Chapter 2

Business Rules Using OWL and SWRL

Alan Meech[1]

Absract There are now several implementations of SWRL. This paper examines the usefulness of SWRL for business rules in a commercial setting and examines a collection of available rule engines to see how well they support the standard.

1. INTRODUCTION

Rule engines have been available for decades in various forms, mostly in scientific and engineering settings, but have only become accepted in the realm of business enterprise architecture in the last decade. Acceptance of rule engines has been limited due to a number of factors (unfamiliar technology, lack of educational support, and obscure rule languages, to name a few). However with the advent of the Business Rules Manifesto [13], the business analyst and enterprise architecture communities are becoming more open to the use of rules in commercial enterprise systems.

The SWRL [1] rule language, which combines OWL and RuleML, has been an accepted part of the W3C semantic web technology stack for almost five years now. Starting as a W3C member submission, it is now well-documented, is described in major Semantic Web textbooks (including [2] and [3]), it has well-defined syntax and semantics, integrates with the other languages in the family (OWL and RDFS) and has some well-supported implementations available for use (including plug-ins for the Protege ontology editor). While a new language (RIF [4]) may soon cover the same technological areas, SWRL is currently the primary W3C rule language.

This paper attempts to determine whether the features of an OWL/SWRL rule engine would be an improvement over conventional rule engines and whether the existing implementations are complete enough to be of use in enterprise architectures. One of the main issues will be the difference in features between existing rule engine technologies (normally production rule systems) and OWL/SWRL. The issue of relative performance between existing OWL/SWRL engines, and between these and the existing established production rule systems, will be left for a future paper. (Comprehensive work has recently been done in performance testing engines [5, 6], though not specifically SWRL engines.)

1.1 Business Rule Engines

Business Rules are increasingly part of the system specifications (requirements) used to design enterprise information systems. Business rules can be visualized as system requirement statements in the form of "IF…THEN…" sentences (with a segregated condition clause and result) or they can be stated using logical quantifiers (such as "All emergency patient records must have Medicare Numbers"). Along

---

[1] CGI (alan.meech@cgi.com)

with rules, Base Facts are normally included in a business rule engine to state unchanging (constant) knowledge that is needed to execute rules ("Postal Service Air Mail shipments to Iran must be less than 5.7 KG"). The Business Rules are specified in a Rule Language which captures the rules and facts in a human-readable form.

A Business Rule Engine is a component in an enterprise architecture that encapsulates a number of the specified rules for a system in to a single architectural component to perform a logical task, typically a decision or control task. A Business Rule Engine could be used for a task like checking whether a shipment conforms to postal regulations, returning any errors it finds, or determining the triage level of a patient entry form in an emergency room (based on symptoms, for instance).

The term "Business Rule Engine" is used in this paper to refer to the rule engine as an architectural component for decision and control tasks in business systems, while the term "Rule Engine Technology" is used to refer to a specific technology or an instance of a specific technology that performs inference. Business Rule Engines typically will use one or more rule engine technologies in its implementation, though it can be coded in a conventional technology (Java, COBOL or as a stored procedure in a database).

Business Rule Engines frequently require access to Base Facts (reference information, such as timetables, constant values, price lists and customer information, needed to complete the decision task) which is not available in the data provided with the input. Base Facts and rules for a business rule engine are included in one or more Rule Bases. If the information is small and self-contained, it can be hard-coded directly into the rule base. Otherwise, there needs to be a mechanism to allow the business rule engine to access the facts from external sources (web services, databases, the Web).

In enterprise systems, where resources are shared on a massive scale, the typical usage for rule engines is a repeatable one-shot decision service. Given a defined set of rule bases to apply to data, a business rule engine takes one set of input data (a case) at a time, asserts the case data as facts into memory ("working memory"), applies the rules and base facts to perform the decision task and retrieves the results which are returned to the caller. The rule bases are reused and memory is cleared for each new set of case data. (This usage differs from many Knowledge Systems, where results are accumulated and used in future decisions, like a typical database application.) Business rule engines can be designed as embedded sub-systems or as stand-alone services.

1.2 Rule Engine Technologies

Rule engines have existed in various forms since the early 1960's and many different approaches have been used over time. While the approach and capabilities of the various engine technologies is radically different, all implement some kind of inference engine that is used to execute logical statements and make inferences based on them.

