TRIPLE - an RDF Rule Language with Context and Use Cases

Stefan Decker
DERI
NUIG, Ireland
stefan.decker@deri.org

Michael Sintek
DFKI
Kaiserslautern, Germany
sintek@dfki.uni-kl.de

Andreas Billig
Fraunhofer ISST, Berlin, Germany
andreas.billig@isst.fraunhofer.de

Nicola Henze
Peter Dolog
Wolfgang Nejdl
University of Hannover, Germany
henze@kbs.uni-hannover.de

Andreas Harth
DERI
NUIG, Ireland
Andreas.Harth@deri.org

Andreas Leicher
Susanne Busse
Jörn Guy Süß
Technische Universität Berlin, Germany
aleicher@cs.tu-berlin.de

Zoltán Miklós
Technical University of Vienna, Austria
z.miklos@infosys.tuwien.ac.at

Jose-Luis Ambite
Matthew Weathers
ISI/USC, USA
ambite@isi.edu

Gustaf Neumann
Uwe Zdun
Vienna University of Economics, Austria

Abstract

TRIPLE was designed as a practical rule language for data manipulation applications.

Over the last couple of years the language has been deployed in various applications and use case studies. In this paper we first introduce the design principles of TRIPLE and then present some of the applications for which this language has been used.

1. Introduction

Not surprisingly, everyone perceives the Semantic Web in a different way. One view is that the Semantic Web is about semantics, and semantics is about AI-style knowledge representation, which leads to knowledge representation languages like OWL.

Another view is that the Semantic Web is about overcoming the syntax of data so that users and developers can concentrate on the semantics of information. Following this view means that languages and tools for the Semantic Web must focus on practical problems rather than generic KR tasks. That is, they should make it easier and cheaper to publish, understand, use, and reuse data and services on the Web in an interoperable and scalable way. Languages that help define how different data sets and vocabularies relate to each other are necessary; they are able to provide the glue between (distributed) information systems and data sets. Following this view also has consequences for designing rule languages for the Semantic Web.

A major task to achieve on the Semantic Web is to provide tools that drive down the cost of establishing interoperability between different data providers. A rule language can help here: writing rules is usually faster and cheaper than writing program code since a rule language has more declarative features and is usually not burdened with the details of a general programming language. Rules provide benefits over a software product's life cycle; they are simpler to write than code, more concise, and easier to understand, to share, and to maintain.

Standardizing such a rule language has several benefits. Interested parties can invest in building the infrastructure because a market is being created. Standardization also enables competition to drive innovation. And last but not least, it allows rule sharing (that is, knowledge about how to achieve interoperability).

In other words, a rule language for the Semantic Web may be seen as a data transformation and glue language - in contrast to a knowledge representation language, which captures knowledge about a certain domain.

Of course such a rule language needs a defined semantics (as a basis for implementation) and efficient evaluation mechanisms.

Starting from modest beginnings (SiLRI) reported in [6], with TRIPLE [3][4] we were aiming at a practical language suitable for applications. In the following sections we outline the requirements for TRIPLE and present some existing applications and use cases of TRIPLE.

2. Design Rationals for TRIPLE

A rule language for the Semantic Web has to support RDF querying, derivation, transformation, and generation. Support for RDF implies more requirements, such as support for namespaces, reification and blank nodes.

Another requirement for TRIPLE was support for contexts:

In the Semantic Web many different communities are publishing their data. Although the data might be identical it is often the *context* of the data that determines its true meaning or value. E.g., if the White House publishes a set of facts about the President of the United States, we may (or may not) treat this data as more trustworthy as if the same data had been published by some other source.

The conclusion is that RDF data needs to be judged with its respective context. The *source* is only one of the most obvious contexts a piece of data can have. In general, many more forms of context can be identified: e.g., time, location (or a combination of both), communities etc.[13].

Pushing the idea of context further, one possible context for a piece of RDF data is its intended semantics. E.g., RDF data can be interpreted with a specfic semantics in mind - e.g., OWL, UML, TopicMaps, RDF Schema, Entity Relationship Models, DAML+OIL, and more, highly specialized data models. RDF data may be queried with or without a certain semantics in mind. Uniting two different datasets might require two different semantics being applied (e.g., think of a mixed dataset containing TopicMaps and OWL ontologies).

Pushing the idea of context even further, we may even allow parameterized contexts - e.g., certain sets of RDF data may be dependent on a parameter, which then determines the real content of the RDF data.

