A model of semantic web service in a distributed computer system

Nataliia Kulykovska[1[0000-0003-4691-5102]], Stepan Skrupsky[2[0000-0002-9437-9095]], Tetiana Diachuk[3[0000-0002-2478-0588]]

[1] "Zaporizhzhia Polytechnic" National University, Zhukovsky str., 64,Zaporizhzhia, 69063, Ukraine
natalya.gontar@gmail.com
[2] "Zaporizhzhia Polytechnic" National University, Zhukovsky str., 64,Zaporizhzhia, 69063, Ukraine
sskrupsky@gmail.com
[3] "Zaporizhzhia Polytechnic" National University, Zhukovsky str., 64,Zaporizhzhia, 69063, Ukraine
diacht@gmail.com

Abstract. One of the main tasks of any distributed computer system is the analysis of the properties of the obtained data, and their further use for logical reasoning. This is a difficult task due to the frequently encountered dynamics and heterogeneity of information. The aim of the work is to present a knowledge-based web service model. The web service integrates with the technologies of the semantic web in order to study the data, their processing and inference. The goal is achieved as follows: the meta-metamodel of the web service ontology and the metamodel of the semantic web service are structured using UML class diagrams and formalized using ALC description logic. The scientific novelty of the results is the presentation of a service description through an ontology. Through the introduction of ontologies, a transition is being made in data mining. In contrast to the application of standards: SOAP, WSDL, HTTP, the use of ontologies is argued that they contain structured information about the functional and non-functional characteristics of the service, and give flexibility to the description. When working with ontologies, there is a powerful logical apparatus for searching, combining and comparing ontologies. By introducing ontologies, knowledge of the web service is represented. The development of an ontology / web service pair gives an advantage over the SAWSDL, OWL-S standards, because the web service developer has the ability to create a more accurate semantic description of the service. We presented the language features and constructs of ontology of web service descriptions. Step by step, we created semantic description to describe web services. After learning how to use OWL to formally express web service semantics, we moved on to the issue of actually adding semantics to service descriptions. For the first time, formal models of service ontology and semantic web service are proposed.

Keywords: semantic web service, service oriented architecture, distributed systems, model, ontology, description logic

1 Introduction

Distributed computer systems (DS) based on a service-oriented architecture (SOA) are independent of development technologies and platforms, while applications running on one platform can call services running on other platforms in a standard way [1]. The main burden of performing computational operations with such an architecture rests with web services (WS) that solve all the problems of modeling the designed systems; client applications have only the simplest functions for preparing data and displaying simulation results.

Since WS can operate at a higher level of abstraction, analyzing and processing data types in a dynamic way, the individual software components are given the opportunity to interact more openly [2]. When using the universally described interfaces, it becomes possible to reuse software components, which reduces the complexity of the development of DS and correctly accumulate the data.

SOA is a heterogeneous environment, and it is developing towards creating a more structured set of solutions where WS should be represented in some unified way and equally detectable, able to communicate with other objects, and also be directly integrated with the Internet infrastructure and other services, independently from the functionality of the service [3].

Using semantic web technologies in the development of WS in the DS allows you to:

- go to the intellectualization of the description of XML-based interfaces and interactions, combining any type of application with another application, and providing freedom of change and development over time as long as the corresponding interface is supported;
- use a higher level of software abstraction and data description logic;
- take into account the weakness of the software, due to which the interaction between the applications of the service is not broken every time the design or implementation of a service changes;
- provide existing or legacy software service interface without changing the original applications;
- make decisions based on available data.

Key idea of semantic web services is to annotate web services with concepts which are defined in formal logic-based ontologies such that, from an artificial intelligence perspective, intelligent agents and service-based applications can actually reason on such formal service semantics. In contrast to web service descriptions, this may facilitate not only the semantic interoperation between services but their automated logic-based composition planning and a more precise service search [4].

The aim of the work is to present a knowledge-based web service model. The WS integrates with the technologies of the semantic web in order to study the data, their processing and inference. The scientific novelty of the results is the presentation of a service description through an ontology. Through the introduction of ontologies, a transition is being made in data mining.

