RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS
No. 85 2009
Advanced Information Technologies for Management – AITM 2009

Rolf Kluge(1,2), André Ludwig(1), Bogdan Franczyk(1), Leszek Maciaszek(2,3)

(1) University of Leipzig, Leipzig, Germany
e-mail: {rkluge, ludwig, franczyk}@wifa.uni-leipzig.de
(2) Macquarie University, Sydney, Australia
e-mail: {rkluge, leszek}@science.mq.edu.au
(3) Wrocław University of Economics, Wrocław, Poland
e-mail: leszek.maciaszek@ue.wroc.pl

TOWARDS A SERVICE-ORIENTED CAPABILITY-DRIVEN SELECTION OF COTS BUSINESS SOFTWARE

Abstract: This paper provides an insight to the in-progress research on a service-oriented capability-driven approach that supports the selection of COTS business software. Service-oriented means that the concepts of service-orientation (i.e. models, methods and techniques) are adopted in the selection and that COTS business software is perceived as service provider having a set of service capabilities. Capability-driven means that the selection process is driven by matching required service capabilities of service consumer with provided capacities and abilities of COTS business software.

1. Introduction

Commercial off-the-shelf (COTS) solutions refer to software products that one can buy off a manufacturer's virtual store shelf (i.e. from a price list or a catalogue). In contrast to developing software from scratch, COTS products provide prebuilt solutions that satisfy certain standard needs at reasonable cost. COTS solutions are defined as software products "sold, leased, or licensed to the general public; offered by a vendor trying to profit from it; supported and evolved by the vendor, who retains the intellectual property rights; available in multiple, identical copies; used without modification of the internals" [Oberndorf et al. 2000].

According to this definition a COTS solution can be an operating system, a database, an office suite, an Enterprise Resource Planning (ERP) solution, etc. This paper focuses on COTS business software (CBS) which refers to cross-industry application software or to a vertical market application software (cf. [*North American...* 2009]).

In comparison to the development of new software from scratch, the use of CBS products has many benefits, but it also raises new challenges. One benefit is that CBS solutions are built for several customers. Hence, the development costs are shared and CBS solutions are available at a reasonable price (i.e. cheaper than developing a unique solution). Further, using a prebuilt solution might be faster than developing a new one [Alves, Castro 2001]. Another benefit is that customers can take advantage of a proven product which has been tested in-the-field with continued improvements in quality [Alves, Finkelstein 2002a]. CBS products provide established and standardized solutions. However, challenges and issues are, for instance, that vendors are interested in selling as many products as possible and, therefore, design their CBS to meet the needs of the marketplace instead of satisfying individual requirements. Thus, it is not ensured that a candidate CBS product will meet all requirements of a particular user [Alves, Finkelstein 2002b]. Furthermore, organizations have limited access to the internal design of a CBS product. Solutions are designed as a "black box". The description of a CBS solution might be a confusing incomplete textual document, which limits the chance for customers to verify in advance whether the desired requirements can be fulfilled. Hence, the selection and evaluation of COTS solutions (and CBS solutions respectively) is a non-trivial task.

This paper gives an insight into the in-progress research on a service-oriented approach that supports the evaluation/selection of CBS solutions. Service-oriented means that the concepts of service-orientation are adopted. In particular, business entities and CBS are perceived as service consumer and service provider. From that perspective, CBS provides functionalities in terms of services and, in turn, business entities require a set of services that fulfil their demands. In other words, a service consumer (the business entity) is looking for a set of services (from the CBS), which suits the requirements best. By adopting the service-oriented paradigm, the concept of Semantic Web Service is also applied. Hence, concepts of service discovery are applicable for a (semi-) automatic reasoning about services (i.e. functionality of CBS) that satisfy certain business goals (i.e. requirements of business entities).

The remainder of this paper is structured as follows. In Section 2 we describe the problem areas and research questions, which are tackled. Section 3 presents related work in these areas. Section 4 introduces the service-oriented approach, including foundations, the overall picture and the explanation about the methods, models and techniques. Section 5 discusses the approach in general and models, methods and mechanisms in detail. Moreover, it provides an outlook for further steps.

2. Problem description

From the viewpoint of our service-oriented capability-driven approach a research question is: How to evaluate and select CBS in an effective and efficient way in order to find a product that suits the business requirements? This question refers to three areas.

