*i*SemServ: Facilitating the Implementation of Intelligent Semantic Services

J.Mtsweni, E.Biermann and L.Pretorius

School of Computing, University of South Africa, Pretoria, South Africa
e-mail: mtswejs@unisa.ac.za; bierman@xsinet.co.za; pretol@unisa.ac.za

Abstract

The process of developing semantic services is viewed by service developers as being complex, and tedious. The main barriers that have been identified include a steep learning curve for emerging semantic models and ontological languages, the lack of integrated tool support for developing semantic services, and lack of interoperability between emerging semantic technologies and matured Web service technologies. In addition, current efforts that are meant to ease the implementation of semantic services are fragmented; that is, developers are required to use a combination of disconnected tools to realize semantic services. Moreover, existing semantic technologies are tightly coupled to specific semantic models and service architectural styles; leading to restrictive development environments. In this paper, an *iSemServ* framework is proposed, and implemented as an Eclipse plug-in with the core objective to facilitate, unify, and accelerate the process of developing intelligent semantic services using semantic models and service architectural styles of choice. Experimental evaluations demonstrate that a solution, such as *iSemServ* has the potential to minimize some of the barriers associated with building intelligent semantic services.

Keywords

iSemServ, Eclipse, Semantic Web Services, Ontologies, Intelligent Agents, Model-driven

1. Introduction

The applicability and benefits of employing Semantic Web Services (SWS) also referred to as *semantic services* in this paper are well studied and documented in academia and industry. In (Bachlechner, 2008, de Bruijn et al., 2005a, Janev and Vranes, 2010), the benefits that could be relished by businesses from utilizing semantic technologies are highlighted. These include: (1) improved representation, sharing, searching, reasoning, and re-use of data and services on the Web, (2) anywhere and anytime dynamic connection of business partners through services, and (3) automation of various tasks on the Web such as service discovery, selection, composition, choreography, orchestration, execution, and monitoring.

Nevertheless, in spite of all the benefits, the practical uptake and real-world implementation of semantic services has remained minute to date. A number of scholars (Bachlechner, 2008, Agre et al., 2007, Cardoso, 2007, Daniel et al., 2010) have identified some major challenges that are contributing to this lack of implementation and usage. Some of the major issues that are common include the lack of unified semantic service development tools, non-integration of semantic technologies into existing matured Web services technologies, steep learning curve for emerging semantic models, complex semantic description languages, and the lack of standardisation within the semantic services domain. In addition, existing platforms do not support the development of semantic services that are intelligent beyond the application of ontologies as envisaged within the Semantic Web Services Architecture (SWSA) (Burstein *et al.*, 2005). Other observations are that existing semantic tools are fragmented and tightly coupled specific semantic models and languages; leading to prolonged service development process, and restrictive environments.

In this paper, we thus attempt to address some of the identified challenges by proposing a unified semantic service creation framework that support multiple semantic description models, and service architectural styles called *iSemServ* (from: *I*ntelligent *Sem*antic *Serv*ices). The key objective of the proposed framework is to facilitate the process of implementing intelligent semantic services through a unified, interoperable, and extensible environment.

The remaining sections of this paper are structured as follows. Section 2 provides background information related to intelligent semantic services and its fundamental building blocks. Section 3 discusses the underlying *iSemServ* framework that provides a blueprint for facilitating the process of building intelligent semantic services. Section 4 demonstrates the implementation of the proposed *iSemServ* framework as an Eclipse plug-in. In Section 5, the evaluation results for the suggested solution are presented with the main focus on performance, and Section 6 concludes the paper with a summary and possible future research.

2. Intelligent Semantic Services

Semantic services are geared towards addressing the drawbacks of syntactic services by mapping syntactic descriptions with semantic descriptions (Lu et al., 2007). Semantic descriptions purport to describe what a service does (i.e. service capability), how does it achieve its functionalities (i.e. service behavior), and how to access its functionalities (i.e. orchestration and choreography) (de Bruijn et al., 2005a). Within the semantic services domain, a number of conceptual ontology models have been proposed on how to map semantic descriptions; which are derived from ontologies (Gruber, 1993), into syntactic services. Some of the prominent semantic models to date are heavy-weight models such Web Ontology Language for Services (OWL-S) (Martin et al., 2004) and Web Service Modeling Ontology (WSMO) (Roman et al., 2006).

