classified in the *Late Train* class as soon as it has some *Delay*).

Since DL is sometimes below the expressivity needs for real systems representations, several proposals were developed to extend it with rules in order to embed more business logic in the model itself and not spread this additional logic in software code. SWRL [6] was proposed as extensions to this model, but is felt insufficient since the expressivity of the rule and the expressivity of the DL model can

[1] OGC, http://www.opengeospatial.org/
[2] W3C, http://w3c.org/

lead to undecidability [16]. These standards also suffer from lack of skill from users who are not familiar with knowledge representation and Description Logics.

From a corporate point of view, while production rule engines are already widely spread in enterprise applications they are not yet fully integrated with semantic models. Moreover, rules suffer from heterogeneity of expressivity (Production Rule, Logic Programs) and heterogeneity of formats. Several standardization processes are on-going, such as the JSR94 standard (addressing rule interoperability at Java level) and, at a more general level, the OMG Production Rule Representation (PRR) [9] and W3C Rule Interchange Format[3] (RIF) proposal for a rule interoperability language[4].

In term of general framework for Semantic Sensor Web, different works highlight the added value of semantics, such as [1,2,3]. They propose different architectures gathering SWE, Ontology and Rules to process sensor data. These standard-based prototypes illustrate the added-value of such architecture to answer concrete use-cases. However they not address the soundness of the system, the scalability issue and the user interaction in the system.

Scalability issue mainly comes from the reasoning engine, able to apply the logic of the model. This issue comes from the complexity of the algorithms based on DL (e.g. NExpTime-complete) and of logic programming rule systems.

Regarding user interaction, these systems focuses on monitoring applications and do not allow to perform action on the underlying systems linked by sensors. Sensors can be available as Web Service, but current SSW architecture does not take into account their potential operations. In particular ontologies do not include the notion of action. In this area, Semantic Web Service attempts to add semantic metadata to the Web Services standards. Some standards such as SAWSDL[7] or OWL-S[8] propose different supports of the semantics in Web Services. The first one allows semantically annotating the service when OWL-S allows to entirely define the service using semantic concepts. In the case of OWL-S it is then possible to define the goal of the service and how to perform some processes.

Based on these assessments, we propose a framework able to go beyond the observed limitations, that is to say able (1) to provide a generic communication layer with sensors, (2) to semantically define a model and its logic to aggregate information from various sensors, (3) to allow the definition of the model of the managed system by business experts thanks to a targeted standard, (4) to deal with large scale systems, (5) to perform actions on objects connected to sensors and (6) to display a pertinent interface to End-users.

[3] W3C RIF (Rule Interchange Format) working group, http://www.w3.org/2005/rules/wiki/RIF_Working_Group

[4] There is an overlap in scope between W3C RIF and PRR. While PRR focuses on the standard metamodel definition and modeling of production rules with an XMI format, RIF focuses on a rule interchange format based on XML for web applications and also defines interactions between ontologies and rules, see [9] for more details.

3 The SEMbySEM Project

3.1 Project Overview

SEMbySEM (SErvices Management by Semantics) is a 30-months European project carried out under the EUREKA ITEA2 framework and begun July 1st, 2008. This project aims at creating a lightweight, adaptive monitoring software system dedicated to the management of systems of all sizes. The Human-Machine Interface (HMI) will be dedicated for each End-user’s “business role”, displaying to each End-user only the pertinent information about the monitored system.

The software core of SEMbySEM will constitute the initial contribution of an Open Source project aiming to promote the use of domain specific semantics for the management of large systems in various domains like logistics, computing and system monitoring.

Supervision software dedicated to future systems need to be easier to deploy and to maintain than the present ones, while addressing the increasing complexity of “systems of systems” and keeping an overall management capability for the users. The approach envisioned for SEMbySEM to address this issue is the extensive use of semantics in the system description allowing the active contribution of expert users for the monitoring system design and configuration.

