

# *MoRe* Semantic Web Applications

Maksym Korotkiy<sup>α</sup> and Jan L. Top<sup>α,β</sup>

<sup>α</sup>Vrije Universiteit Amsterdam, Department of Computer Science  
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

<sup>β</sup>Wageningen Centre for Food Sciences  
P.O. Box 557 6700 AN Wageningen, The Netherlands  
maksym@few.vu.nl jltop@few.vu.nl

**Abstract.** We present *MoRe* – a framework that allows one to extend a domain ontology with a remotely invocable reasoning service applicable to concepts defined in that ontology. Our approach bridges the gap between ontology and application developers. We allow any reasoning service to be wrapped by a *MoRe* ontology extension, the services ranging from generic logics-based reasoners to specific black box software components. The application developer directly accesses these reasoning services through documents stated in terms of domain concepts rather than dealing with remote procedure calls. We describe a case that applies *MoRe* to an OWL-ontology of units of measure, and we demonstrate how this extended ontology can be integrated into a unit conversion application.

## 1 Introduction

Ultimately, the Semantic Web [1] aims to significantly improve the experience of web application end-users. To achieve this, the Semantic Web is to provide an environment that enables an application developer to advance web applications beyond what we observe nowadays. Ontologies are the keystones of the Semantic Web, and ontology developers play a crucial role in developing that enabling environment.

It is believed that the current approaches to the Semantic Web make it rather difficult to develop applications needed so much to materialize the Semantic Web vision [2, 3]. A number of solutions to facilitate design of Semantic Web applications has been proposed, ranging from authoring [4], browsing and annotation frameworks [5, 6] to infrastructures for the Semantic Web Services [7]. These approaches address specific aspects of the Semantic Web application development. In *MoRe* we take a step back to see how ontologies can be extended to make them more attractive for application development in general.

Presently, application developers do not profit much from the increasing availability of domain ontologies. The latter are typically devised for representation of static domain knowledge, whereas applications require problem-specific answers and computations. Generic reasoners and query languages associated with formalisms like RDFS or OWL are often not expressive, efficient and transparent enough to be used in applications. We propose *MoRe*<sup>1</sup> – a simple approach to extend ontologies with application-oriented, but

<sup>1</sup> “MoRe” used to be an acronym but with the development of our approach its original interpretation has become irrelevant.

still generic concepts and reasoning services. This provides for solutions, arbitrarily on the continuous scale between domain- specific ontologies (using generic reasoners) to task-specific applications (using dedicated procedures). The objective of our approach is to simplify application development by means of increased (re)usability of ontologies.

To improve the usability of ontologies *MoRe* extends them by providing an elementary mechanism for attaching a reasoning service to these ontologies. We believe that the availability of a readily accessible reasoning service allows the application developer to faster evaluate an ontology and to incorporate it more readily into an application.

Our approach can be illustrated by providing an example from e-Science, our field of application. e-Science aims at providing automated support to researchers in performing experiments and constructing models and theories. A simple but very important quality requirement for scientific work is correct and consistent use of units of measure. Traditionally, an application developer would construct a specific algorithm for unit conversion, using an internal database of units and their values in terms of reference units. With the availability of a units ontology, the application developer could instead access this ontology to determine for example the conversion factor between two units. In this case, he would profit from the shared knowledge provided by the ontology, but he would still have to write specific code to employ this knowledge in the application.

In *MoRe* we extend the units ontology with an associated reasoning service. For unit conversion, we define an additional but still generic concept `ConversionExpression`. In this case, the application developer only needs to specify a document with the following content (simplified):

```
UnitsOntology
  ConversionExpression
    sourceUnit:      inch
    destinationUnit: yard
```

The appropriate middleware detects the ontology underlying the document, locates it on the Web and applies the associated reasoning service to this input document. After that the middleware sends the following document back to the application:

```
UnitsOntology
  ConversionExpression
    sourceUnit:      inch
    destinationUnit: yard
    factor:          0.02777778
```

The resulting document contains a new fact (value of the `factor` property) allowing the developer to convert inches to yards.