2 Literature review

SOA is an approach for designing and developing DS [5]. By this approach, design, development, and implementation of DS are possible due to the web technology [6]. However, interoperability of services in SOA is not limited to web services [7, 8], the WS are the most suitable technology for successful SOA [9]. In DS based on SOA, information source and business functions can be converted into modular services units for control and management [10].

Web services are modular, self-describing, selfcontained applications that are accessible over the Internet [11]. WS is a software component invokable over the web via an XML [12] message that follows the SOAP [13] standard. The component provides one or more operations for performing useful actions on behalf of the invoking client. These operations and the formats of the input and output messages are described using WSDL [14].

Description of services in a language e neutral manner is vital for the widespread use of WS. For general usability, a service must be described and advertised. WSDL takes care of the description by providing a language to describe a service in enough detail to invoke any of its operations. Service providers describe their WS and advertise them in a universal registry called UDDI [15, 16]. This enables service requestors to search the registry and find services, which match their requirements. UDDI allows for the creation of registries that are accessible over the Web. A registry contains content from the WSDL descriptions as well as additional information such as data about the provider. Clients may use one or more registries to discover relevant services.

Prominent languages and formats for semantic service description are OWL-S (Web Ontology Language for Web Services) [17, 18, 19], WSML (Web Service Modeling Language) [20], the W3C standard SAWSDL (Semantic Annotations for WSDL and XML Schema) [21], USDL (Unified Service Description Language) [22, 23], Linked USDL [24], as well as the microformats hRESTS [25], SA-REST [26], and MicroWSMO. These description models mainly differ in their formal logic-based foundation and the possible extent of service annotation [27, 28, 29].

3 SOA

The SOA information model is registered with UDDI. Further, the client of the service can find it, call it, perform a certain task. The standards for exchanging information between services are SOAP, WSDL, UDDI, and a normal HTTP request. The SOAP and WSDL protocols provide a unified markup language for transmitted messages. It represents the functionality of a device connected to its managed resource. A user interface that provides a clear and standardized interface offers all the necessary functionalities for interacting with objects and related processes.

In order for SOA to enjoy greater success than it predecessors, it should consider the following attributes:

- scalable: The past solutions were not designed with the scale of the web in mind. SOA should work in a variety of settings, such as within an organization, between business partners and across the world [30];
- loosely-coupled: SOA is an evolution from tightly coupled systems to loosely coupled ones. Senders and receivers of a SOA should be independent of each other; the source can send the message independently of the target. Tight coupling is not suitable for SOA since it leads to monolithic and brittle distributed applications. Even trivial changes in one component lead to catastrophic breaks in function. Small changes in one application require matching changes in partner applications 31];
- interoperability: One party should be able to communicate with another party regardless of the machine they are running on;
- discovery: One party should be able to communicate with a second party selected from a set of competent candidates. Services need to be dynamically discoverable. This is accomplished through services such as a directory of service descriptions;
- abstraction: A SOA abstracts the underlying technology. Developers can concentrate on building services for business users rather than connecting systems and applications [32];
- standards: Interaction protocols must be standardized to ensure the widest interoperability among unrelated institutions. Contracts should also be standardized. Explicit contracts define what may be changed in an application without breaking the interaction. Furthermore, standards are the basis of interoperable contract selection and execution.

The current description of the interaction between WS is as follows [33]:

1. WS is created using some programming language.
2. WS is described by a WSDL document (normally generated by using a provided tool or by the built-in support of the development environment).
3. The service provider publishes the WS into the UDDI repository.
4. A web server hosts this WS by listening to HTTP traffic.
5. A client application (probably written in another programming language) searches the UDDI registry and discovers this service.
6. The client accesses the WSDL document, and a SOAP request message is generated based on the WSDL document.
7. The web server receives the SOAP request as part of a HTTP POST request, and it forwards this request to a WS request handler (a system-level application that is always running).
8. The WS request handler parses the SOAP message, invokes the right WS, and also creates the SOAP response. It finally sends the response to the web server.
9. The web server formulates a HTTP response, which includes the SOAP response, message and sends it back to the client.

4 Distributed knowledge-based systems

The semantic web should enable greater access not only to content but also to services on the web. Users and software agents should be able to discover, invoke, compose, and monitor web resources offering particular services and having particular properties, and should be able to do so with a high degree of automation if desired. Powerful tools should be enabled by service descriptions, across the web service lifecycle [34].

The semantic seb concept introduces formal definitions called ontologies, which allows you to build models of heterogeneous objects in a domain, share knowledge and support the automation of the formulation of logical conclusions from this knowledge.

To create a holistic system for deploying distributed knowledge-based systems (DKBS) semantic web services in a specific field of activity, it is necessary to develop a formal ontological model of objects (web services) that are part of SOA.

By combining the key elements of SOA with knowledge-based systems, you can get a formal DKBS model. Figure 1 shows the structural model of DKBS. We single out the concept of the main artifact of the system through the Element class, its subclasses are such DKBS elements as a service, system, event, human actor, semantic service [35, 36].

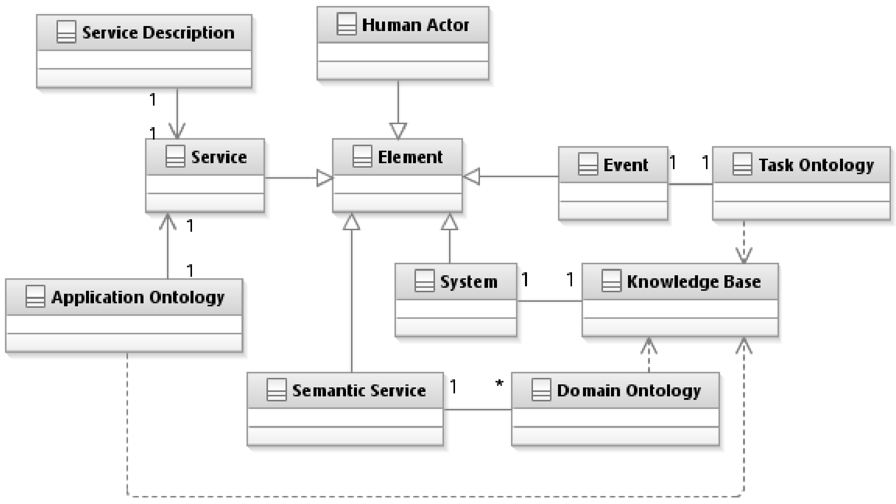

Fig. 1. Structural model of DKBS

The diagram shows that the service class consists of three elements: service description, application ontology, service interface. The semantic principle is reflected through such elements: domain ontology, application ontology, task ontology and knowledge base [37, 38].

A formal semantic service model should include:

1. Analysis of the structure of knowledge of the subject area of the DKBS, the main objects and relations:
   (a) the use of least-logical and ontological methods for the formation of the terminology of the system domain and knowledge structure;
   (b) structure of OWL ontology.
2. Repeated use of existing thesauruses, taxonomies, and ontologies of the DKBS domain:
   (a) semantic search for relevant objects and analysis of means of representing knowledge and standards in the DKBS;
   (b) a brief overview of relevant ontologies and other knowledge structures;
   (c) integration of existing taxonomies and domain ontologies.
3. The architecture of methods for the automated improvement of the formal ontological model of the DKBS:
   (a) the architecture of methods for the automated extraction of knowledge (terms and relationships) from natural language texts that relate to the DKBS;
   (b) methods of automated linguistic processing of natural language texts;
   (c) advanced OWL ontology.
4. Semantic search in the system based on the domain ontology:
   (a) semantic search for objects;
   (b) methods of semantic search for objects of the DKBS;
   (c) methods of semantic search for SOA services;
   (d) recommendations regarding the use of WS.

Search data for web services, their interactions, reviews, recommendations from service customers can be analyzed and converted into active knowledge, which allows us to better understand the physical world and create more value-added products and services.

DKBS is a synthesis of SOA and semantic technologies, which defines a service-oriented presentation of software and hardware components and a description of their formal semantics. Table 1 shows the architecture of the meta-modeling of the DKBS.

Table 1. The architecture of meta-modeling

Meta level	Simulation level
M3	Meta-metamodel / Meta-metamodel of service ontology
M2	Metamodel / Meta-model of service
M1	Data / Service
M0	Model / DKBS

The following should distinguish the basic concepts:

A web service is a software system identified by a unique web address (URL) with standardized interfaces.

Semantic web service (SWS) - complete elements of program logic with uniquely described semantics, accessible via the Internet and suitable for automated search, composition and execution, taking into account their semantics.

5 Semantic web service

A semantic meta-modeling of a WS is proposed (a meta-metamodel of a web service ontology, a meta-model of a WS) Thus, a new system object is formed - a SWS. In contrast to the application of standards: SOAP, WSDL, HTTP, the use of ontologies is argued that they contain structured information about the functional and non-functional characteristics of the service, and give flexibility to the description. When working with ontologies, there is a powerful logical apparatus for searching, combining and comparing ontologies. By introducing ontologies, knowledge is represented in the system. Developing an ontology / web service pair provides an advantage over SAWSDL, OWL-S, because a web service developer can create a more accurate description.

The meta-metamodel of the service ontology is shown in Figure 2, where Concepts are shown, i.e. units of information or data. Each concept has its own Logical Definition and a Textual Definition. In accordance with the concept, there may exist Instances, which, through the presentation of specific entities, connect abstract concepts with objects from the real world. In the same way, each Property of a concept has different Term - these are interpretation functions.

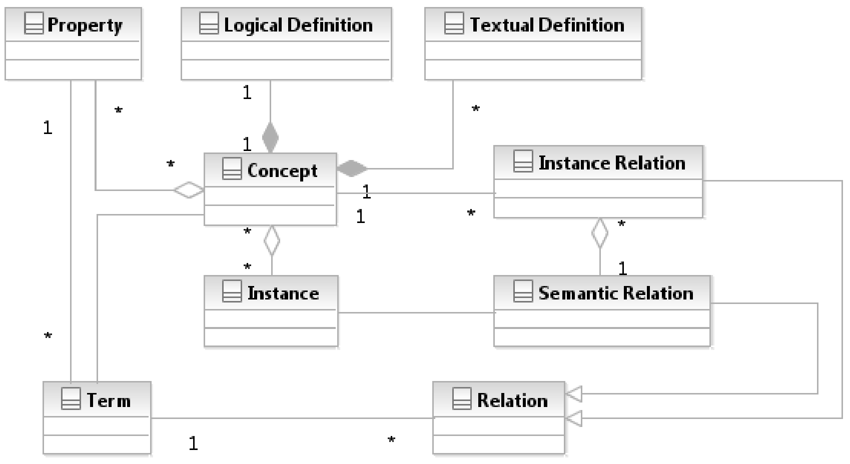

Fig. 2. Meta-metamodel of service ontology

Therefore, the meta-model of the service is its ontology, which describes all the characteristics of the service and its parameters (Fig. 3). A service primarily consists of Methods and Operations. The next component of the service is its Interface, which depends on already defined methods and operations. Binding is a component that provides the detailed information needed to access EndPoint. EndPoint is a specific service implementation where EndPoint associates a specific binding with a real address so that the service can be called.