The SEMbySEM project is based on the definition of two standards and several tools:

- A MicroConcepts standard for the semantic description of manageable objects and a standard allowing the mapping of real world Manageable Objects to MicroConcepts;
- A consistent set of tools including a common software framework comprising runtime tools and authoring software.

The targeted managed system size is between one thousand and one hundred thousand of concepts instances with ten thousands rules.

3.2 Project Limitations

The project is mainly dedicated to event-based supervision, aiming at hiding any technological issue under a semantic abstraction layer and specific HMI for each End-user. This framework is very flexible and can be extended for further applications depending on specific needs.

Some limitations will appear in the first version of the project. This one will mainly focus on ontological system representation and rules reasoning. For instance planning or workflow processing are not included in the SEMbySEM framework. The second drawback, common to all event-based systems, is that commands from End-users may not be available to sensors as they will not be connected or available at any time.

3.3 Illustrative Use-Case

An illustrative Use-Case is the management of sensors in a railways station. In a station, several sensors may exist notably for building management and security (smoke sensors, doors sensors, ...) or for the operations of the station (sensors in the engines and wagons,...) Managed objects also exist, that are linked to sensors and on which actions are also possible: escalators, lifts, cameras, live departure boards, TV screens and the station announcement system.

Sensors are accessed by End-users through a representation of managed objects and local grouping: security officers consider rooms or areas more than sensors. A train is not a physical object, it is a railways-domain concept composed of engine(s) and wagons and having its own properties (number, schedule, etc.). Therefore sensors composition and abstraction are mandatory from a business point of view.

Actions may be done on managed objects. Cameras can be rotated, doors can be closed, live departure boards are regularly modified. Therefore the Actions on managed objects are also to be considered when we design such a system. Sensors are not enough to describe this system, as only bottom-up information collection is insufficient.

Any automatic procedures that are embedded in the existing information system can be expressively described in rules. For example, when a fire alarm is triggered, the fire doors close automatically. Describing such rules in the system is interesting from a business point of view.

4 Semantics of the system

4.1 MicroConcepts, a business-driven standard for representation of objects

The definition of a semantic model able to deal with the specificity of Sensor Web is important. As mentioned earlier there are two trends to model sensor web data. First is to use OGC syntactic standards, which are specifically designed for sensors but lack for semantics, and other trend is to use Semantic Web standards such as OWL to bring semantics to the definition.

Before choosing any standard we started a bottom-up analysis of the business needs to propose a business-driven solution and eventually chose or design an appropriate standard. We firstly pointed out the need of a high level standard to allow easy system management by end-users, receiving semantic information from the Façade, itself connected to sensors. Our need was then to define the semantics used by experts compared to the needs.

Our study shown that OWL and the use of Description Logic are difficult to handle by business experts. In particular, users familiar with enterprise data management and more specifically databases are confused with the Open World Assumption[5]

[5] Definition from Wikipedia: In formal logic, the **Open World Assumption** is the assumption that the truth-value of a statement is independent of whether or not it is *known* by any single

principle. The use of Close World Assumption and Unique Name Assumption[6] enables a better adoption of this standard since it is closer to databases and more generally to enterprise data management, compared to Open-World-Assumption which targets open web environment. In this context, various OWL axioms can be transformed in DB-like constraints as proposed in [10] and experimented in [11] to ensure the consistency of the model.

Additionally, OWL expressiveness is somewhat limited to express some business needs because models are often very sophisticated. In particular qualified cardinality restrictions, property composition roles and efficient management of n-ary relationships and meta-modelling are missing compared to some real business needs. At the time of our study, OWL 2 working group published a working draft of the next OWL standard [12], extending the language by a number of new features such as qualified cardinality restrictions, property composition roles, definition of interval restriction for literals, etc. and then answering to several of our needs.

Compared to our needs, further extensions can be proposed, notably Advanced Property Composition[7] (saying for example that a property value of an instance equals the average/min/max/sum of some of the value of its components), Actions enabling acting on objects (for example "start" or "stop" a device managed by the system) and Parameters.

We then defined a business-oriented model, named MicroConcept, developed in the scope of the SEMbySEM project. This is a business-driven standard to be publicly released, and comprising a limited set of axioms. The main ones are the following:

- Ontology, as container of all objects of a given domain.
- Concept, as classifier for objects sharing some common features.
- Property (with object or literal value), defined independently from concepts and then able to be used in different classes. Property can use:
    - Domain.
    - Range.
    - Cardinality restrictions.
    - Qualified Cardinality Restrictions.
    - Properties of properties (transitive, symmetric, etc.)

observer or agent to be true. It is the opposite of the closed world assumption which holds that any statement that is not known to be true is false. […] Semantic Web languages such as RDF(S) and OWL make the open world assumption. The absence of a particular statement within the web means, in principle, that the statement has not been made explicitly yet, irrespectively of whether it would be true or not, and irrespectively of whether we believe (or would believe) that it is (or would be) true or not. In essence, from the absence of a statement alone, a deductive reasoner cannot (and must not) infer that the statement is false.

[6] Definition from Wikipedia: The **Unique Name Assumption** is a concept from ontology languages and Description Logics. In logics with the unique name assumption, different names always refer to different entities in the world. The ontology language OWL does not make this assumption, but provides explicit constructs to express that two names denote distinct entities [4].

[7] Advanced Property Composition was part of OWL 2 discussions but seems not appear in latest working drafts.

    - Default value for properties.
    - Static values (values shared by all instances of a concept).
    - Property composition (a property value is equal to the property value of a linked component).
    - Advanced property composition (similar to previous one but using mathematical functions).
- Concept and property subsumption to define inheritance.
- Instance of a concept.
- Enumeration.
- Action, defining the way to act on the real object represented by its instances.
    - Actions have input and output parameters.
- All elements contain identification (unique ID), versioning, localized name and description.

4.2 Adding rules to MicroConcepts

Rules bring added-value by avoiding spreading the business logic across company models, code and documentation. It ensures the uniqueness of the behaviour attached to semantic objects. A drawback of this approach is that the addition of a rule language on top of an ontology language (such as OWL) can lead to inconsistency because axioms of the language and rules can affect each others. Different approaches were proposed such as Semantic Web Rule Language (SWRL)[6] and Description Logic Program (DLP) [13]. SWRL extends OWL with rules in a non-native way; in the DLP approach, the intersection of Description Logic and Logic Program is used, using only a subset of DL but providing a better computability.

For higher scalability we developed the MicroConcept standard in order to be used with a production rule engine (such as JESS or DROOLS) implementing RETE [14] algorithm. The scalability of such approach was proven and enables its use in industrial environment as RETE-based algorithms are already used in many enterprises.

In order to cope with the heterogeneity of rule standards, we define rules in a neutral format linked to the MicroConcept standard. In particular, the rules are able to directly address the semantic objects of the model (concepts, instances, properties) and benefit from the logic of the model: for instance, if a rule uses a concept, the matching is done for the more general concept as well.

Integration of a RETE-based rule engine, giving good performances is then smooth.

4.3 Implementation strategy of the MicroConcept standard

Our studies enable us to design specifications and language semantics of the MicroConcept standard, based on the needs expressed by real use-cases, without limitation to existing standards. Compared, for example, to OWL 2, MicroConcept adds several axioms (notably Action and Advanced Property Composition) and moreover uses the closed-world and unique-name assumptions.

Besides these differences, we want to leverage existing standards for the implementation in order to benefit from existing design tools, API, serialization forms and repositories. We identified two strategies of implementation:

(1) Directly define MicroConcept based on MOF 2 [15] models (an OMG recommendation). Similar to OWL2, whose structural definition is based on the MOF, our language can be expressed in term of MOF meta-model, giving it a formal, computable definition.
    As a result, MicroConcept is a meta-model and can be serialized in XMI format, edited with compliant editors (such as UML tools with an appropriate profile), and moreover can benefit from a powerful programmatic environment. In particular we can benefit from technologies such as model transformation implemented in the Eclipse Modeling Framework[8]. This ensures to limit specific code to the minimum and to be able to maintain the standard in the future.