The first area is Requirements Engineering. The question is: How to describe business requirements (i.e. required capabilities) in a formal way? This includes the elaboration of a formal model for expressing requirements and a method for its implementation (i.e. a process that defines how to elicit a formal description from business objectives).

The second problem area deals with the formalization of CBS. The question would be: How to describe a CBS in a formal way? Similar to the question about requirements, this question encompasses establishing a formal model for expressing CBS and the elaboration of a method that describes how to move from graphical and verbal descriptions (e.g. graphical user interface, manuals, training material, etc.) to a formal description of a CBS.

Finally, the third question deals with comparing and matching the two formal models. The question is: How to compare a formal description of business requirements and a formal description of CBS in order to find a set of CBS functionalities that suit the business requirements?

It is important to note that the three problem areas are interrelated, i.e. an adequate comparison and matching relies on a proper formal description.

3. Related work

COTS software has been discussed widely in theory and practice. Research in this area can be divided into four groups: (1) definition and classification, (2) selection and evaluation, (3) integration, and (4) maintenance. Since this paper addresses the evaluation and selection of CBS, related work in this area is discussed next (Section 3.1). Related work in the areas of the formal description of requirements, formal description of CBS (or software systems respectively), and the matching are discussed in Sections 3.2 to 3.4.

3.1. COTS selection and evaluation

Many papers have been written pursuing the question “How to select the software product that fits certain requirements best?” Some of them address problem descriptions only (e.g. [Carney 1999; Carney, Wallnau 1998]), others provide methods and techniques.

The OTSO (Off-the-shelf Option) method [Kontio 1995] was proposed in 1995 as one of the first publications. OTSO provides some criteria for evaluation, mentions cost-benefit analysis and supports decision-making by the consolidation of results. J. Kontio [1995] points out that requirements are the key for COTS evaluation. However, the process of requirements specification is not mentioned (cf. [Alves, Castro 2001).

PORE (Procurement-Oriented Requirements Engineering) [Ncube, Maiden 1999] is an iterative template-based method for Requirements Engineering.

Although it is focused on COTS solutions, it does not address how software features are described and how to relate requirements to COTS software features. N. Maiden and C. Ncube [1998] noted that "requirements engineer[s] must model not only customer requirements but also each software product". However, PORE refers to Requirements Engineering techniques for describing COTS features, but it does not provide a model and a formal description.

Other COTS evaluation methods and techniques are available [Alves, Castro 2001; Leung, Leung 2003]. These approaches (similarly to OTSO and PORE) address COTS solutions in general. They do not explicitly focus on CBS. Further, no comprehensive method and model for formalization requirements and COTS features is provided. Moreover, all approaches do not address semantics. Accordingly, they give guidelines for evaluators, but do not allow (semi-)automatic comparison of requirements and COTS features as put forward in this paper.

3.2. Formal description of requirements

Since Requirements Engineering (RE) addresses methods, models and techniques for describing requirements, the formal description of requirements is a task of RE. We consider formalisms in terms of semantics.

As one of the first publications [Ramesh, Jarke 2001] suggest the usage of semantics in RE. The following discussion presents recent efforts related to the use of semantic web service technologies in RE.

Dobson and Sawyer suggest the use of ontologies within RE for modelling domain knowledge explicitly and in a machine interpretable way. With this, traceability and automatic checking of consistencies should be possible. However, they just mention the limitations of traditional RE and only points out the benefits of a semantic-driven RE.

Mayank et al. [2004, p. 5] propose semantic web technologies in RE to represent requirements. For that purpose the properties: descriptive name, comment and requirement description (as text), are specified (among others) in order to describe requirements in a semantic way. However, these properties are more like requirements' meta-data, since the requirements description is still in textual form, for instance. This helps finding the requirements, but it is insufficient for comparing functional requirements with CBS features.

Approaches to semantically describe services are defined [Akkiraju et al. 2005; Battle et al.; Martin et al. 2004; Roman et al. 2006]. Related work in this area is examined next (Section 3.3). One of the approaches – WSMO [Roman et al. 2006] – does not only address services, but also corresponding goals. Goals are defined as "problems that should be solved" [Roman et al. 2006, p. 3]. They are specified by a service consumer in order to express what he/she expects from a service. So, since goals are interrelated with requirements [Lamsweerde 2001], it is reasonable to illuminate the structure of service goals. Basically, a goal contains an ontology, which embodies the used terminology, and a capability description. Ontology and capability are described by the semantic mechanisms: concepts, relations, instances and axioms (cf. [Lacy 2005]). Unfortunately, WSMO provides only the formal model and the language. There is no method or process that explains how to express business requirements in terms of service goals.