Parameterized contexts for RDF data can be used for applying a set of rules to a set of RDF data. The use of contexts as parameters enables one to define the semantics of a particular data modelling language (e.g., RDF Schema) as a derived context over some source data, which is passed as a parameter.

3. Use Cases, Applications and Extensions of TRIPLE

In the last couple of years the TRIPLE system has been used in a variety of applications. In the following sections we report on some of them.

3.1. Querying Semantic Web Resources Using TRIPLE Views

Resources on the Semantic Web are described by metadata based on some formal or informal ontology. It is a common situation that casual users are not familiar with a domain ontology in detail. This makes it difficult for such users (or their user tools) to formulate queries to find the relevant resources. Users consider the resources in their specific context, so the most straightforward solution is to formulate queries in an ontology that corresponds to a user-specific view. We developed an approach based on multiple views expressed in ontologies simpler than the domain ontology. This allows users to query heterogeneous data repositories in terms of multiple, relatively simple, view ontologies. Ontology developers can define such view ontologies and the corresponding mapping rules. The use of the language TRIPLE turned out to be very useful here: we could use it not only to represent RDF data but also to define views and mappings at the same time. TRIPLE provided us an infrastructure to evaluate queries issued against the user-specific ontology and in this way to realize the view. The view techniques have central importance when creating interoperability. More details can be found in [1] .

3.2. Conflict Analysis and Model Transformation based on TRIPLE

Component composition is an important objective of software engineering. It promises component reuse and therefore a productivity gain because of shorter time-to-market and improved quality. Unfortunately, composition is diffcult to achieve in practical terms because of technological incompatibilities and diverging component specifications.

The TU Berlin implemented a rule-based framework to identify conflicts that impede component composition. The framewok uses the concepts of Architecture Description Languages (ADLs) and captures structural (type) and behavioral (protocol) conflicts as well as conflicts between communication properties of components [11].

The framework for component conflict analysis consists of several RDF schemata as well as of TRIPLE rules for component transformation and conflict identification. It is implemented on top of the Ontology-Based Domain Repository (ODIS) and extends ODIS by adding several specialized tools for component conflict identification. At present it uses Haskell to check for type conflicts, Aldebaran and fc2tools to check for simulation and bisimulation conflicts, and LTSA to identify deadlocks.

Conflict identification is directly integrated with the software development process. Therefore, components are defined in UML according to a UML profile for software components. Because of the 'context construct' of TRIPLE, this also enables model transformation as proposed by the OMG in their MDA.

The transformation of generic UML components into platform specific components is realized in two steps: In a first step a generic UML-to-RDF transformation is applied, which results in the representation of UML models as RDF instances of the UML metamodel. This is a generic transformation of a UML model into a RDF representation.
The second step converts the RDF model into a specified target representation using TRIPLE rules. The result is a platform specific model of a particular technology such as CORBA or .Net.

In summary, using TRIPLE as a transformation language it becomes possible to reuse a conceptual model for different technologies. One can create an EJB model or a CORBA model or a model for another language from a single conceptual model. Based on properties in a feature model, model transformation can also be parameterized in a more flexible way. In [12], we demonstrated that model transformations can be parameterized based on J2EE patterns.

3.3. Ontology Management with TRIPLE

Organizations increasingly recognize the necessity to create and manage project entities like (meta-) models, document metadata, and classification ontologies in such a manner that the integrative usage of these entities (called *domain artefacts*) is supported. Figure 1 shows an example from the automotive domain with various artefact groups. A repository prototype for managing, processing, and querying such domain artefacts has to focus on the following requirements:

1. *Partitions*: the repository language has to support the partitioning of the whole artefact set into different units according to the artefact groups in figure 1.
2. *Derivations*. The system has to offer the functionality to derive new information on the base of logical rules.
3. *Transformations*: the repository language has to support transformation between the elements of the artefact groups.

TRIPLE [4] adequately fulfils the above requirements because TRIPLE is constructed over RDF which allows for the management of concepts (like elements of thesauri and glossaries), classes, instances, and their interrelations as first class citizens. One of the main enhancements of TRIPLE is the explicit representation of the *context* of statements which extends statements by a fourth dimension to quadruples (cf. [10]) and thereby supports requirement 1. TRIPLE enables the user to define logical rules over RDF graphs according requirement 2. Moreover, the feature of TRIPLE to state nearly full predicate logic in the antecedent of an inplication increases its expressiveness. The most important enhancement is the concept of *parameterized contexts*. In particular transformations between the elements of the artefact groups from figure 1 are supported in a highly concise way which fulfils requirement 3.