(2) Define MicroConcept based on OWL 2 meta-model. In this case, our standard represents a meta-ontology which can be instantiated by business ontology taking the benefits from all the logic of the standard and from all the tools developed around this language: parsers, inference engines (e.g. Pellet[9]), programmatic environment (such as Jena[10] or OWL API[11]) and repositories.
    Extensions proposed in our standards (in particular Action) are addressed by the rule engine and by a set of rules not editable by users, given a way to easily maintain the standard and be able to make some evolution. Closed-World-Assumption is addressed by the specific architecture of the core of the application (Cf. subsection 5.4).

4.4 Illustrative examples

We give here Micro-Concepts and rules for the illustrative use-case presented in section 3.3. Full specifications of these languages will be published later on the project website[12].

4.4.1 Micro-Concepts

The following Micro-Concepts are defined:

- Train
- Engine
- Wagon
- Station
- Camera

[8] http://www.eclipse.org/modeling/emf/
[9] http://clarkparsia.com/pellet
[10] http://jena.sourceforge.net/
[11] http://owlapi.sourceforge.net/
[12] http://www.sembysem.org

- Light

4.4.2 Properties

The following Properties are defined:

- **speed**: relation possessed by a Train or an Engine with a decimal value.
- **serialNumber**: relation possessed by an Engine or a Wagon with a string value.
- **trainNumber**: relation possessed by a Train with an integer value.
- **hasEngine**: relation possessed by a Train with a value that is an instance of Engine.
- **hasWagon**: relation possessed by a Train with values that are instances of Wagon.
- **inPlatform**: relation possessed by a Train with a value that is an instance of the Platform.
- **hasLight**: relation possessed by a Platform with values that are instances of Lights.
- **hasCamera**: relation possessed by a Platform with values that are instances of Camera.

4.4.3 Actions

The following Actions are defined:

- Engine has **'Start'** and **'Stop'** actions.
- Camera has a '**Focus_on_platform**' action. This action has a parameter '**to_platform**' taking an instance of '**Platform**' as parameter.
- Light has '**Switch_On**' and '**Switch_Off**' actions.

4.4.4 Rules

Rules can be defined directly on top of MicroConcepts. We give as example the expression of the rule "If a train arrives at a given platform, turn the camera to that platform and switch on all the lights on this platform". This rule used the proposed rule serialization.

```
rule "TrainInPlatform"
if
{
// If a train arrives at a given platform
?t := Train (?tPlatform := inPlatform, ?cams := one(hasCamera), ?lights :=
one(hasLight) )
}
then
{
```

```
// Then turn the camera to the given platform
?CameraFocusAction := createAction(?cams, Camera/Focus_on_platform);
?CameraFocusAction->to_platform := ?tPlatform;
execute(?CameraFocusAction);

//Then switch on the lights of this platform
excecute(?lights, Light/Switch_On);

}
```

5 SEMbySEM general architecture

Let us consider an existing set of communicating objects or elements, constituting what from now we will call indifferently a *universe* or a *managed system*. This universe will be monitored with sensors dispatched on several fixed locations and on some moving objects. The deployment and operational use of a management system for this universe will be done in two phases, design time and runtime. Design time operations will encompass the detailed definition of all the sensors which can contribute to the universe, the ontology of the universe including all the existing and required concepts related to the universe sensors and their associated business rules, and the viewpoints of each stakeholder including a display HMI. Runtime will be the operational use of the management system controlling this universe.

5.1 Design time

The design is intended to be done by expert users in the domain, assisted by ontology designers, rules designers and sensors communications designers.

Firstly, the ontology definition concerns the mandatory concepts defining and operating the universe, including first the objects that are managed and on which sensors acquire data, objects composed from several elementary objects and abstract objects that correspond to business concepts. The associated rules to permanently update the ontology are a whole part of the universe dynamic model. The ontology must also support the actions defined on the concepts and linked to actuators on the real managed objects.