3.3. Formal description of software systems

"... requirements engineer[s] must model not only customer requirements but also each software product" [Maiden, Ncube 1998]. According to this statement, it is important to establish a method and a model that formally and semantically supports a description of CBS functionality. Related work in this area can be broadened (i.e. not focused on CBS only), since the functional description of software systems, software components and software services is similar and just a matter of granularity.

In the area of Semantic Web Services there are four approaches for a semantic description of services [Akkiraju et al. 2005; Battle et al. 2005; Martin et al. 2004; Roman et al. 2006]. The concept for describing service functionality is similar in all approaches, i.e. service functionality is described by pre- and post-condition using a common terminology/ontology and logical axioms. However, these models are not focused on CBS. Further, although the models are comprehensive, there is no method of defining how to gain a semantic description from existing services.

All-in-all, there are some models that might be useful for describing CBS functionality, but there is no method that describes how to derive a formal semantic descriptions from CBS. Some papers (such as [Maiden, Ncube 1998, pp. 53]) refer to RE methods for this purpose. However, none of them addresses semantic descriptions.

3.4. Matching requirements and software functionality

In service-oriented research, matching mechanisms relate to semantic service discovery area. Basically, the service discovery is defined as the "location of services that fulfil a user goal" [Keller et al. 2004]. Service discovery is a popular research topic (e.g. [Alonso et al. 2004; Gonzalez-Castillo et al 2001; Keller et al. 2004; Pathak et al. 2005]). However, all investigations are focused on services only. They do not consider CBS. There is no approach that uses service discovery mechanisms in order to support the selection of CBS for certain business requirements.

4. Approach

The service-oriented approach for the selection of CBS solutions is based on adapting service-oriented concepts. Since the terminology is not established and controversially discussed in the literature, Section 4.1 lays the foundation for a common understanding. Sections 4.2 and 4.3 explain the overall framework as well as methods, models and techniques needed.

4.1. Foundation

Service-orientation is a design paradigm that considers services as fundamental building blocks for business and computational capabilities. The paradigm encompasses design principles for services, an architectural model (Service-Oriented Architecture) and related concepts, technologies and frameworks (Service-Oriented Computing). The following paragraph illuminates the service concept.

Service concept. The fundamental logical element of service-orientation is the service. In literature there are many definitions of service – varying in granularity and giving different interpretations. The Open Group defines a service as "a logical representation of a repeatable business activity that has a specified outcome, such as 'check customer credit', 'provide weather data', or 'consolidate drilling reports'. It is self-contained, may be composed of other services, and is a 'black box' to its consumers" [The Open Group 2007, p. 6]. Other definitions are " 'services' [...] are well defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services" [Papazoglou, Heuvel 2007] and "A service is an exposed, self-contained, and platform-independent piece of functionality with well-defined interface that can be dynamically located and invoked" [Bonati et al. 2006]. The definitions agree on a service as a self-contained logical representation, but refer to different entities, i.e. business activity, business functionality or platform-independent piece of functionality. This shows that the term is differently used in business and in computational contexts.

A general model which explains the service concept including its components is provided by [Karakostas, Zorgios 2008]. An adopted picture is illustrated in Figure 1.

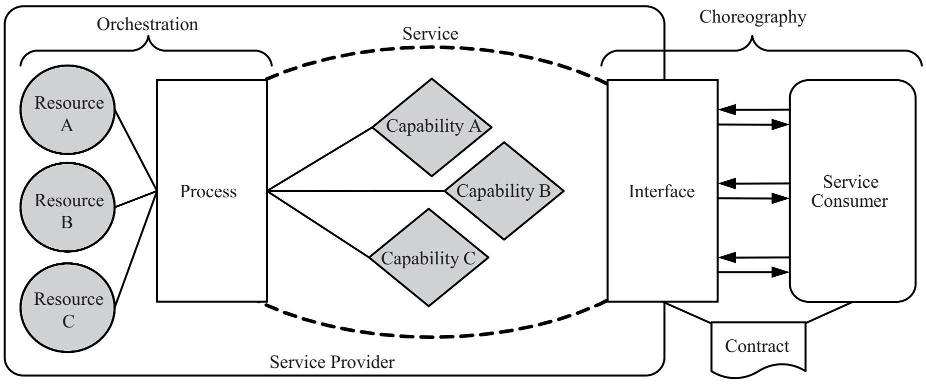

Figure 1. The service concept

Source: [Karakostas, Zorgios 2008, pp. 4 ff., 23; Fensel et al. 2007, p. 67].