Each will also contain some kind of Rule Language used to specify base facts and rules. Many support several languages.

1.2.1 Types of Rule Engine Technologies

Some of the earliest and most widespread systems were "Production Rule Systems" (including expert system shells). These systems typically used a language that described "IF … THEN …" rules, a working memory to hold asserted facts, a forward chaining engine that looked for rules that matched the current facts in the working memory and "fired" the corresponding rule if it matched. New assertions were made when rules fired, and rules would fire until no more rules were found to fire. Examples of these systems include CLIPS (and its Java successor JESS), DROOLS and Oracle's rule engine. Many production rule systems, such as DROOLS 4, are designed to enforce a strict separation of rules and facts, so any base facts needed for inference normally need to be asserted into the working memory each time along with the case facts.

Logic programming languages are often used for rule engines, such as variants of the Prolog language, which is one of the earliest and most widely-used languages in this class. Prolog uses a different approach to inference, but the structure of a Prolog program is very similar to the IF-THEN rules used in production systems. The added advantage is that Prolog is a full-fledged programming language and many Prolog environments have libraries for interacting with databases and other standard parts of a development environment.

The most powerful group of engines that can be used as rule engines are a class called "Automated Theorem Provers". The most common and mature technologies in this group implement various forms of first order logic which is generally versatile enough that subsets of it are used for most of the languages currently in use (that is, the engines need to be "throttled down" in most cases). First order logic is the logic that is normally taught in college symbolic logic courses, so the match between the capabilities of these machines is similar to the capabilities of what is considered "classical" (human, philosophical) logic. The algorithms in these engines vary widely. Examples of these engines include Prover9, Isabelle, CYC and Vampire. For the purposes of this paper, engines like Pellet and HermiT are also included in this group.

1.3 Rule Languages

Facts and rules for a Business Rule System are specified in a Rule Language. Each rule engine technology supports one or more rule languages. Most systems have their own unique "native" language, and given the complexity of the systems, it normally will take a considerable amount of time for developers to learn the language for a system. Also, since there are differences in the logical models and behaviour of engines, it is difficult to do automatic translation between the native engine languages.

Some common languages for use with logical systems have appeared. KIF was a popular common language created in the early 1990's (from which variants like SUO-KIF have been written) and the ISO Common Logic specification was recently accepted. While every common language has to make trade-offs (to reach common ground) they have many advantages. Developers using standard languages can transfer language skills between projects. Rule bases in standard languages can be interchanged and published. In general, having a well-specified common language allows a technology to grow and gain acceptance much more quickly.

At the beginning of the Semantic Web project, the W3C defined a stack of technologies (the so-called "layer-cake" – see http://www.w3.org/2007/03/layerCake.png) that needed to be specified and appointed technical committees to cover each of them. These specifications (available at the Semantic Web Specifications web site – see http://www.w3.org/2001/sw/Specs.html) matured into four main language groups: RDF (data modeling), OWL/RDFS (ontology), SPARQL (query) and SWRL (rules).
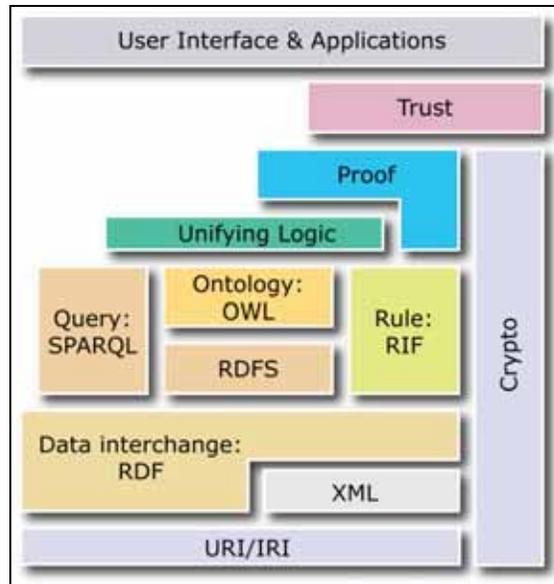
Figure 1 - W3C Semantic Web "Layer Cake"

There are a number of variants that have been developed, but these are the most stable. As seen in Figure 1, work is currently being done on RIF [4], a possible successor to the SWRL language for rules. The following sections describe the features of these languages that are important in business rule engine development. (For primer and tutorial information on these languages, see the references.)