This example provides a simple illustration of the application of our approach. The main motivation behind *MoRe* is to make ontologies more (re-)usable to application developers. Moreover, *MoRe* can help to overcome the lack of expressiveness, efficiency and transparency of present ontology languages (such as OWL) and associated generic reasoners. We emphasize that we do not claim to replace or improve existing



```

computeConversionFactor (srcUnit, dstUnit, factor)

    srcFactor=db.query('
        SELECT SI_unit_factor
        FROM ...
        WHERE unit=srcUnit')
    ).get('SI_unit_factor')

    dstFactor=db.query('...WHERE unit=dstUnit').get(...)

    factor= srcFactor/dstFactor

```

**Fig. 1.** Pseudocode of a traditional DB-based approach. Using general purpose ontology middleware leads to a similar solution.

```

computeConversionFactor (srcUnit, dstUnit, factor)

    reqDoc=
        'MyConversionExpression
          type          ConversionExpression
          hasSourceUnit srcUnit
          hasDestUnit  srcUnit'

    resDoc=MoRe.process(reqDoc)

    factor = resDoc.getProperty('hasConversionFactor')

```

**Fig. 2.** Pseudocode of a *MoRe*-based approach.

The second way to exploit an ontology in an application is to employ general purpose ontology middleware, such as Jena [8] or Sesame [9], that provides storage, reasoning and query facilities for ontologies.

However, in our scenario the reasoning capabilities of the associated ontology languages do not allow us to compute the conversion factors in a feasible way (we elaborate on this in Section 4). As a consequence, the second approach would be very similar to the previous one, only the queries would be expressed in a different language and applied not to a database but to a stored ontology.

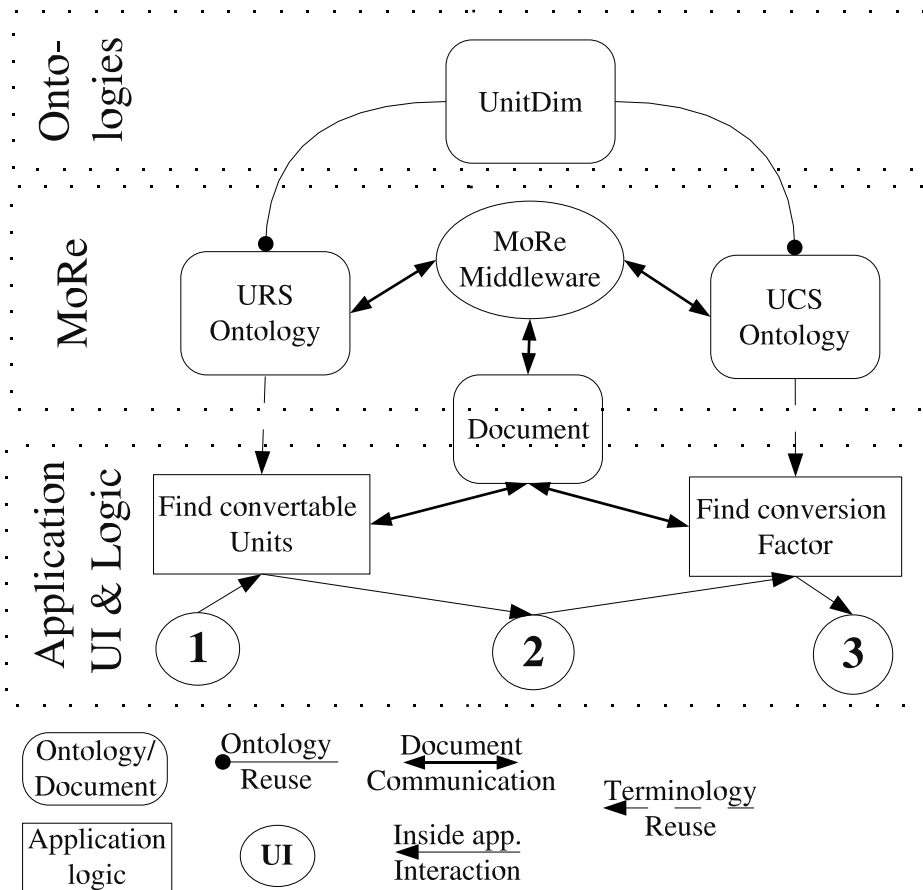
In both cases, the application developer has to incorporate part of the domain knowledge into the application. Nevertheless, it is natural to expect that the way a conversion factor is computed is part of the units of measure domain. *MoRe* allows an ontology developer to incorporate such domain knowledge as a domain-specific reasoning procedure connected to concepts from the units of measure domain. If an application developer would have such a *MoRe*-ontology to his disposal, the pseudocode depicted in Fig. 2 could be used.











**Fig. 4.** Main steps of the Unit Converter application









