AN ONTOLOGY MANAGEMENT TOOL FOR QOS-BASED WEB SERVICES DESIGN

Marco Comerio, Flavio De Paoli and Gianluigi Viscusi
Universitá di Milano-Bicocca, via Bicocca degli Arcimboldi 8, 20126 Milano, Italy

Keywords: Web services, Ontology, Qualities of Service (QoS), Value perspective, Web Service Design Tool.

Abstract: Web Services are the current most promising technology based on the concept of Service Oriented Computing that provides the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols. Quality of Services (QoS) represents a major issue that covers the challenges in the service composition area and in the service oriented engineering. In this paper, we present the Web Services Design Tool (WSDTool). This tool allows the different actors involved in Web services design to define and address QoS requirements.

1 INTRODUCTION

Web Services are the current most promising technology to make business processes accessible inside and across organizational boundaries (Papazoglou, 2003; Khalaf et al., 2006; Leymann et al., 2002). The increasing availability of Web services that offer similar functionalities underlines the need to augment service descriptions with a well-defined set of non-functional properties (NFP) and, in particular, of qualities of service (QoS). In fact, even if a service fulfils all of its functional requirements by providing the required features, it can still be unacceptable if, for example, availability is too little, performance is too poor, or usability does not meet user expectations.

Functional aspects deal with system behavior and its features. Non-functional requirements deal with a large number of quite different types of requirements. Some of those have been deeply analyzed, such as security (e.g., safety, privacy, and authentication), performance (e.g., throughput, response-time, jitter, and scheduling) and dependability (e.g., availability, reliability, and robustness). Others, such as usability and adaptability, are more complex to define and measure.

QoS is grounded in the value system in which service provision and composition are involved. In fact, it is related to the improvement of business perspectives in service design. According to the Service-Oriented Computing roadmap (Papazoglou et al., 2006), such a perspective represents a major issue that covers the challenges (i) in the service composition area (QoS-aware service compositions and Business-driven automated compositions), and (ii) service-oriented engineering, by focusing on the real world grounding of the services and on user-driven design. The latter is important to involve the different stakeholders of an organization (e.g. managers, business modelers, service designers, etc.) in the design activity.

In this paper, we present the Web Services Design Tool (WSDTool) that allows the different actors involved in the design of a Web service to deal with quality requirements. WSDTool provides a graphical interface aiming at supporting ontologies management and modeling activities. The tool was designed to primarily guide the execution of the Web Service Modeling Design (WSMoD) methodology.

The WSMoD approach consists in incorporating and refining NFP, as well as functional requirements, along the design process. The advantage of managing NFP is twofold: achieving at the end of the process, a ready-to-implement specification, and reducing the risk of delivering unsatisfactory services.

WSMoD makes use of ontologies to understand, discover, classify and reason on NFPs that are related to user constraints, user preferences, technolog-

Comerio M., De Paoli F. and Viscusi G. (2007).
AN ONTOLOGY MANAGEMENT TOOL FOR QOS-BASED WEB SERVICES DESIGN.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Web Interfaces and Applications*, pages 234-239
DOI: 10.5220/0001279602340239

ical features (such as devices and networks), and domain characteristics. Ontologies provide a formal organization of knowledge that is exploited to rationalize decision and evaluation processes. In the current version, we refer to a set of ontologies that were developed in the MAIS project (Pernici, 2006; Comerio et al., 2007). These ontologies provide a description, classification and characterization of services, context (user and channel characteristics) and QoS.

The methodology is composed of five main phases. The *Service Identification* phase specifies, from a business point of view, the features the service will offer, and the business constraints. This phase delivers a complete, informal specification of functional and non-functional requirements. The functional requirements are the input of the *Service Modeling* phase, which has the goal of defining the functional model of the Web Service. The non-functional requirements and the functional model are inputs of the *High-Level Re-Design* phase, whose goal is to revise, and possibly change, the functional model to include non-functional requirements. The next phase, *Customization*, revises the current design model by considering a possible context of execution. Finally, the *Web Service Description* phase translates functional model into standard WSDL interfaces augmented with quality descriptions in WSOL (Tosic et al., 2002). In this paper, only the phases dealing with NFP aspects are considered. For a complete description of WSMoD, the reader can refer to (Comerio et al., 2007).