1.3.1 Important Rule Language Features

Existing rule engines such as DROOLS demonstrate a number of the useful features that are needed for business rule engines. To maintain comparable usability, a rule language needs the following features:

- Readability – Rule languages need to be easy to read and understand to allow subject matter experts and analysts to write and review rules in them.
- Support for Data Types - It needs support for handling values in a wide variety of the common computing types, from simple dates, numbers and strings to collections and compound objects (like addresses). Most existing rule languages cover these well, with many using variations of "slots" to represent compound objects.
- Object Type Hierarchies – Much of basic data modelling done in a domain involves class trees and object properties. Most knowledge representation also requires multiple-inheritance of class structures.
- Base Data Integration – All rule systems have the ability to act on data, but it is frequently convenient to define some constant data as part of the rule base. Many older systems strictly separate the rules from the data and do not allow this which is inconvenient for data modelling in rule systems.
- External Data Integration – Enterprise rule systems frequently have to "reach out" to databases and other existing data stores for information needed during decision processes, typically for data that is too large to duplicate in the rule base or for data which changes frequently. Many existing rule engine technologies have mechanisms to do this.
- Built-in Functions – Rules frequently have to match patterns, do calculations and other operations. To avoid cluttering a rule language, a number of basic groups of functions are

normally provided to help rule designers with these tasks, along with a mechanism for defining arbitrary custom types.

- Inference – Naturally, rules have to support various types of inference operations.
- Query – Once inference is done, queries filter the resulting inferred facts and extract them for use outside the rule engine.

1.3.2 The W3C Languages

RDF (the Resource Description Framework [8]) defines the data model for the Semantic Web. It builds on existing XML data types and URL identification standards so it supports a comprehensive set of data types. The RDF standard defines the abstract structure of data in a model (constant data values, objects, object properties and relations between objects), the method of importing external data into a model, the base data types and data type extensions. Since the structure of the data is well-defined, the serializations (file formats and other storage methods) are flexible, and while the RDF/XML format is the reference serialization (normally used by applications for data exchange), there are several currently available which are very human-readable.

RDFS (RDF Schema [9]) was provided to give basic object schema definitions to RDF data. It extends RDF data by defining class hierarchies, property hierarchies and some basic rules of inference. OWL (the Web Ontology Language [10], and the successor, OWL2 [11]) was built on the RDFS foundation, extending it by adding functional and inverse property definitions, classes defined using restrictions and cardinality constraints. OWL and RDFS add the meaning layer to the RDF raw data.

SWRL [1] is the current de-facto standard for rules on the semantic web. The structure of the language is in the form of "horn clauses" and follows the familiar condition/result rule form. SWRL provides a suite of built-in predicates that allow it to do arithmetic operations, string comparisons, regular expression matches and other common operations not available in OWL. SWRL rules can be mingled with RDF facts and OWL class descriptions. This allows facts and rules to be split or combined into logical sets which is a flexible and useful feature when structuring and sharing rule bases. This feature is also very handy when designing rule bases since the constant data and the definitions and rules that use them can be kept close to one another for easy reference.

The RDF, OWL and SWRL definitions do not currently have any specific mechanism for "reaching out" to external data sources. For enterprise business rule systems, this is an issue, since it forces all required data to be converted and brought into the rule engine as RDF or OWL. This leads to a lot of overhead, since it is difficult to know what data will be needed beforehand in many cases.

1.3.3 OWL/SWRL in Rule Systems

Aside from external data access, OWL/SWRL capture all the important features needed for business rules modelling. The fact that the language specifications for the OWL/SWRL combination are mature, well-publicized and have available training materials means that rules developers will be more likely to adopt the languages, technology implementations will be more conformant and it will be easier to find qualified developers for a project. Well-defined languages also lead to the creation of development tools and there are existing development suites for these languages now. All of these factors will be positive considerations for rule projects.

OWL documents are very flexible in the ways they can be joined and shared, allowing many different arrangements of rule bases. Smaller groups of rules and facts can be stored in files of various kinds (local or served from a web server internally), while larger rule bases can be stored in triplestore databases which can take advantage of the triplestore indexing strategies. Some REST web services can also provide data by masquerading as OWL files on web servers, allowing rule bases to import some types of dynamic data.