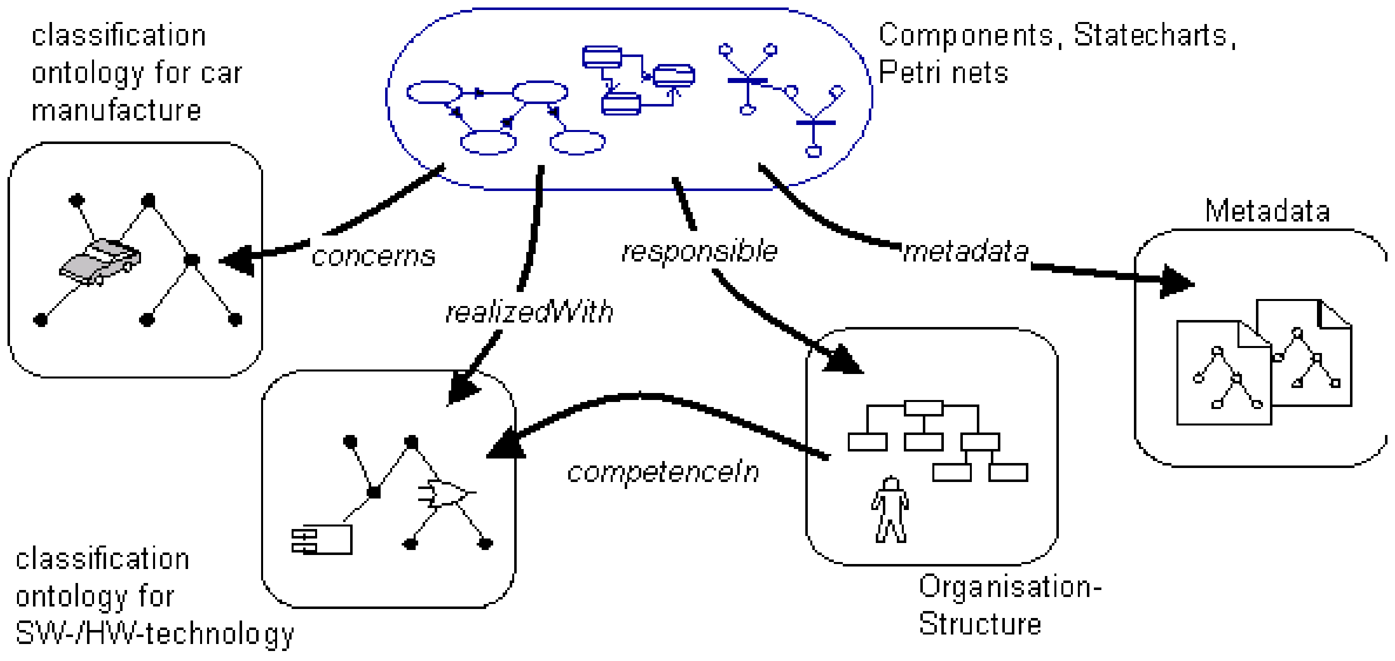
classification ontology for car manufacture
Components, Statecharts, Petri nets
Metadata
concerns
realizedWith
responsible
metadata
competenceIn
classification ontology for SW-/HW-technology
Organisation-Structure

Figure 1

To gain experiences in managing, processing, and querying domain artefacts we developed the repository prototype ODIS (**O**ntology-based **D**omai**n** Repo**s**itory) which is built around TRIPLE. Furthermore, ODIS introduces some syntactical and symantical extensions of TRIPLE related to the requirements above. To support requirement 1 ODIS introduces the separation of the rule base and the fact base inspired by the distinction between intensional resp. extensional predicates from deductive databases. More precisely, for statements we introduce a two-valued sixth dimension. Figure 2 shows the intended architecture wrt. this construct. Place 1 in this figure shows the classical usage of contexts to modularize the whole rule set. Places 2 and 3 show the usage of the '@@'-construct which express the referencing of statements only within the fact base.

Beside this 'semantical' extension ODIS offers two syntactical extensions of TRIPLE to simplify tasks related to requirement 2 and 3. First of all we reintroduce the 'multi-value'-construct from F-Logic on instance level which allows for expressions like *s [p->{o1, o2}]*. Secondly, ODIS enables implications in the postcedent of an implication, i.e. formulas of the form *((a <- b) <- c)*. To realize the mapping to horn logic we apply prolog-based transformation rules to TRIPLE terms.This kind of expression is extremely useful for model transformations because it successively constructs the parts of the source model which have to be transformed. In other words *((a <- b) <- c)* can be read as "find the pattern *c* in the source model. If the pattern *b* is found (using bindings of *c*) then produce the target model *a*".