The main objective of the heavy-weight semantic models is on exploiting commonly agreed upon vocabularies in a form of domain and service ontologies (Kuropka et al., 2008) to represent and describe different aspects of Web services (WS) separately from syntactic descriptions.

OWL-S is one of the first heavy-weight semantic efforts based on the Web Ontology Language (OWL). It provides a structure for defining semantic descriptions through the use of *service profiles*; which semantically describe what a service is capable of offering to prospective consumers, *service model*: describes the specific behavior of a service in terms of its input, output, pre-conditions, and effect, and *service grounding*: describes how a semantic service can be invoked and executed. Comparably, WSMO is an ontology-based model for describing semantic services - based on the WSML (Web Services Modeling Language) (de Bruijn et al., 2005b). WSMO focuses on four elements that are essential in describing semantic services. These are: (1) *Ontologies*: provide formal concepts that can be used by other WSMO elements (2) *Web Services*: describe functional, non-functional, and behavioral aspects of a service, (3) *Goals*: semantically capture users' request that could invoke the service capability, and (4) *Mediators*: handle incompatibilities and mismatches between terminologies used by WSMO elements.

In the context of this paper, an *intelligent semantic service* (IsS) extends and leverages conventional Web services with intelligentt characteristics adopted from the domain of intelligent agents (Mtsweni et al., 2010). This is done to realize the emergence of services that are machine-processable and interpretable; thus leading to minimal human intervention when it comes to service provisioning on the Web (García-Sánchez et al., 2011). The intelligence concept is commonly associated with autonomy, reactive, proactive, and collaborative or social ability properties. Henceforth, adopting the core intelligent agent properties found in (Jennings and Wooldridge, 1998, Protogeros, 2008), and the Semantic Web key enablers found in (Studer et al., 2007, de Bruijn et al., 2008), an *intelligent semantic service is defined as a semantically enabled software unit representing some business functionality that could be accessed through the Web, and is capable of being: (1) Autonomous, (2) proactive, (3) reactive (4) machine-processable and understandable, as well as (5) collaborative* (Daniel et al., 2010).

Furthermore, the fundamental building blocks that comprise intelligent semantic services are: (1) *syntactic descriptions*, (2) *domain ontologies*, (3) *semantic descriptions*, and (4) *intelligence*. These building blocks are essential as they provide a base on how intelligent semantic services could be simply developed within a unified service-creation environment. The following section delves into the suggested framework that is meant to facilitate the realisation of these fundamental building blocks within a unified environment.

3. iSemServ: The proposed framework

In Figure 1, the proposed *iSemServ* framework is demonstrated. The framework adopts a model-driven engineering (MDE) (Qafmolla and Cuong, 2010) technique to enable modelling, specification, and description of intelligent semantic services.

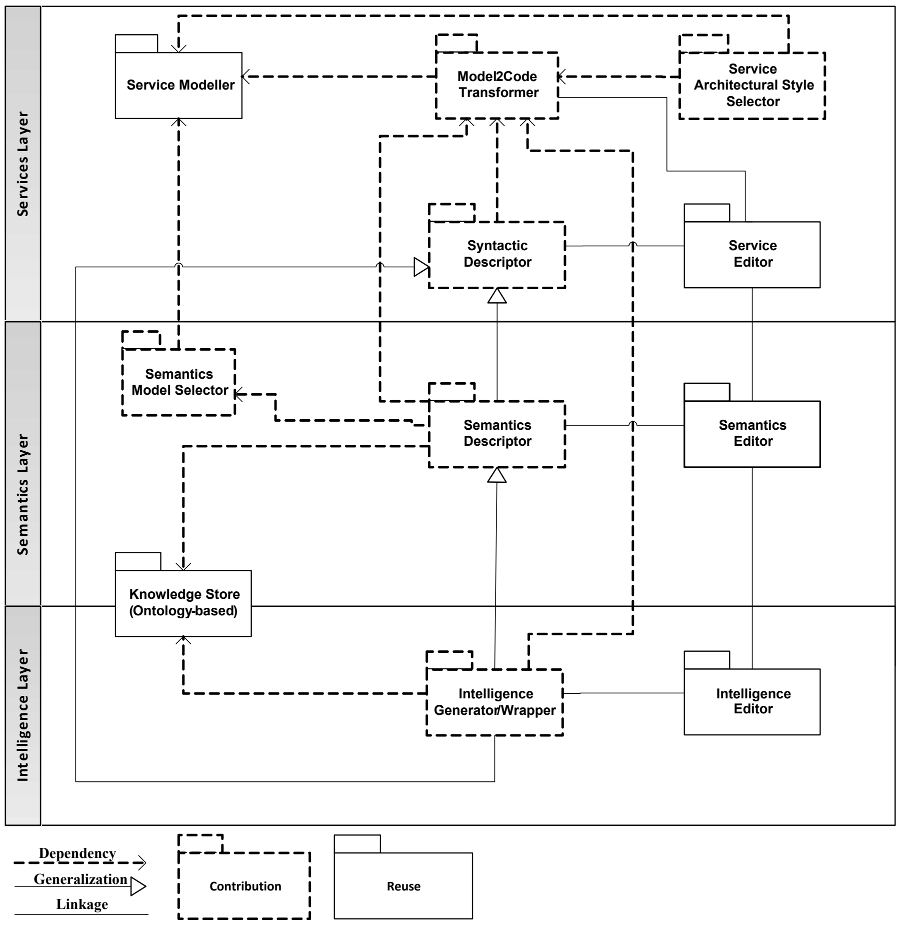


Figure 1: iSemServ framework

The framework is designed based on the overarching principles of uniformity, simplicity, interoperability, decoupling, multiple language support, and intelligence. These principles were identified through an in-depth literature review of related work (Daniel et al., 2010, Mtsweni et al., 2010), and from the challenges that the *i*SemServ framework purports to address. The framework is presented in a multi-layered format for supporting the development of service descriptions, semantic descriptions, and the intelligence functionality in an independent manner. The core layers are: *services layer*, *semantics layer*, and *intelligence layer*. The following subsections explain the purpose of each layer without focusing on the technologies that could be used to implement these layers. Section 4 will highlight some of the technologies that could be used to implement the specific modules of the suggested framework.

3.1. Services Layer

This layer deals with the initial core phases of producing conventional Web services - without semantics and/or intelligence. The first step involves creating structured and annotated service models that represent the functional and non-functional aspects of intended services, and this process is facilitated by the *Service Modeller* module. The service model serves as a basis for other layers as well, particularly for automatically generating relevant service source code stubs (e.g. semantic descriptions), thus reducing service development time. The *Model2Code* transformer uses the defined templates and transformation rules to automatically translate service models into syntactic descriptions assisted by the *Syntactic Descriptor* module; which houses different description templates. In addition, the *Model2Code* transformer translates the service model into partial service logic (e.g. Java). The key in this layer is that services of different architectural styles such as SOAP or RESTful (Filho and Ferreira, 2009) can be auto-generated with the help of the *Service Architectural Style Selector (SASS)* module. New architectural styles or standards could easily be accommodated by simply defining and integrating new templates and transformation rules into the *Model2Code* transformer.

3.2. Semantics Layer

This layer uses the *Service Modeller* module, and relies on the *Model2Code transformer* module to automatically generate domain ontologies and semantic descriptions. Since there are a number of semantic models that could be used to semantically describe Web services, the proposed framework provides the developer with the ability, through the use of a defined UML profile with key stereotypes (e.g. <<WSMO>>), to choose the preferred semantics model. Depending on the annotations; semantic descriptions and domain ontologies can be automatically generated – using the *Model2Code* transformer and the *Semantics Descriptor* module. Because the semantic descriptions and ontologies auto-generated could be incomplete depending on the defined service models; the service engineer is provided with a *Semantics Editor* module, in order to visualize, edit, augment, and validate the generated semantic descriptions and/or ontologies. In this layer, an ontology-based *knowledge store* is also provided, so that developers could also re-use existing ontologies to semantically describe services – where applicable. Additionally, the knowledge store is shared with the intelligence layer, as depicted in Figure 1, for purposes of using the same domain and service knowledge to embed semantic services with intelligence.