The sensors definition includes all the sensors that can be encountered within the universe from an operational point of view, meaning the communication protocols to access them, the type of communication they support, the kind of message they deliver, the potential actions on the managed objects, the operational flow rate of data, an identifier to the associated concepts in the ontology, etc.

The stakeholders' viewpoints definition includes all the graphic data (icons, widgets, buttons, etc.) and the links to the related semantic data (in the ontology). These two features are grouped in several HMI models, each model containing one or several different views. Each model corresponds to a set of End-users and will present only pertinent information for this set of users.

5.2 Runtime

During runtime the dynamicity of the managed system is very important. The fixed and mobile sensors emit messages when an event occurs or when they are scheduled for it, while End-users connect and disconnect through their interfaces, act on the managed objects or on virtual objects in the semantic model. Each event from sensors is registered and processed in order to update the semantic model through direct modification and modifications triggered by the business rules. The display of the connected End-users must be updated accordingly when the semantic model updates are pertinent for them. Each action from an End-user or from rules is processed internally and sent to the right managed object when necessary.

5.3 Overall architecture

The architecture we have retained to address these issues is composed of three layers: the Façade layer, the Core layer and the Visualisation layer. The Façade layer is the interface with sensors and the Visualisation layer is the interface with End-users. The Core layer contains the semantic model.

The goal of the Façade layer is to be the interface between the sensors and the semantic model. All the technical diversity concerning protocols, communication matters, sensor types and so on is addressed in this layer. The Façade transforms heterogeneous messages and events from sensors to standardized messages addressed to one or several concepts transmitted to the Core layer. The Façade also transforms actions messages from the Core to the actuators.

The Core processes the events from the Façade in order to maintain an up-to-date semantic model of the universe. For this the arrival of a message from the Façade triggers a short process: identification of the concept instance related to the message or creation of this instance if it does not exist, consistency validation of the update with regards to the model requirements and update of the semantic model. Afterwards, the rule engine is called, taking as input the successful model changes and processing until no rule is left to trigger. The second main task of the core layer is to send the pertinent semantic data to the Visualisation layer. Each time an End-user connects to the system, the Core layer is notified of the semantic concepts instances requiring data display. Then each time these instances are updated the data is also sent to the Visualisation layer until the End-user disconnects.

The Visualisation layer aims at displaying to the End-users the pertinent information they require to perform their task. Therefore each End-user has access to tailored viewpoints, designed by expert users and HMI experts and displaying data from the semantic model. This information is continuously updated each time an event occurs. The End-users may also perform actions on the instances of the semantic model through their HMI. The Visualisation layer performs several tasks: it gets all the semantic data that is of interest for the End-user and links it to graphical components for display, according to HMI models.