Given the above, acceptance of the OWL/SWRL standards for corporate applications will depend on two cases:

- Existing Business Rule Engines – For projects that are already in existence, acceptance of a new system and language will be driven by old technologies needing to be replaced (removal of expensive proprietary commercial systems, or replacement of limited legacy rule engines with new and more expressive technologies). It is unlikely that corporate mandates will force acceptance of new rule technologies like SWRL for the sake of standards (the corporate world has not embraced them that closely and will probably not do so for at least 3 years, if at all). Mandates to consolidate rule technologies could happen, but they typically would favour the entrenched systems. For the remaining cases where existing production rule systems or logic language systems are already employed, the expressiveness of OWL/SWRL will not be enough to force rewrites of existing systems because the jump in features is just not compelling enough as a reason.

- New Business Rule Engines – Most projects will fall into this category, since the acceptance of rule engines in the enterprise is a fairly new development. When choosing technologies for new projects, it will be easier to choose OWL/SWRL engines because the preference toward standards (any standards) is already there, the expressiveness and support on the engines can be demonstrated and code is available.

2 CURRENT IMPLEMENTATIONS

Since the current OWL/SWRL combination has enough functionality (as defined in the language specifications) for use in Business Rule Engines, the next question is whether there are usable engines that support these standards and whether the support in these engines is complete enough. The following sections will examine a group of engines that claim to support OWL and SWRL.

2.1 Compliance Testing

One important issue is just how much the engines support the standards. The Semantic Web is currently at a level of development that SQL had in the mid-nineties. Many of the engines are not currently supporting the full standard, and since the usefulness for rules work depends on more or less full feature support, it is important to have a metric for this.

2.1.1 OWL Compliance
OWL (notably OWL 2) supports an active web-based compliance suite and publishes test results for various engines. The suite is comprehensive and new test cases are being added monthly by the community. All published results are shown on the Test Suite Status page on the OWL WIKI (see the summary at http://www.w3.org/2007/OWL/wiki/Test_Suite_Status). The test suite is broken into groups based on status and profile. Profiles are a new feature in OWL 2, but the subtle differences between them may be of little consequence to rules developers initially. For engine testing, the important ones will be the "Approved" OWL DL test cases (many of the tests are repeated in the other categories, so this is a good sampling).
For the sake of this comparison, entries that are incomplete in the OWL tests will be counted as failures. This means that the numbers should be revisited regularly, but given the constant change in the implementations, this would be necessary anyway.

2.1.2 SWRL Compliance
SWRL has had less community support than OWL over the last five years and has not developed a similar standard compliance suite. In the absence of any existing suite, a new suite has been created as part of the investigation of this paper. The suite has been released to help engine developers test and measure

their SWRL support. It is available on SemWebCentral (see http://projects.semwebcentral.org/projects/swrl-test-suite/).

The suite contains 38 tests currently. These are packaged as 13 SWRL files which contain rules designed to produce results if specific features are active. The first set of files test several types of basic operations:

- Class predicate matching and assertions
- Object property conditions
- Basic data type property comparisons

The remainder of the tests verify the support for several groups of the built-in routines required in the SWRL specification (notably numerical comparisons and string operators). These have proven to be critical functions in many applications.

The SWRL test cases are done in a number of different OWL dialects (RDF/XML, N3 and Turtle, currently).

This test suite is still under development, but it can be used to provide compliance metrics for the available engines in this paper.

2.2 Engine Comparison

The engines in this comparison are some of the current OWL engines that support SWRL. The list is not comprehensive and there are new OWL engines appearing every few months, however it is proving hard to find engines that also support SWRL.

- **Pellet** (http://clarkparsia.com/pellet) – This engine is several years old and has an active community supported through its mailing list, as well as commercial support through Clark&Parsia LLC. Pellet has been recognized being a full and complete OWL implementation. Pellet also has solutions for large triplestore support in PostgreSQL and Oracle relational databases.
- **HermiT** (http://hermit-reasoner.com/) – The first versions of this engine were released earlier this year, and the first SWRL-enabled version was released over the summer. The team behind this engine includes expertise from KAON2 and FaCT++, among others and its performance so far has been very promising.

Initially, some other engines were included, but proved to have minimal or no support for SWRL, so these were removed from the list. The following table gives a side-by-side comparison of these engines that did support both OWL and SWRL.