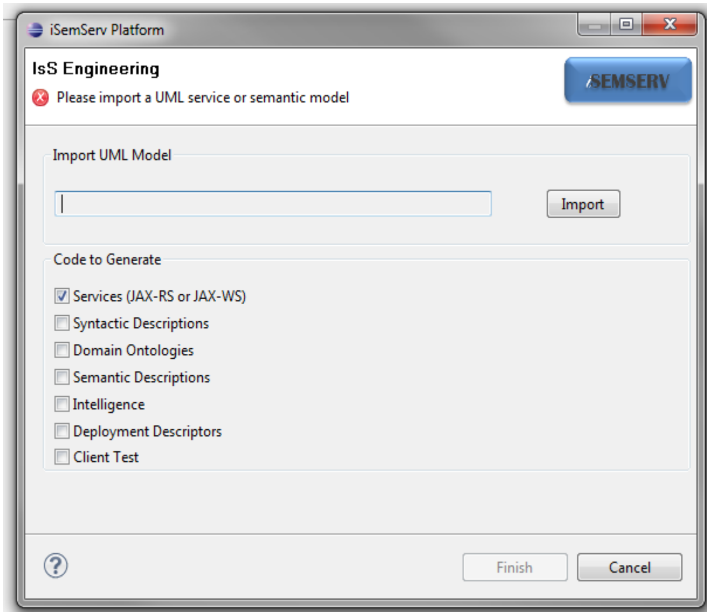
The key in this layer is also that service developers are not restricted in terms of semantic models to use for deriving semantic descriptions. The developer needs to only annotate the service model with the relevant stereotypes that corresponds to the semantic model of choice. The *Model2Code transformer* and the *Semantics Descriptor* will then use the defined templates and transformation rules to auto-generate the relevant code. New semantic models could also be easily accommodated in the framework by defining new templates and transformation rules.

3.3. Intelligence Layer

The *Intelligence Generator/Wrapper* module, as depicted in Figure 1, is responsible for generating all the necessary *intelligence logic* (e.g. automatic service execution) for the developed semantic services and wrapping of semantic descriptions and syntactic descriptions with intelligent properties. The module also relies on the service model and re-usable *intelligence logic* (i.e. agents' behaviour and operations) available through the Intelligence Editor. The properties that are essential for completely realizing the *intelligence* for semantic services are those that implement *autonomous, proactive, reactive,* and *collaborative* behaviours.

4. iSemServ Implementation: an Eclipse Plug-in

The *i*SemServ framework was partially implemented as a unified plug-in within the Eclipse platform; which encompasses a variety of re-usable service engineering components, such as the UML2 SDK (for enabling service models development, UML profile with specific stereotypes), Acceleo (model-to-text language generation framework was used for implementing the *Model2Code* transformer – templates and transformation rules) and JADE (a Java-based agent development environment was used for implementing the intelligent features) and JESS (a Java-based rule engine for realizing other intelligent features, such as reasoning) (Balachandran, 2008). Eclipse is touted as a mature, well-designed, and extensible architecture (Rivières and Wiegand, 2004). The Eclipse environment was chosen because of its openness, and wide acceptance in the service development domain. Figure 2 depicts a user interface for the *iSemServ* Eclipse plug-in; which is meant to facilitate the implementation of intelligent semantic services in a simpler and unified approach.


iSemServ Platform

IsS Engineering

Please import a UML service or semantic model

Import UML Model

Code to Generate

- [x] Services (JAX-RS or JAX-WS)
- [ ] Syntactic Descriptions
- [ ] Domain Ontologies
- [ ] Semantic Descriptions
- [ ] Intelligence
- [ ] Deployment Descriptors
- [ ] Client Test

Figure 2: iSemServ Eclipse plug-in

This plug-in integrates with a number of technologies and templates that makes it possible for the service engineers to only design and import an annotated service model, select the different artefacts (e.g. domain ontologies) that are preferred for the new or existing Web services, and click finish to initiate the auto-generation process. Once the different artefacts have been generated into Eclipse project workspace, the service developer would then use Eclipse integrated editors and other tools to augment, edit, and validate some of the artefacts before deploying the final intelligent semantic service into available semantic service repository for discovery and consumption. A use case scenario implemented in (Mtsweni et al., 2011) demonstrates the type of an annotated service model, semantics models used, and the samples of the generated code (e.g. RESTful services and WSMO-based semantic descriptions).