Basically, a service is hosted by an entity called service provider and consumed by an entity called service consumer. The service provider has access to some resources and coordinates their delivery. Resources can be either tangible (e.g. the trains of a public transport company) or intangible (e.g. train schedule information). Resources can be owned by the service provider (e.g. trains of a public transport company) or acquired from external entity (e.g. electric power from an electrical company). Usually, a combination of tangible, intangible, owned and acquired resources are necessary for a service. Resources are coordinated by a process. The coordination process, which is also termed the orchestration, leads to service capabilities. Service capabilities embody the functionality of a service. In general a service provides more than one capability. Service capabilities provide certain utilities to service consumers. However, the service capabilities can only be accessed through the service interface, i.e. the interface is the only way for a consumer to communicate and interact with a service. The interaction between the service consumer and the service is a process called choreography. Usually service provider and service consumer negotiate a contract before interaction (cf. [Karakostas, Zorgios 2008, pp. 18]).

As suggested by the service concept and stated in one of the definitions [The Open Group 2007], a service is a "black box" for the service consumer. The service implementation, i.e. the orchestration process and concrete resources, is hidden behind the service interface. Only the specification, which explains what the service does and how to interact with it, is available to the consumer.

Service description. According to M.P. Papazoglou and D. Georgakopoulos [2003, p. 23], service descriptions "are used to advertise the service capabilities, interface, behaviour, and quality". Service capability and service interface refer to the terms used in the aforementioned service concept definition. The description of service capabilities states the conceptual purpose as well as the expected outcome of the service. The service interface description contains signatures, i.e. input parameters, output parameters, error parameters and message types, which are important for the consumer in order to access/bind the service. The description of the (expected) behaviour of the service refers to a description of the interaction between the service consumer and the service. Basically, this can be done by depicting a scenario-based workflow of the choreography process. Behaviour and capability description are interrelated. Finally, the Quality of Service (QoS) description publishes non-functional attributes[1], such as performance metrics (e.g. response time), service cost, security attributes as well as reliability, scalability, and availability [Papazoglou, Georgakopoulos 2003].

[1] Although M.P. Papazoglou and D. Georgakopoulos define Quality of Service (QoS) as a description of "important functional and nonfunctional service quality attributes" [Papazoglou, Georgakopoulos 2003, p. 26], we consider QoS in a narrower context of non-functional service attributes, in order to have a clear distinction between them and the functional attributes exposed by the service capabilities.

Semantic service description. The semantic description of services aims at making services machine detectable. This means, for instance, that a search query for a certain service is not based on the syntax, but the semantic meaning. A semantic service description includes the description of service capabilities (i.e. pre- and post-condition), service interface (i.e. in- and output parameter), service behaviour (i.e. workflow), and service quality properties (i.e. non-functional properties) by using ontologies (including the elements: concepts, relations, instances and axioms [Fensel et al. 2002]). This can lead to a machine understandable description of a service (cf. [Fensel et al. 2007]). Since this paper is focused on the functional properties, the semantic description of service capabilities is addressed first.

4.2. The overall picture

The overall approach is depicted in Figure 2. The rectangle in the middle contains the service-oriented elements used within the approach. As shown at the top of Figure 2, there is a service consumer and service provider. Service provider offers a set of services (one or many). The functionality of a service is expressed by one or many service capabilities. On the left-hand side the service consumer expresses its needs in terms of one or many goals. Each goal contains one or many goal capabilities which express the needed functionality of the service consumer. Thus, both types of capabilities hold descriptions of functionality inside. The goal capabilities express the functionality needed by the service consumer. The service capability expresses the functionality offered by the service (the service provider respectively). Having both descriptions in a similar shape, required capabilities and provided capabilities can be compared and matched.

Further, Figure 2 contains the business entity on the left hand side. This refers to an enterprise, a company, an organization or a business unit. The business entity has one or many business requirements concerning a proposed IT solution. Business requirements contain one or many functional requirements. In general, requirements can be divided into functional and non-functional requirements. Since our approach is initially focused on functional aspects, only functional requirements are mentioned here. The shape of the requirements is depicted as a cloud. This refers to requirements which are not formalized and expressed by e.g. text documents. On the right hand side of Figure 2, there is the CBS which contains one or many components. Each component bears some functionality. Like in the case of requirements, only the functional aspects are considered at first. Again, the functionalities are illustrated as a cloud, referring to the fact that the descriptions of the functionalities are not formalized, i.e. hidden behind the user interface or expressed in text documents.

The arrows between business and the service consumer, and between CBS and service provider, expose the main idea behind the service-oriented approach for matching business requirements to CBS functionality. It means that business entities are perceived as the service consumer and CBS are perceived as the service

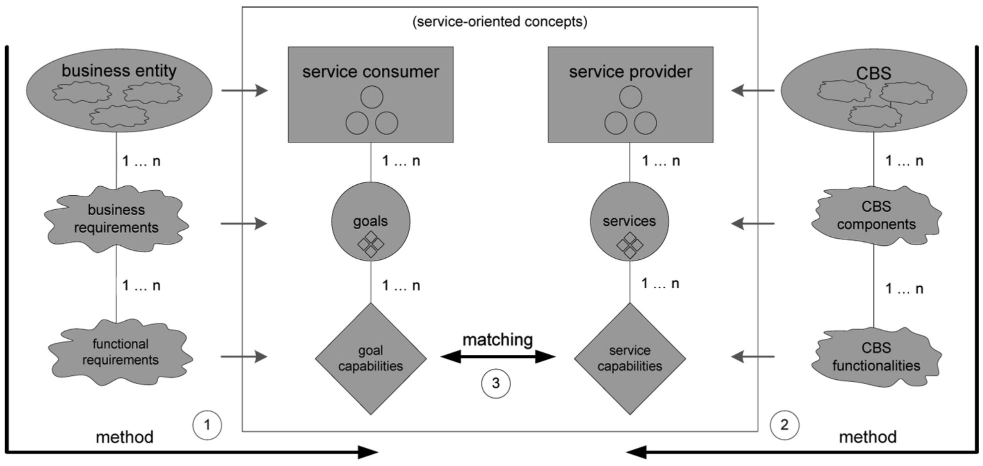

Figure 2. The overall picture

Source: own elaboration.

provider. From this perspective business requirements refer to goals and CBS components to services. Hence, functional requirements can be expressed in terms of required capabilities and CBS functionalities in terms of provided capabilities. Having both descriptions in terms of capabilities expressed as semantic service models, techniques of service discovery can be applied in order to match CBS functionalities to functional requirements. This allows a (semi-)automatic matching of business requirements and CBS components.

4.3. Methods, models and techniques

The work on methods, models and mechanisms is still under investigation. At this stage we can provide a draft about methods, models and techniques needed. Specifications for these will be proposed in the forthcoming papers.

First of all, a method for describing requirements is needed (cf. Figure 2: 1). This method will contain three major steps: eliciting requirements, specifying requirements and formalizing requirements. This, in turn, requires three description models. For eliciting requirements, an informal text document should be sufficient. The requirements specification should be described by a semiformal language, such as UML.[2] The formal description of requirements should be done in terms of goal capabilities. Thus, the formal model of requirements is expressed in terms of pre- and post-conditions using a semantic language (i.e. ontologies). Since a lot of work has been done in the area of eliciting requirements and specifying them with UML,

[2] http://www.uml.org/.

the main challenge will be to establish a method that ensures the translation of semiformal models into semantic formal models.

Secondly, a method for describing CBS functionality is needed (cf. Figure 2: 2). Starting from CBS's graphical user interface and manual descriptions, a CBS can be described in a semiformal model. UML class diagrams and workflow diagrams can be used in order to describe the functionality and the business cases for which a CBS is intended. Next, the CBS description in terms of UML can be transformed into a semantic description in terms of service capabilities. The transformation process from CBS's graphical user interface and textual description to UML will follow recognized RE methods. The main challenge is to translate the semiformal model (i.e. UML) into a formal description of semantically defined service capabilities.

Thirdly, an in depth mechanism and corresponding algorithm needs to be defined that compares goal capabilities and service capabilities in order to find a set of CBS capabilities that fit the discussed business case. For this purpose, the matchmaking mechanisms researched in the service discovery seem to be applicable.

5. Discussion

The service-oriented capability-driven approach for the selection of CBS products contains three problem areas: First, the formal and semantic description of requirements, second, the formal and semantic description of CBS, and third, the comparison between both. Each of these problem areas have already been discussed in theory and practice (cf. Section 3). Main challenges are the definition of proper description models and methods for elicitation. We think that Semantic Web service descriptions and RE methods are suitable for that purpose. Further, methods and models should enable later on the comparison of requirements and CBS. For this purpose, we think that service discovery mechanisms are applicable. Since all of the three problem areas are still under investigation, there is no final solution offered here for each of these. At any rate, we do not claim a fully-automated solution for selecting/evaluating CBS products, but we hope to provide a reasonable and significant improvement in comparison to a manual CBS selection process. Our approach will provide a semi-automatic process supporting the decision making during the selection and evaluation phase of CBS solutions.

Service-orientation is one of the most discussed computing paradigms these days. It attempts to address not only IT concerns but business concerns as well. Comparing business requirements and CBS products fits into this area. We think, by applying service-oriented concepts to CBS area, one can benefit from recent and future research results. Moreover, most of CBS solutions do not have service interfaces. Hence, a CBS description in terms of services as proposed in our research could lead to a service-oriented blueprint for CBSs.

Does it actually make sense to semantically describe requirements and CBS features and can the benefits justify the effort? Indeed, the effort for a complete semantic description of a CBS system is high. However, the knowledge about this could be placed in a central knowledge base. Similar to eCOTS [Mielnik et al. 2003] the information about the CBS could be placed on a web platform and accessed, edited and shared between several participants. Thus, the effort and costs for elicitation could be shared between stakeholders.

Concerning the effort for a semantic description of requirements, we argue that in case of CBS not all requirements have to be elaborated. We suggest an iterative procedure of elimination, similar to PORE [Ncube, Maiden 1999], i.e. one starts with a small set of requirements and eliminates those CBS products that do not fit to them until only one CBS is left. This approach would minimize the effort for semantic description of requirements.

6. Conclusion

This paper has offered an insight to the work on an approach that supports the selection of CBS products by using service-oriented capability-driven approach. The basic idea is that business entities are perceived as service consumers having functional expectations (in terms of goal capabilities). On the other hand, CBS products are perceived as service providers bearing a set of service capabilities. By defining goal capabilities and service capabilities in the same format (i.e. based on the same model), the two can be compared and matched.

This paper has described methods, models and techniques needed within the approach. Current and future work and the forthcoming papers will address the definition of distinct models and methods for elicitation and techniques for matching.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research by the project “Logistics Service Bus” (03IP504) at the Information Systems Institute of the University of Leipzig.

References

Akkiraju R. et al. (2005), *Web Service Semantics – WSDL-S, W3C Member Submission*, http://www.w3.org/Submission/WSDL-S/.

Alonso G., Casati F., Kuno H. (2004), *Web Services: Concepts, Architectures and Applications*, Vol. 1, Springer, Berlin.

Alves C., Castro J. (2001), CRE: A systematic method for COTS components selection, [in:] *XV Brazilian Symposium on Software Engineering (SBES)*, Rio de Janeiro, pp. 193-207.

Alves C., Finkelstein A. (2002a), Challenges in COTS decision-making: A goal-driven requirements engineering perspective, [in:] *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, ACM, Ischia*, ACM, New York, pp. 789-794.

Alves C., Finkelstein A. (2002b), Negotiating requirements for COTS-based systems, [in:] *Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, Essen, pp. 1-6.

Battle S. et al. (2005), *Semantic Web Services Framework (SWSF) Overview*, http://www.daml.org/services/swsf/1.0/overview/.

Bonati B., Regutzki J., Schroter M. (2006), *Enterprise Services Architecture for Financial Services*, SAP Press, Walldorf.