Table 1. SWRL Engine Comparison

Engine Name	Pellet	HermiT
Version	2.0.0-RC7	1.0 (Oct 23, 2009)
OWL Compliance: tests run	347	347
tests passed	346	347
SWRL Compliance: tests run	38	38
tests passed	36	11

The current version HermiT scored high in OWL 2 support, which was one of the main focus areas of development for the 1.0 release. SWRL work was just starting at that point and it should be noted that the majority of the tests that failed were due to the fact that these tests were built-in tests, which had not yet been added to the version tested. Built-ins were to be one of the next areas of development and should be available shortly.

The summary is that today, Pellet has the most complete support for both OWL and SWRL languages and is the best all-around engine for general rules use. These tests will be revisited in a future paper.

3 CONCLUSIONS

More than five years have passed since the first OWL and SWRL specs were released, and there are finally usable engines with fairly complete implementations of the languages and good support bases. The OWL and SWRL combination has the expressive power to be used for Business Rule Engines. For most purposes, this gives a powerful alternative to existing rule engine systems.

However, current implementations will be limited in some enterprise rule bases due to their lack of a mechanism for directly accessing external data sources (relational databases and web services) during inference.

4 FUTURE WORK

There are a number of factors that need to be investigated:

- Performance statistics and a standard OWL/SWRL performance benchmark are needed. This also needs to compare the performance of the OWL/SWRL engine technologies against some existing rule technologies (DROOLS, Oracle Rules) for similar rule bases. (As a reference, the RuleBench test suite [5, 6] and its techniques could be used as a baseline.)
- Performance statistics need to be extended to test the performance of triplestores under large volumes of data. Again, some standard benchmark test is needed.
- Recent work has begun to provide mechanisms for retrieving relational data and converting it to RDF. The W3C RDB2RDF Incubator Group produced a report on the current industry state [7] has applied to the W3C to become listed as a working group. These efforts may solve the issue of external data access to relational databases.
- While the focus of this paper has been OWL/SWRL, rules work is continuing at the W3C in the creation of the RIF languages (such as RIF/BLD [4] and RIF/PRD). This set of standards starts life with a more comprehensive technical specification, an existing set of conformance tests and fairly strong community support. Given that the rule structures are similar to SWRL, it is likely that engines that support SWRL now could easily adopt the RIF standards as well. It would be useful for a comparison to be done between the languages and a comparison of available implementations (like the current paper).

REFERENCES

1. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M. [2004] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* http://www.w3.org/Submission/SWRL/
2. Antoniou, G. and van Harmelen, F. [2008] *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems)*. The MIT Press.
3. Hebeler, J., Fisher, M., Blace, R., Perez-Lopez, A., Dean, M. [2008] *Semantic Web Programming*. Wiley.

4. Boley, H. and Kifer, M. [2009] *RIF Basic Logic Dialect*. http://www.w3.org/2005/rules/wiki/BLD

5. Liang, S., Fodor, P., Wan, H. Kifer, M. [2009] *OpenRuleBench: An Analysis of the Performance of Rule Engines*. http://semwebcentral.org/docman/view.php/158/70/paper.pdf

6. Liang, S., Fodor, P., Wan, H. Kifer, M. [2009] *OpenRuleBench: Detailed Report*. http://semwebcentral.org/docman/ view.php/158/69/report.pdf

7. Sahoo, S., Halb, W., Hellmann, S., Idehen, K., Thibodeau, T., Auer, S., Sequeda, J., Ezzat, A. [2009] *A Survey of Current Approaches for Mapping of Relational Databases to RDF*. http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf

8. Manola, F. (ed.) and Miller, E. (ed.). [2004] *RDF Primer*. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

9. Brickley, D. (ed.) and Guha, R. (ed.). [2004] *RDF Vocabulary Description Language 1.0: RDF Schema*. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

10. Dean, M. (ed.), Schreiber, G. (ed.), Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L. [2004] *OWL Web Ontology Language Reference*. http://www.w3.org/TR/2004/REC-owl-ref-20040210/

11. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. Rudolph, S. [2007] *OWL 2 Web Ontology Language*. http://www.w3.org/2007/OWL/wiki/Primer

12. Prud'hommeaux, E., Seaborne, A. [2008] *SPARQL Query Language for RDF*. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

13. Ross, R. (ed.) and the Business Rules Group. [2003] *Business Rules Manifesto: The Principles of Rule Independence*. http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf