## From Ontology-enabled Services to Service-enabled Ontologies

Korotkiy, M.

2009

***document version***
Publisher's PDF, also known as Version of record

# Summary

In this thesis we aim at improving usability and reusability of knowledge by combining ontologies and Service-Oriented Architectures. To achieve this we propose the Onto⇔SOA framework that describes how ontologies and services can be designed in a way that enables their *natural integration*. We maintain a consistent, simple and pragmatic approach that can be deployed without significant investment of effort. The design guidelines underlying Onto⇔SOA are a combination of well-established practices from the Software Engineering and Service-Oriented Architectures fields. Our approach complements state of the art research in the fields of Ontology Engineering, Semantic Web, Semantic Web Services, Service-Oriented Architectures and Software Engineering.

We apply Onto⇔SOA to a number of use cases from the e-Science domain – our target application domain. These use cases are: retrieving documents matching a given query (Chapters 2 and 3), conversion (Chapter 5) and consistency checking of units of measurement (Chapter 4). In these use cases we employ the guidelines proposed in Onto⇔SOA to design and implement ontology-based and document-oriented services that facilitate the above-mentioned e-Science tasks.

Onto⇔SOA provides a uniform ontology- and service-oriented framework for the solutions to the target cases. In Chapter 2 we introduce Onto⇔SOA as an architectural framework for Ontology-Enabled services. This framework is based on a restricted service model that constrains internal properties of a service to induce *domain alignment* and *loose coupling* characteristics. These constraints allow, among others, to simplify the model of a service and to provide guidelines on the design of ontology-enabled services.

Onto⇔SOA relies on direct exchange of ontology-based messages between a *document-oriented* service and its consumer; and employs an ontology as a *service schema* (referred to as a *service ontology*). The main purpose of ontologies is to transfer domain conceptualization to services (Service-Oriented Architectures), thus enhancing their domain (business) alignment.

In addition to the addressed e-Science use cases, in Chapter 6 we investigate the more general problem of linking relational and ontology-friendly data models. We propose the FDR2 approach that relies on three components: an automatically generated RDFS schema of tabular data, an RDF serialization of the data itself, and a manually created map. The proposed approach is purely RDF/S-based and does not require any additional software components except an RDFS reasoner. Moreover, the technique underlying FDR2 is general enough to be applicable to data sources different from relational ones. By explicating the schema of the original data, serializing the data according to that schema and linking the schema to a target ontology we can semantically enrich the data and improve its accessibility by ontology-enabled software.

To facilitate the evaluation and application of Onto⇔SOA to the targeted use cases, we further specialize it into *MoRe* (Chapter 5) – an operational framework and corresponding middleware based on RDF/S languages and REST Services. *MoRe* aims to provide a simple and pragmatic foundation for the development of ontology-based Web applications. We believe that it also facilitates bridging the gap between ontological domain knowledge and software development. On the one hand, it provides a pragmatic application-driven view on domain ontologies. On the other hand, it facilitates software development by integrating domain-specific inference services into software solutions.

Collaboration between services enables effective construction of complex functionality from simpler services and, thus is an important component of any service-oriented approach. In Chapter 4 we integrate a Blackboard-style composition into the Onto⇔SOA approach. The devised composition mechanism utilizes an application-independent controller and a homogeneously structured ontology-based repository (a blackboard). The proposed approach requires neither an extensive service model nor an explicit workflow specification and enables composite functionality to emerge by bringing a number of ontology-enabled services together.

By applying ontologies and services to the e-Science domain we have gained insights on how ontologies and services could benefit from each other. In Chapter 3 we introduce Service-enabled Ontologies – an ontology-oriented perspective on Onto⇔SOA that shifts the focus to service ontologies. Service-enabled Ontologies re-interpret Onto⇔SOA as a mechanism that allows to attach an arbitrary service to an ontology, thus capturing *application semantics* of domain concepts.

These service attachments can facilitate practical application of ontologies, potentially at the costs of their overall reusability. The trade-off between declarative (general, reusable

but difficult to utilize in practice), and procedural (application-specific but easy to exploit) knowledge still holds in Service-enabled Ontologies. Finding a suitable balance between these two ways of representing knowledge ultimately depends on the requirements of a certain application scenario. With Service-enabled Ontologies we provide a novel framework that contributes to flexibility in specifying a domain conceptualization.

In this dissertation we also investigate the overall impact of ontologies on software engineering practice. In Chapter 7 we analyze the effect of ontologies on existing models for estimating software quality (Quint2) and development effort (WEBMO). We estimate that ontology can improve many quality dimensions by supplying domain conceptualization. This conceptualization can be employed across all development stages amplifying the cumulative effect of ontologies. We also optimistically estimate that in the long run, ontologies can cause a significant decrease of development effort. This becomes possible due to the ability of ontologies to facilitate transfer of application domain knowledge into a development team, to provide a unified conceptual view improving communication within the development team as well as to external world, to improve reusability of design artifacts, and, ultimately, to improve efficiency and effectiveness of development techniques.

The reported estimation is based on theoretical analysis and our experience in employing ontologies in software and service engineering. Empirical validation is problematic to carry out due to lack of data on ontology-enabled software development projects. Nevertheless, we believe that not only our estimation can be used as an, admittedly optimistic, indication of overall viability of ontologies for software engineering practices; but also that the performed analysis reveals valuable insights on the interaction between ontologies and widely accepted quality characteristics of software products as well as of properties of the development process.