Carney D. (1999), Requirements and COTS-based systems: A thorny question indeed, *The COTS Spot*, Vol. 2, No. 2, pp. 1-7.

Carney D.J., Wallnau K.C. (1998), A basis for evaluation of commercial software, *Information and Software Technology*, Vol. 40, pp. 851-860.

Dobson G., Sawyer P. (2004), *Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web*, http://www.comp.lancs.ac.uk/~dobsong/papers/dre06_ontology_based_re.pdf.

Fensel D., Bussler C., Maedche A. (2002), Semantic web enabled web Services, [in:] *Proceedings of the First International Semantic Web Conference on The Semantic Web*, Springer-Verlag, London.

Fensel D. et al. (2007), *Enabling Semantic Web Services: The Web Service Modeling Ontology*, Vol. 1, Springer, Berlin.

Gonzalez-Castillo J., Trastour D., Bartolini C. (2001), Description logics for matchmaking of services, [in:] *KI-2001 Workshop on Applications of Description Logics*, Vienna, HP Laboratories Bristol, HPL-2001-265, http://www.hpl.hp.com/techreports/2001/HPL-2001-265.pdf, pp. 1-13.

Karakostas B., Zorgios Y. (2008), *Engineering Service Oriented Systems: A Model Driven Approach*, Idea Group Publishing.

Keller U. et al.(2004), *WSMO Web Service Discovery – WSML Working Draft (D5.1v0.1)*, DERI, Innsbruck.

Kontio J. (1995), *OTSO: A Systematic Process for Reusable Software Component Selection*, University of Maryland, College Park.

Lacy L.W. (2005), *OWL: Representing Information Using the Web Ontology Language*, Trafford Publishing, Crewe, Cheshire.

Lamsweerde A. v. (2001), Goal-oriented requirements engineering: A guided tour, [in:] *5th IEEE International Symposium on Requirements Engineering*, Toronto, pp. 1-14.

Leung H.K.N., Leung K.R.P.H. (2003), Domain-based COTS-product selection method, *Lecture Notes in Computer Science* (LNCS) 2693/2003, pp. 40-63.

Maiden N.A.M., Ncube C. (1998), Acquiring COTS software selection requirements, *Software, IEEE*, Vol. 15, No. 2, pp. 46-56.

Martin D. et al., *OWL-S: Semantic Markup for Web Services, W3C Member Submission*, http://www.w3.org/Submission/OWL-S/.

Mayank V., Kositsyna N., Austin M.A. (2004), *Requirements Engineering and the Semantic Web: Part II. Representation, Management and Validation of Requirements and System-Level Architectures*, University of Maryland, College Park.

Mielnik J.-C. et al. (2003), eCots Platform: An inter-industrial initiative for COTS-related information sharing, [in:] *Second International Conference on COTS-Based Software Systems*, Springer-Verlag.

Ncube C., Maiden N.A.M. (1999), PORE: Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm, [in:] *International Workshop on Component-Based Software Engineering*, Los Angeles, pp. 1-12.

North American Product Classification System (2009), http://www.census.gov/eos/www/napcs/napcs.htm (access date: 01-07-2009).

Oberndorf T., Brownsword L., Sledge C.A. (2000), *An Activity Framework for COTS-Based Systems*, Carnegie Mellon Software Engineering Institute, Pittsburgh.

Papazoglou M.P., Georgakopoulos D. (2003), Service-oriented computing – introduction, *Communications of the ACM*, Vol. 46, No. 10, pp. 24-28.

Papazoglou M.P., Heuvel W.-J. (2007), Service oriented architectures: Approaches, technologies and research issues, *The VLDB Journal*, Vol. 16, No. 3, pp. 389-415.

Pathak J. et al.(2005), A framework for semantic web services discovery, [in:] *Proceedings of the 7th annual ACM International Workshop on Web Information and Data Management*, ACM, Bremen, pp. 45-50.

Ramesh B., Jarke M. (2001), Toward reference models for requirements traceability, *IEEE Transactions on Software Engineering*, Vol. 27, No. 1, pp. 58-93.

Roman D., Lausen H., Keller U. (2006), *Web Service Modeling Ontology – WSMO Final Draft (D2v1.3)*, DERI, Innsbruck.

The Open Group – SOA Working Group (2007), *Service-Oriented Architecture (SOA) – Whitepaper*, San Francisco, CA.