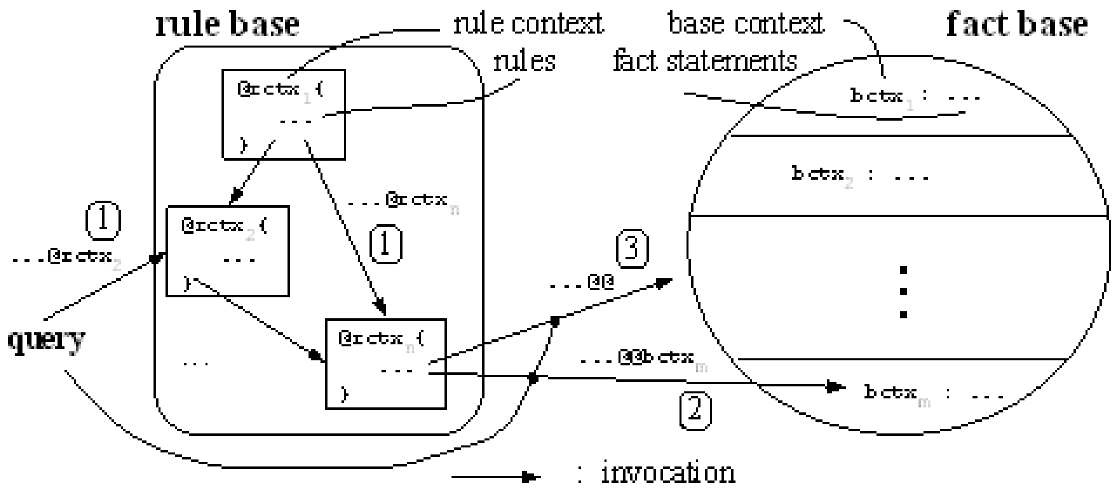

Figure 2

3.4. Towards Personalization Services for the Semantic Web with TRIPLE

The Personal Reader framework implements a service-based architecture for developing and maintaining personalization functionalities on the Semantic Web. A modular framework of services - for providing the user interface, for mediating between user requests and available personalization services, for user modeling, for providing personal recommendations and context information, et cetera, is the core of the Personal Reader framework. When a user is viewing some Web Content (the "Reader" part of the Personal Reader) s/he receives additional, personal information on the context of this particular Web content (the "Personal" part of the Personal Reader). The first prototypes developed with the Personal Reader framework project are developed in the area of e-Learning [7][8] (Java, Semantic Web), and for browsing scientific publications [9]. Personal Readers for e-Learning: The Personal Readers for e-Learning provide a learner with a personal interface for regarding learning resources: the Personal Annotation Service recommends the learners next learning steps to take, points to examples, summary pages, more detailed information, etc., and always recommends the most appropriate of these information according to the learner's current knowledge, his/her learning style, learning goal, background, etc. The necessary reasoning for the personalization services is realized using TRIPLE. Reasoning rules for personalization, user modeling, and for interpreting the actual user request at runtime are realized as separate TRIPLE models, which are combined as necessary. RDF-descriptions of Web resources and associated domain ontologies are converted into TRIPLE and provide the necessary runtime input data for the reasoning rules. Prototypes (for learning the Java programming language, or for learning about the Semantic Web) can be accessed via http://www.personal-reader.de.

3.5 Ontology based matchmaking on the Grid with TRIPLE

Grid is an emerging technology for enabling resource sharing and coordinated problem solving in dynamic multi-institutional virtual organizations. In the Grid environment, shared resources and users typically span different organizations. The resource matching problem in this environment involves assigning resources to tasks in order to satisfy task requirements and resource policies. These requirements and policies are often expressed in disjoint application and resource models, forcing a resource selector to perform semantic matching between the two. Grids are used to join various geographically distributed computational and data resources, and deliver these resources to heterogeneous user communities. These resources may belong to different institutions, have different usage policies and pose different requirements on acceptable requests. Grid applications, at the same time, may have different constraints that can only be satisfied by certain types of resources with specific capabilities. Before resources can be allocated to run an application, a user or agent must select resources appropriate to the requirements of the application. We call this process of selecting resources based on application requirements 'resource matching'. In a dynamic Grid environment it is necessary to automate the resource matching.

We have designed and prototyped our matchmaker using TRIPLE [2][5] to use ontologies encoded in W3C's Resource Description Format (RDF) and rules (based on Horn logic and FLogic) for resource matching. Resource descriptions, request descriptions, and usage policies are all independently modeled and syntactically and semantically described using RDF Schema. Finally, we utilize inference rules for reasoning about the characteristics of a request, available resources, and usage policies to appropriately find a resource that satisfies the request requirements. The thight integration of RDF with TRIPLE together with the context mechanisms made it convenient to handle multiple ontologies and mediate between them.

3.6 Other Projects involving TRIPLE

Other projects involving TRIPLE are:

- SmartWeb: Mobile Broadband Access to the Semantic Web. Recent progress in mobile broadband communication and Semantic Seb technology is enabling innovative internet services that provide advanced personalization and localization features. The goal of the SmartWeb project (funded by the German government; duration: 2004 - 2007) is to lay the foundations for multimodal user interfaces to distributed and composable Semantic Web services on mobile devices. TRIPLE is beeing used for accessing an RDF/OWL-based knowledge server. See http://www.smartweb-project.de/.
- ARGOS [14] develops a flexible data query and analysis system based on the web services paradigm. As an application domain ARGOS examines several goods movement planning problems and their effects on urban structure. TRIPLE has been used to model the data sources and operations uniformly as web services that produce or transform measurements, and to describe the semantics of these web services, e.g., by using TRIPLE logic rules built with terms from the ontology to accurately specify the inputs and outputs of sources and operations.

4. Conclusions

We have developed a rule language for RDF which has been used and extended in many applications. Especially the notion of context has been crucial for these application, and a future rule standard should support the TRIPLE notion of context. We also believe that the use cases presented here provide valuable input for a standardization effort.

5. References

[1] Zoltán Miklós, Gustaf Neumann, Uwe Zdun, Michael Sintek: *Querying Semantic Web Resources Using TRIPLE Views*, Second International Semantic Web Conference (ISWC), Sanibel Island, Florida, USA, 2003.

[2] Hongsuda Tangmunarunkit, Stefan Decker, Carl Kesselman: *Ontology-Based Resource Matching in the Grid - The Grid Meets the Semantic Web*, Second International Semantic Web Conference (ISWC), Sanibel Island, Florida, USA, 2003.

[3] Michael Sintek, Stefan Decker: *Using TRIPLE for business agents on the Semantic Web.*, Electronic Commerce Research and Applications 2(4): 315-322 (2003).

[4] Michael Sintek, Stefan Decker: *TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web.*, First International Semantic Web Conference 2002: 364-378.

[5] Andreas Harth, Stefan Decker, Yu He, Hongsuda Tangmunarunkit, Carl Kesselman: *A semantic matchmaker service on the grid*, World Wide Web Conference 2004: 326-327.

[6] Stefan Decker, Dan Brickley, Janne Saarela, Jürgen Angele: *A Query and Inference Service for RDF,* Query Languages Workshop 1998 (QL'98), W3C Workshop.

[7] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek: *The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies*, International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Aug.2004.

[8] Nicola Henze and Matthias Kriesell: *Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementations*, First International Workshop on Engineering the Adaptive Web (EAW 2004), Eindhoven, The Netherlands, Aug. 2004.

[9] Robert Baumgartner, Nicola Henze, and Marcus Herzog: *The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web*, European Semantic Web Conference 2005, Heraklion, Greece.

[10] Andreas Harth and Stefan Decker: : *Yet Another RDF Store: Perfect Index Structures for Storing Semantic Web Data With Context*, DERI Technical Report, 2004.

[11] Andreas Leicher and Jörn Guy Süß: *Augmenting UML Models for Composition Conflict Analysis*, Middleware 04, ACM/ IFIP/ USENIX International Middleware Conference (2004), H.-A. Jacobsen, Ed., no. 3231 in LNCS.

[12] Andreas Billig, S. Busse, Andreas Leicher and Jörn Guy Süß: *Platform independent model transformation based on TRIPLE*, Middleware 04, ACM/IFIP/ USENIX International Middleware Conference (2004), H.-A. Jacobsen, Ed., no. 3231 in LNCS.

[13] Robert M. MacGregor, In-Young Ko: *Platform independent model transformation based on TRIPLE*, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems, Sanibel Island, Florida, USA, October 20, 2003. CEUR Workshop Proceedings 89.

[14] Jose Luis Ambite1, Genevieve Giuliano, Peter Gordon, Andreas Harth, LanLan Wang, Qisheng Pan , and Stefan Decker: *Argos: Dynamic Composition of Web Services for Goods Movement, Analysis and Planning.*, http://dgrc.org/dgo2004/disc/presentations/transportation/ambite.pdf