5. Evaluation

Using real-life project scenarios (Mtsweni et al., 2011), a number of service models were designed in order to experiment and evaluate the usefulness and performance of the *iSemServ* Eclipse plug-in. The models comprised 6 to 1152 classes. In Figure 3, an overall graph is depicted highlighting the performance of the plug-in under varying service model sizes. The overall execution times, presented in seconds (s) illustrate the time it took by the platform to generate the artefacts involved in engineering intelligent semantic services.

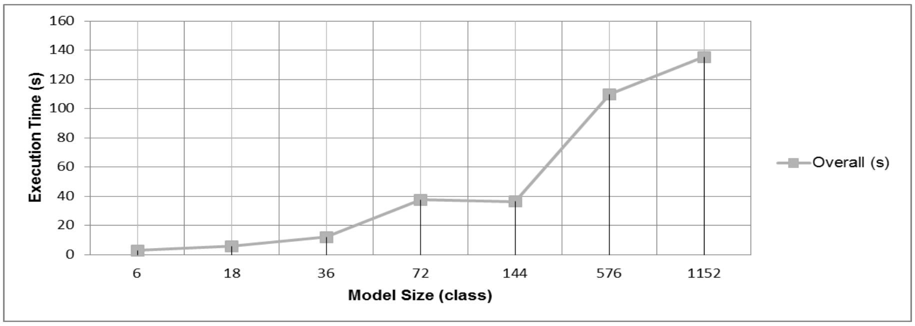


Figure 3: Overall performance of the plug-in

The graph suggests that, as the size of the service model grows, so does the amount of time required to automatically transform the model to different artefacts. For instance, processing a service model with 144 classes required about 36 seconds. An average of 73 seconds was additionally required to generate a total number of 576 Java classes and mapped building blocks, such as semantic descriptions. The code generation execution time increased to 135 seconds for processing a service model made up of 1152 classes. Further evaluations revealed that the auto-generation of syntactic descriptions in the services layer took most of the processing time as compared with other artefacts in the semantics and intelligence layer, as the service model size became bigger. Moreover, it was derived that the auto-generation process can be split into different phases, thereby minimizing the processing load and improving the performance of the plug-in as it is not a requirement that all artefacts need to be auto-generated at the same time. From the experiments, it was evident that the times to generate different artefacts that make up an intelligent semantic service are significantly smaller as compared to when the service developers have to manually generate all the relevant intelligent semantic services' components. The plug-in was developed in a manner that ensures uniformity, simplicity, interoperability, decoupling, and multiple language support. Hence, as it can be noted in Figure 2, the developer only needs to import an annotated model and select all the necessary artefacts that need to be generated. The process that is not accommodated within the plug-in is that of editing the generated artefacts. That process is generally accommodated by the different editors within the Eclipse environment.

6. Conclusion

Semantic services are seen as the main building blocks for the dynamic Web, where user intervention is kept to a minimum. However, there are still a number of challenges that need to be addressed in order to make semantic services commercially relevant. The main hindrance tackled in this paper is the lack of unified tool support for developing semantic services. In addressing this challenge, a model-driven *iSemServ* framework has been proposed that purports to facilitate the implementation of intelligent semantic services independent of a particular semantic model or service architectural style.

The proposed framework follows a top-down approach, where semantic services are modeled, and seamlessly transformed into specific service implementations, syntactic service descriptions, semantic descriptions, and intelligence using transformation rules and templates implemented specifically for the suggested approach. The approach was implemented through a simple Eclipse plug-in with the support of a number of matured tools, such as the UML2 SDK and Acceleo. The plug-in was then evaluated through real-life project scenarios for performance and utility. It was deduced that the suggested approach is one possible solution in minimizing the current hindrances of semantic services, especially due to its principles of standards independency, and automatic code generations.

The main contributions emanating from the suggested and implemented are: (1) an end-to-end approach of developing intelligent semantic services, thus enabling the developer to use one platform to realize all the modules comprising intelligent semantic services, (2) enable service engineers to focus on developing intelligent semantic services in a structured and extensible, and platform-independent manner, (3) support different architectural styles and semantic models by exploiting template-based code generators, and (4) intelligence mapping of services at message and knowledge levels for the purposes of automatically processing semantic service requests and responses by machines with minimal user intervention. Nevertheless, further research points to improving our approach with regard to multiple-language support as currently the framework only relies on templates and transformation rules, which need to be developed for each semantic model or service architectural style.