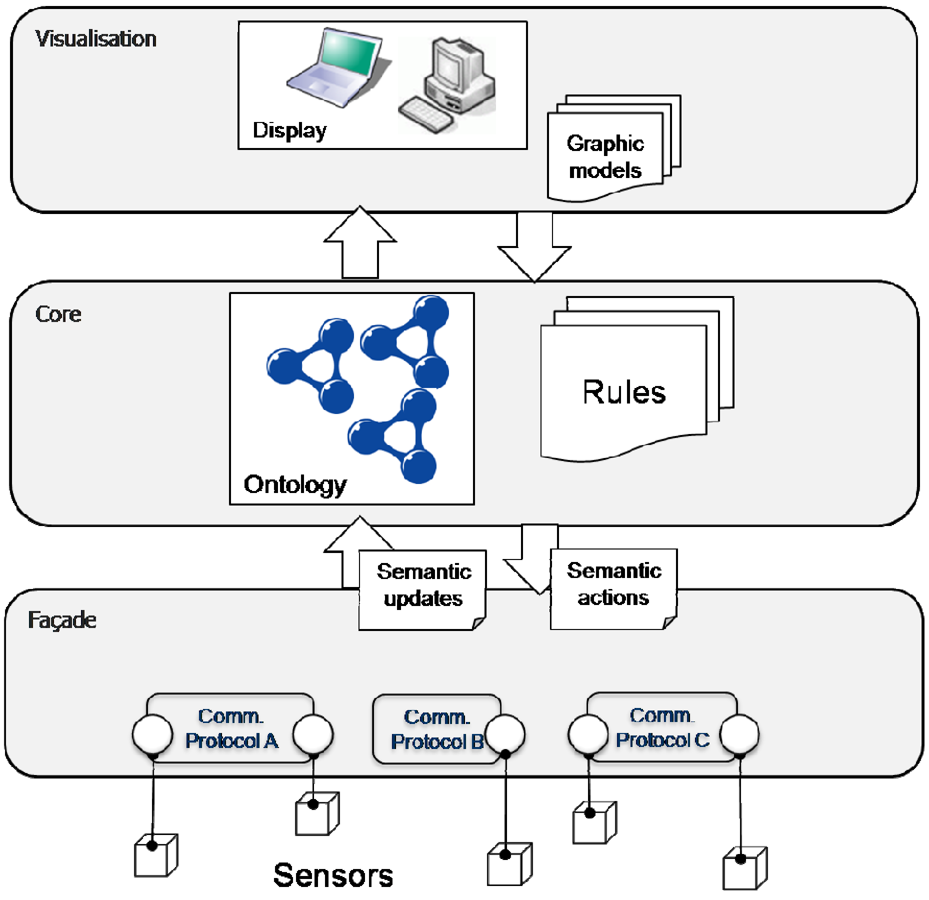

Fig. 1. Overall SEMbySEM architecture

5.4 Architecture of the semantic processing layer

In the two implementation strategies described at section 4.3, the embedded logic is not exactly the same as OWL, especially regarding the concept of Closed-World-Assumption. In this context, we use three levels to process the logic of our model.

First, a Constraint Checking module is responsible for consistency checks similarly to DB-style constraints [10]. This module ensures the consistency of the model in the Closed-World-Assumption, it is applied, in particular, on Cardinality (e.g. if a property has a *MaxCardinality* of 1 and has already one value), and Functional Property.

Secondly a reasoner is responsible to apply the general logic of the MicroConcept model. This module expands the asserted data with inferred data resulting of classification, use of property composition, symmetric, inverse property, etc.

Finally, a rule engine based on RETE algorithm, applies the additional business logic defined by the business user.

Additionally, a query engine is responsible to handle queries received from the visualization layer. It interprets the query and answer according to the logic of the model (already inferred by the 3 previously described modules since we use forward chaining inference).

The model itself benefit from the advantage of the chosen implementation strategy. In particular memory, disk representation, serialization and persistency use state-of-the-art standards to provide a powerful and maintainable solution.

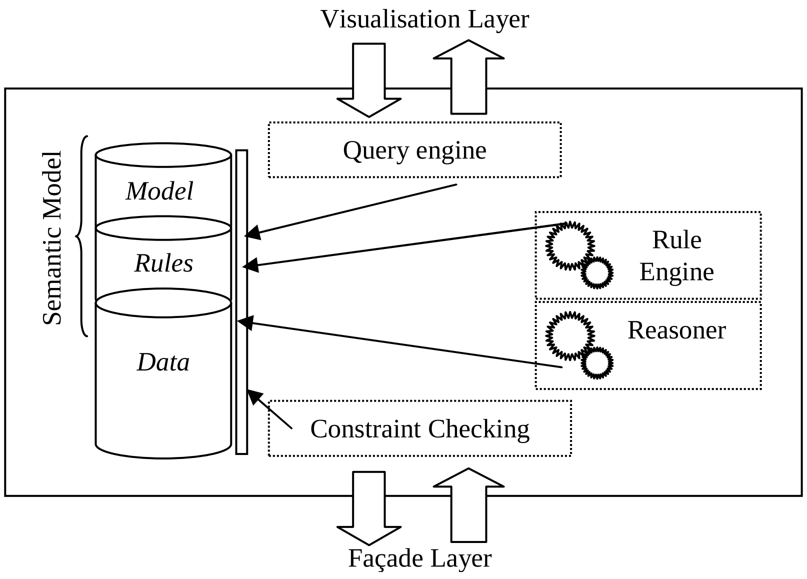

Fig. 2. Core layer general architecture

6 Current status of the project

At the time of the redaction of this paper, the SEMbySEM project is still in its first year. Architectural choices had been done as well as functional and technical specification of most parts of the framework. The MicroConcept standard was drafted and will be checked against the use-cases before release.