The paper is organized as follows: Section 2 motivates the introduction of a value perspective in the Web service design and discusses the QoS issue in the area of the Service-Oriented Computing. Section 3 outlines WSDTool characteristics and functionalities. Section 4 illustrates the use of WSDTool during the design process. Finally, Section 5 draws conclusions and presents future works.

2 VALUE PERSPECTIVES ON QOS-BASED WEB SERVICES DESIGN

In business interactions, a major issue consists in enhancing the business perspective over service provision, by developing strategies and tools that provide support in the selection of services according to business goals and processes.

In such a scenario, a major challenge is to understand and make explicit the bind of services to business processes from a value perspective; this knowledge allows the creation of representations that can be exploited to evaluate the competitive advantage of services according to strategic models, such as the value-chain model (Porter, 1985).

The Service Oriented Computing paradigm is centered on the relevance of business modeling for service design and development (Papazoglou et al., 2000). The analysis of different business models creates a shared understanding of how business operates and what kind of services are needed to support a given process at a given time. Value-based approaches to Semantic Web Services (Akkermans et al., 2004) point out the importance of modeling services that are grounded in the real world. These approaches are supported by a value-based perspective on requirements and engineering (Gordijn and Akkermans, 2003), which enhances the relevance of business models in service design. The focus on business models identifies different layers of service design, which span from the business domain, where a service has an outcome, to the technological layer that concerns the implementation of the Web service. In each domain, such different layers need a sharable representation of the involved objects and concepts to make the relations between them accountable at different levels. Furthermore, since a service scenario usually involves different organizations, representations of business domain and implemented Web services are needed.

Besides sharable representation of business objects and business processes, integrated value systems (Papazoglou et al., 2000) need representations of the qualities of service that are requested by the actors involved in the business interaction. These quality issues enhance, according to the business criteria, the value of a service and improve composition across organizational boundaries.

At the state of art, ontologies (Guarino, 1998) are the most suitable way to provide a common representation of different domains of an organization, and to support the interactions among organizations. A discussion on ontologies in the business domain can be found in (Gailly and Poels, 2005). To express the potentiality of ontologies, let's consider, for example, the scenario in which an agent is engaged in a process to find a location for a new point of sales. The agent has to communicate the results of the data-collect activity to the actors that are involved in the strategic management to let them verify whether a location fits certain requirements. This information can be sent via different delivery channels according to the receiver's preferences and available channels (e.g., PC with Internet connection or cellular phone). An ontology can help to support the process; in facts, a service needs a semantic representation of the involved elements, such as delivery channels, organizational functions (e.g., the management levels and roles), and types of message required (e.g., unstructured e-mail, structured data for data base and warehouse).

To the best of our knowledge, the one presented in this paper is an original effort to combine these research topics to provide, at design level, an integrated view on the linkage of services to business processes through a value-chain perspective (Barone et al., 2006).

The approach is to supply suitable representations of QoSs to software designers, which develop the Web services, and to business actors, which (i), as clients, ask for services that are compliant with their value activities and (ii), as providers, want to supply services augmented with rich information about valuable qualities associated with their services. In fact, a major issues in business transactions is the *business commitment* (Papazoglou et al., 2000), which is the result of an agreement between business parties; such a commitment is based on a contract (implicit or explicit) where the qualities of services are key factors.

This model should drive the definition of an ontology of qualities to deliver a tool that can be exploited in different phases of a service life-cycle. In this context a critical issue deals with the wide range of properties that can characterized a QoS (Cappiello et al., 2004). The WSDTool makes use of an ontology that includes complete descriptions of quality concepts -and their relations- to manage non-functional issues during the design process. Back to the process of searching a location for a new point of sales described above, a flexible QoS-based Web Service for message notification need to be developed. In Section 4, the design of FlexSend, a delivery service over different channels, will be discussed as a case study that illustrates how WSDTool uses the formal organization of knowledge provided by the ontology of qualities to rationalize and improve the design process.

3 THE WEB SERVICES DESIGN TOOL (WSDTOOL)

The main objective of the WSDTool is to support the design of a Web service that involves persons with different roles and skills: business experts, domain experts and software designers. Each of these categories is supplied with suitable interfaces and interaction. For example, the interaction with business experts, which are often unfamiliar with technology and tools, should be made as easier and straightforward as possible by user-friendly editors and tailored presentations. On the contrary, the interaction with software designers includes more technical interfaces.

The need of creating different perspectives and views influenced our decision to implement the tool as an Eclipse plug-in (Eclipse, 2005) . Eclipse provides the Plug-in Development Environment (PDE) for creating plug-ins and integrating them with the Eclipse Platform. The possibility to edit views, wizards and editors makes PDE very effective for the development of WSDTool features.

To link the different activities performed by the different actors, ontologies are used as gluing tools. In fact, on one side, they establish a common vocabulary and a repository of information that guide the whole design process. On the other side they collect and organize the semantic associated with the designed services.

To support the execution of the WSMoD methodology, WSDTool provides for the following features:

- *Ontology Management*: import, Navigation and Editing of ontologies. In particular, the tool allows the browsing of the ontology concepts (classes, properties, relations and instances) though an advanced 3D graphical interface.
- *Service Modeling*: management of UML diagrams to develop models of Web services.
- *NFP-aware Modeling*: integration of the functional model with the non-functional model of a Web service via graphical manipulation.
- *QoS Evaluation*: guided evaluation of QoS through graphical interface.
- *Web Service Description*: translation of UML diagrams into WSDL and WSOL documents.

Ontology visualization is based on OntoSphere3D that was chosen for its simple and user-friendly navigation facilities (OntoSphere3D, 2006). The approach, described in (Bosca et al., 2005), follows two different principles: (i) increase the number of “dimensions” (colors, shapes, transparency, etc.) which represent concepts features; (ii) automatically select which part of the Knowledge Base has to be displayed and the level of detail that has to be used in the process, on the base of user direct manipulation of the scene (rotation, panning, zoom, object selection, etc.)

WSDTool provides three different perspectives (one for each user role) composed of different views:

- *Main view*: composed of four different panels: the *3D viewer panel* for 3D representation of ontologies; the *Relation viewer panel* for 3D visualization of relations between concepts or instances; the *New service editor panel* for editing functional and non-functional requirements; finally, the *QoS evaluation panel* for evaluating of QoS.

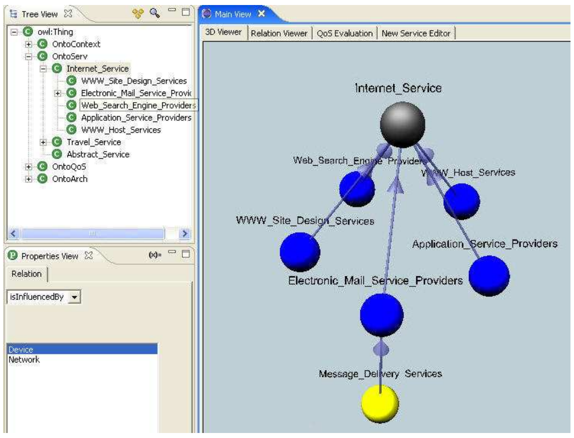


Figure 1: Discovery a category for FlexSend.

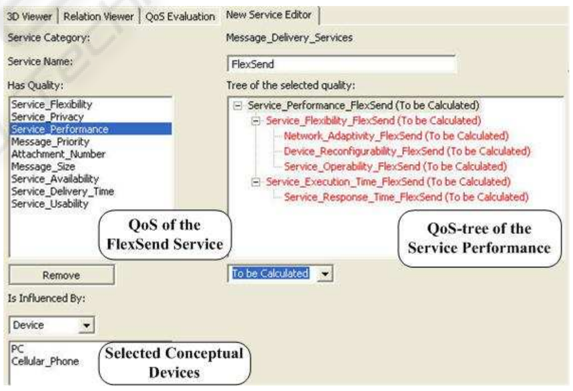
- *Tree view*: a 2D-tree representation of the ontologies to search for concepts, by *Keyword search action*, and to create new instances of services, by *New service action*.
- *Property view*: a visualization of the properties of a selected concept or instance to access the quality evaluation features, by *QoS evaluation action*.
- *Modeling view*: to create UML diagrams from a list of requirements and specific UML profiles.
- *Web Service Description view*: to automatically create WSDL and WSOL descriptions.

4 WEB SERVICES DESIGN WITH WSDTOOL

To illustrate the features of WSDTool, the design of a Web service with the WSMoD methodology is discussed. In particular, the *Service Identification* and the *Customization* phases that mostly involve the quality issues are examined in detail. As a case study, the design of a notification service, FlexSend, that delivers messages over different channels according to specific receivers contexts (preferences, device, activity) is considered.

The *Service Identification* phase delivers a complete specification of functional and non-functional requirements from the business point of view. The work starts with the definition of the FlexSend functional requirements and the creation of a UML use-cases diagram. Then, the phase proceeds with the identification of the category of FlexSend by creating a new instance in the service ontology. WSDTool supports this activity in two ways. In a way, the user makes use of the *keyword search action* to search for categories that are described with particular keywords. In the FlexSend running case, specifying ”message” as keyword, one of the obtained categories is *Message Delivery Service*. The selection of this category triggers the visualization of its properties in *tree view*, *property view* and *main view*.

In the other way, the user browses the service ontology with the *3D viewer* and the *relation viewer* to identify the best category and create the new instance. As shown in Fig.1, WSDTool supports the browsing of ontologies by highlighting the elements of interest while leaving out the others. In a given scene, every concept is clickable with two different results: a right click permits the visualization of all the concept instances; a left click navigates through elements maintaining the current perspective. Starting from a global view, a left click on the *Internet Service* concept determines the scene illustrated in Fig.1. From this scene, it is possible to select the FlexSend category (*Message Delivery Service*), click on it and obtain the relative properties in the *properties view*. In particular, WSDTool allows the user to see that each service included in the *Message Delivery Service* category is characterized by a set of QoS (e.g., *Service Availability* and *Service Usability*) and it is influenced by the network and the device on which it is delivered.


3D Viewer
Relation Viewer
QoS Evaluation
New Service Editor
Remove
QoS of the FlexSend Service
QoS-tree of the Service Performance
Selected Conceptual Devices

Service Category: Message_Delivery_Services

Service Name: FlexSend

Has Quality:
Service_Flexibility
Service_Privacy
Service_Performance
Message_Priority
Attachment_Number
Message_Size
Service_Availability
Service_Delivery_Time
Service_Usability

Tree of the selected quality:
```
⊟ Service_Performance_FlexSend (To be Calculated)
    ⊟ Service_Flexibility_FlexSend (To be Calculated)
        Network_Adaptivity_FlexSend (To be Calculated)
        Device_Reconfigurability_FlexSend (To be Calculated)
        Service_Operability_FlexSend (To be Calculated)
    ⊟ Service_Execution_Time_FlexSend (To be Calculated)
        Service_Response_Time_FlexSend (To be Calculated)
```
To be Calculated

Is Influenced By: Device

PC
Cellular_Phone

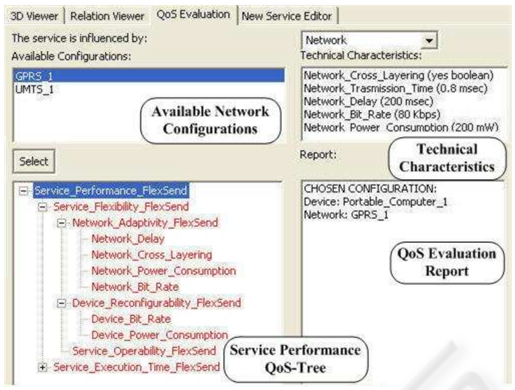
Figure 2: New Service Editor.

After service categorization, the business expert investigates the availability of existing services that satisfy part of the FlexSend requirements to reuse them in the new service. This operation can be performed by examining instances of the selected category, either through the *tree view* (concept instances are visualized as leaves of the tree and marked with the label “I”) or through the *3D viewer* (spheres representing a concept with at least one instance are characterized by a transparent “halo”). Fig.1 shows that no instances are available, so the design of a completely new service is required. The invocation of *new service action* activates the *new service editor*. Fig.2 illustrate what the business expert sees when “FlexSend” is inserted as name of the new service. FlexSend is an instance of *Message Delivery Service* so it inherits all the qualities visualized in the “Has Quality” list. The business expert can change this QoS list by removing or adding new quality requirements. For each quality in the list, WSDTool visualize the related QoS-tree that is created using the dependency relations among qualities specified in the ontology. Fig.2 shows a *Service Performance* QoS-tree in which each quality is labeled as “To be Calculated” meaning that a process of quality evaluation is needed.

The next step is the identification of the influences of context characteristics. A *Message Delivery Service* is influenced by the type of network and device on which messages will be delivered. So, the business expert specifies the conceptual devices and networks (i.e. high-level specification without technological details) that can be used by the final users to interact with FlexSend. As shown in Fig.2, conceptual devices for the FlexSend service are PC and Cellular Phone, while conceptual networks are GPRS and UMTS.

After requirement definition, the FlexSend design proceeds with the others phases of the WSMoD methodology. For space reason, we omit to report details related to *Service Modeling* and *High-level Redesign* phases, so to focus on the *Customization* phase, which is an innovative feature of the tool. The objective is to illustrate how the non-functional requirements are evaluated considering channel profiles concerning real devices, network interfaces, networks and application protocols that providers can exploit for service provisioning. As an example, the evaluation of the *Service Performance* quality requirement is discussed. Fig.3 shows that the considered quality is affected by others, therefore, it must be evaluated by the composition rules that characterize the quality.

The evaluation process starts with the selection of a channel configuration. In our example, this operation consists in the selection of instances of network and device. Technical characteristics of the available configurations are reported in technical characteristics list (e.g., *Network Transmission Time = 0.8 msec*, *Network Delay = 200 msec* etc). The example in Fig.3 shows the evaluation of *Service Performance* with the following channel configuration: *Cellular phone* with *GPRS*. The applied composition rule (an algorithm in this case) is the one associated with the quality in the ontology:


3D Viewer
Relation Viewer
QoS Evaluation
New Service Editor
Available Network Configurations
Technical Characteristics
QoS Evaluation Report
Service Performance QoS-Tree
Select

The service is influenced by:
Available Configurations:
GPRS_1
UMTS_1

Network
Technical Characteristics:
Network_Cross_Layering (yes boolean)
Network_Trasmission_Time (0.8 msec)
Network_Delay (200 msec)
Network_Bit_Rate (80 Kbps)
Network_Power_Consumption (200 mW)

Report:
CHOSEN CONFIGURATION:
Device: Portable_Computer_1
Network: GPRS_1

```
⊟ Service_Performance_FlexSend
    ⊟ Service_Flexibility_FlexSend
        ⊟ Network_Adaptivity_FlexSend
            Network_Delay
            Network_Cross_Layering
            Network_Power_Consumption
            Network_Bit_Rate
        ⊟ Device_Reconfigurability_FlexSend
            Device_Bit_Rate
            Device_Power_Consumption
        Service_Operability_FlexSend
    ⊞ Service_Execution_Time_FlexSend
```

Figure 3: QoS Evaluation Editor.

1. Evaluate *Network Adaptivity* using the Tuple method;
2. Evaluate *Device Reconfigurability* using the Tuple method;
3. Evaluate *Service Flexibility* using the Simple Additive Weighting (SAW) technique;
4. Evaluate *Service Execution Time* using the Average method;
5. Complete the process evaluating *Service Performance* using the SAW technique.

For space reason, we omit the description of methods and techniques involved in *Service Performance* evaluation. The reader can refer to (Ardagna et al., 2005) for a detailed description.

The evaluation of the required qualities in the actual environment supports the validation of the designing service. The service requirements define thresholds that the service needs to meet or overcome (for simplicity, proportional quality dimensions are considered, i.e. the higher value the better quality). The chosen configuration can meet the thresholds but others configurations can provide insufficient values. Considering these results the business expert can decide to re-negotiate business QoS constraints or to revise the Web service design.

5 CONCLUSIONS AND FUTURE WORKS

In this paper, we have presented the Web Services Design Tool (WSDTool) that supports the WSMoD methodology, an ontology-based approach to address QoS issues in the design process of Web services. An innovative features that clearly distinguish WSDTool from other design IDE is the management of QoS representations that cover different perspectives, from business to technical.

The business perspective covers the challenges in QoS-aware service compositions, business-driven automated compositions and service-oriented engineering by focusing on the real world grounding of services and on user-centered design. This approach is important to involve the actual stakeholders in the design activity.

The development of WSDTool is an ongoing work. The current version provides a partial implementation of *Tree view*, *Property view* and the *Main view*. Future works deal with the completion of the WSDTool with the implementation of *Modeling view* and *Web Service Description view*. To create the *Modeling view*, we are evaluating the integration of standard UML 2.0 plug-ins (e.g., Eclipse UML Studio (Omondo, 2006)). Concerning the creation of the *Web Service Description view*, we are working in two directions: one concerning the creation of a WSDL and WSOL-oriented UML profile for the automatic creation of Web service descriptions in WSDL and WSOL; the other concerning the selection of available tools that support the required translation. Currently, we are evaluating UMT (UMT, 2006), a tool that provides for model transformation and code generation based on UML models in the form of XMI.

ACKNOWLEDGEMENTS

The work presented in this paper has been partially supported by the European IST project n. 27347 SEEMP - Single European Employment Market-Place and the Italian FIRB project RBNE05XYPW NeP4B - Networked Peers for Business.

REFERENCES

Akkermans, H., Baida, Z., Gordijn, J., Pena, N., Altuna, A., and Laresgoiti, I. (2004). Value webs: Using ontologies to bundle real-world services. *IEEE Intelligent Systems*, 19(4):57–66.

Ardagna, D., Comerio, M., DePaoli, F., and Grega, S. (2005). An hybrid approach to qos evaluation. In *EMMSAD '05: Proceedings of the International Workshop on Exploring Modeling Methods in Systems Analysis and Design*, pages 581–592, Porto, Portugal.

Barone, D., Viscusi, G., Batini, C., and Naggar, P. (2006). A Repository of Services for the Government to Businesses relationship. In *NGITS '06: Proceeding of the Next Generation Information Technologies and Systems*, Kibbutz Shefayim,Israel.

Bosca, A., Bonino, D., and Pellegrino, P. (2005). Ontosphere: more than a 3d ontology visualization tool. In *SWAP '05: Proceedings of the Semantic Web Applications and Perspectives Workshop*, Trento, Italy.

Cappiello, C., Missier, P., Pernici, B., Plebani, P., and Batini, C. (2004). Qos in multichannel is: The mais approach. In *ICWE Workshops '04: Proceedings of the International Conference on Web Engineering*, pages 255–268, Munich, Germany.

Comerio, M., DePaoli, F., Grega, S., Maurino, A., and Batini, C. (2007). Wsmod: A methodology for qos-based web services design. *International Journal of Web Services Research*.

Eclipse (2005). Eclipse platform technical overview. Technical report, Available at: http://www.eclipse.org/articles/whitepaper-platform-3.1/eclipse-platform-whitepaper.html.

Gailly, F. and Poels, G. (2005). Towards an OWL-formalization of the resource event agent business domain ontology. Technical report, ISWC 2005 - OPSW workshop.

Gordijn, J. and Akkermans, J. M. (2003). Value-based requirements engineering: exploring innovative e-commerce ideas. *Requir. Eng.*, 8(2):114–134.

Guarino, N., editor (Amsterdam, 1998). *Formal ontologies and information systems*. IOS Press.

Khalaf, R., Keller, A., and Leymann, F. (2006). Business processes for web services: Principles and applications. *IBM Systems Journal*, 45(2):425–446.

Leymann, F., Roller, D., and Schmidt, M.-T. (2002). Web services and business process management. *IBM Systems Journal*, 41(2):198–211.

Omondo (2006). *Eclipse UML Studio*. Available at: www.omondo.com.

OntoSphere3D (2006). *OntoSphere3D*. Available at: http://sourceforge.net/projects/ontosphere3d.

Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F., and Krämer, B. (2006). Service-oriented computing: A research roadmap. In *Service Oriented Computing (SOC)*, number 05462.

Papazoglou, M. P. (2003). Web services and business transactions. *World Wide Web*, 6(1):49–91.

Papazoglou, M. P., Ribbers, P., and Tsalgatidou, A. (2000). Integrated value chains and their implications from a business and technology standpoint. *Decis. Support Syst.*, 29(4):323–342.

Pernici, B. (2006). *Mobile Information Systems. Infrastructure and Design for Adaptivity and Flexibility (the MAIS approach)*. Springer.

Porter, M. E. (1985). *Competitive Advantage*. The Free Press, New York.

Tosic, V., Patel, K., and Pagurek, B. (2002). Wsol - web service offerings language. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 57–67, London, UK. Springer-Verlag.

UMT (2006). *UML Model Transformation Tool*. Available at: http//umt-qvt.sourceforge.net.