7. References

Agre, G., Marinova, Z., Pariente, T. & Micsik, A. (2007), Towards Semantic Web service engineering. *Workshop on service matchmaking and resource retrieval in the semantic Web (SMRR 2007)* CEUR

Bachlechner, D. (2008), Semantic Web service research: current challenges and proximate achievements. *International Journal of Computer Science and Applications,* 5**,** 117-140.

Balachandran, B. M. (2008), Developing Intelligent Agent Applications with JADE and JESS. *12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III.* Zagreb, Croatia, Springer-Verlag.

Burstein, M., Bussler, C., Finin, T., Huhns, M. N., Paolucci, M., Sheth, A. P., Williams, S. & Zaremba, M. (2005), A semantic Web services architecture. *IEEE Internet Computing,* 9**,** 72-81.

Cardoso, J. (2007), *Semantic Web Services: theory, tools and applications*, IGI Global.

Daniel, F., Facca, F., Mtsweni, J., Biermann, E. & Pretorius, L. (2010), iSemServ: Towards the Engineering of Intelligent Semantic-Based Services. *Current Trends in Web Engineering.* Springer Berlin / Heidelberg.

De Bruijn, J., Fensel, D., Keller, U. & Lara, R. (2005a), Using the web Service modelling ontology to enable semantic ebusiness. *Communications of the ACM,* 48.

De Bruijn, J., Fensel, D., Kerrigan, M., Keller, U., Lausen, H. & Scicluna, J. (2008) *Modeling Semantic Web Services: The Web service modeling language*, Springer-Verlag Berlin Heidelberg.

De Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M. & Fensel, D. (2005b), The Web Service Modeling Language WSML. . *WSML Final Draft.* DERI.

Filho, O. F. F. & Ferreira, M. A. G. V. (2009), Semantic Web Services: a RESTful approach. *IADIS International Conference WWW/Internet 2009* Rome, Italy.

García-Sánchez, F., Sabucedo, L. Á., Martínez-Béjar, R., Rifón, L. A., Valencia-García, R. & Gómez, J. M. (2011), Applying intelligent agents and semantic web services in eGovernment environments. *Expert Systems***,** 1-21.

Gruber, T. R. (1993), A translation approach to portable ontologies. *Knowledge Acquisition,* 5**,** 199-220.

Janev, V. & Vranes, S. (2010), Applicability assessment of Semantic Web technologies. *Information Processing & Management*.

Jennings, N. R. & Wooldridge, M. (1998), Applications of intelligent agents. *Agent technology: foundations, applications, and market.* Springer-Verlag New York, Inc.

Kuropka, D., Troger, P., Staab, S. & Mathias, W. (Eds.) (2008), *Semantic Service Provisioning*.
Lu, J., Zhang, G. & Ruan, D. (Eds.) (2007), *E-Service Intelligence: Methodologies, Technologies and Applications*, Springer-Verlag Berlin Heidelberg.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. & Sycara, K. (2004), OWL-S: Semantic Markup for Web Services X, W3C.

Mtsweni, J., Biermann, E. & Pretorius, L. (2010), Toward a service creation framework: a case of intelligent semantic services. *2010 Annual Research Conference of the South African*

Institute of Computer Scientists and Information Technologists. Bela Bela, South Africa, ACM.

Mtsweni, J., Biermann, E. & Pretorius, L. (2011), Engineering RESTful semantic services on the fly. *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment.* Cape Town, South Africa, ACM.

Protogeros, N. (2008), Agent and Web services technologies in Virtual Enterprises. IN PROTOGEROS, N. (Ed.), IGI Global.

Qafmolla, X. & Cuong, N. V. (2010), Automation of Web services development using model driven techniques. *The 2nd International Conference on Computer and Automation Engineering (ICCAE).* Singapore IEEE.

Rivières, J. D. & Wiegand, J. (2004), Eclipse: a platform for integrating development tools. *IBM Systems Journal,* 43**,** 371-383.

Roman, D., De Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C. & Fensel, D. (2006), WWW: WSMO, WSML, and WSMX in a nutshell. In R. MIZOGUCHI, Z. S., and F. GIUNCHIGLIA (Ed.) *1st Asian Semantic Web Conference* Beijing, China, Springer-Verlag.

Studer, R., Grimm, S. & Abecker, A. (Eds.) (2007), *Semantic web services: concepts, technologies, and applications*, Springer-Verlag Berlin.