The project starts now its development phase. First results and evaluations are expected at the end of this year.

In order to foster SEMbySEM standard and framework, an open-source version of the framework will be released in early 2010. Standards and framework will be available on the official website of the project (http://www.sembysem.org) where additional information will be added progressively.

7 Conclusion and future work

We have presented here the whole idea of the SEMbySEM project aiming at the creation of a semantic infrastructure for service management. The main idea is to use a business-driven standard called MicroConcept to define the semantic model linked to sensors and manageable objects. MicroConcept was designed according to business needs but will be implemented with respect to state-of-the-art standards in order to provide both the expressivity required to model the use-case and the scalability to implement them. Additionally, a production rule engine supports the business logic in order to minimize specific developments.

In upstream of this core system, sensors and manageable objects low-level communications are transformed by the *Façade layer* to feed the semantic model.

In downstream, users can access to the system through a visualisation layer performing queries to the semantic model and supporting actions from users to the system.

This architecture enables a powerful framework able to answer to a large variety of use-cases. The implementation phase is starting and will helps to validate all the architecture presented in this paper.

8 Acknowledgements

This work is carried out by the EUREKA ITEA2 project SEMbySEM partly funded by French, Spanish, Finnish and Turkish governments. Furthermore we want to thank all partners and contributors to this project.

9 References

1. Sheth A., Henson C., Sahoo, S., "Semantic Sensor Web," IEEE Internet Computing, vol. 12, no. 4, pp. 78-83, July/Aug. 2008, doi:10.1109/MIC.2008.87
2. Huang, V. and Javed, M. K. 2008. Semantic Sensor Information Description and Processing. In *Proceedings of the 2008 Second international Conference on Sensor Technologies and Applications - Volume 00* (August 25 - 31, 2008). SENSORCOMM.
3. Imai, M.; Hirota, Y.; Satake, S.; Kawashima, H., "Semantic Sensor Network for Physically Grounded Applications," *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on* , vol., no., pp.1-6, 5-8 Dec. 2006
4. Smith M.K., Welty C. and McGuinness D.L. OWL web ontology language guide. W3C recommendation, Feb 2004
5. Brickley, D. and Guha, R.V. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, Feb 2004.
6. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, 21 May 2007.

7. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. W3C Recommendation, 2007.
8. Martin, D. et al.: OWL-S: Semantic markup for web services. W3C Member Submission, 2004.
9. OMG PRR (Production Rule Representation), Beta 1, OMG Adopted Specification, November 2007. http://www.omg.org/spec/PRR/1.0/
10. Motik, B., Horrocks, I. and Sattler, U. Bridging the gap between OWL and relational databases, *Conference on World Wide Web*, 2007.
11. Brunner, J-S., Ma, L., Wang, C., Zhang, L., Wolfson, D. C., Pan, Y., and Srinivas, K. 2007. Explorations in the use of semantic web technologies for product information management. *Conference on World Wide Web*, 2007.
12. Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, OWL 2 Web Ontology Language:Structural Specification and Functional-Style Syntax. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/. Latest version available at http://www.w3.org/TR/owl2-syntax/.
13. Grosof, B., Horrocks, I., Voltz, R. and Decker, S., Description Logic Programs: Combining Logic Programs with Description Logic. WWW, 2003.
14. Forgy, C., Rete: A Fast Algorithm for the Many Pattern/Many Object, 1980
15. Meta Object Facility (MOF) 2.0, OMG Document: formal/2006-01-01, http://www.omg.org/cgibin/doc?formal/2006-01-01
16. I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL Rules: A Proposal and Prototype Implementation, J. Web Semantics 3 (2005) 23-40.