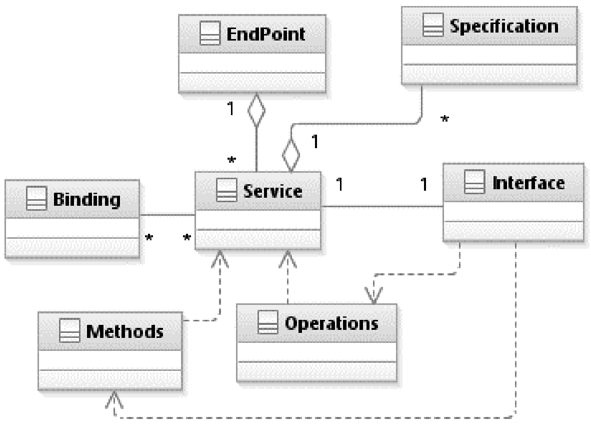

Fig. 3. Meta-model of the service

It is also important to understand the obvious difference between OWL and OWLS: OWL is an ontology language; it provides constructs and features we can use to create an ontology document. OWL-S, on the other hand, is just one such ontology created by using OWL.

Ontology is based on description logic (OWL-DL), which provides a platform for formal and machine-knowledge. To establish its syntax and semantics. Syntax, certain expressions (concepts, axioms, roles, etc.) are considered correctly constructed in this logic. How to interpret these expressions, i.e. gives them formal meaning.

Let $CN = \{A_1,..., A_m\}$ and $RN = \{R_1,..., R_n\}$ be finite nonempty sets of atomic concepts and atomic roles (also called concept names and role names). There are several dialects of descriptive logic that differ in expressive capabilities. The main operators of DL:

$\top$ – concept «THING» ;
$\bot$ – concept «NOTHING» ;
$\cap$ – logical connective (conjunction);
$\exists$ – the quantifier of existence;
$\forall$ – quantifier value limitation;
$\neg$ - logical connective (negation).

The syntax of DL, namely the ALC dialect, is a lot of concepts that are defined by an inductive definition:

- symbols $\top$ and $\bot$ - concepts (called truth and falsehood);
- - every atomic concept A is a concept;
- - if C is a concept, then $\neg C$ is a concept (called a complement to the concept);

— - if C and D are concepts, then $C \cap D$ and $C \cup D$ are concepts (intersection and union);
— - if C is a concept, and R is an atomic role, then $\exists R.C$ and $\forall R.C$ are concepts;
— - no other expressions are concepts.

In what follows, we will use a shorter notation to formulate the syntax. So, the syntax for the ALC logic concepts in this entry is as follows:

$$\perp \mid \top \mid \neg C \mid C \cap D \mid C \cup D \mid \exists R.C \mid \forall R.C \tag{1}$$

where A is an atomic concept, R is an atomic role, D are arbitrary concepts.

A complete description of the meta-metamodel of the service ontology in the formulations of the discription logic with the class diagram (Fig. 2) is given in Table 2.

Table 2. Meta-metamodel of service ontology in formulations of syntax description logic

Concept	Role
Property	isPartOf
LogicalDefinition	hasInstance
TextualDefinition	hasRelation
Concept	hasProperty
InstanceRelation	hasDefinition
Instance	hasObject
SemanticRelation	
Relation	
Term	

In this case, the values of the Concept column are atomic concepts, and the values of the Role column are atomic roles, then the concepts of ALC logic will be expressions:

$$\begin{array}{l} \text{Concept} \cap \text{Relation} \mid \text{Concept} \cap \text{Term} \mid \\ \forall \text{hasPartOf.Concept} \mid \text{Property} \cap \exists \text{hasProperty.Concept} \end{array} \tag{2}$$

The semantics of logic is defined using the concept of interpretation. Interpretation is a pair $I = (\Delta, \cdot^I)$, consisting of a nonempty set Δ, called the domain of this interpretation and the interpretive function $\cdot^I$, which is:

- each atomic concept $A \in CN$ is an arbitrary subset of $A^I \subseteq \Delta$;
- each atomic role $R \in RN$ is an arbitrary subset of $R^I \subseteq \Delta \times \Delta$.

For a service ontology, domain interpretation Δ will be the set of all ontology definitions. Atomic concepts are comparable to the set of all existing rules and terms for describing the ontology. We interpret atomic roles as two-place relations connecting all knowledge with a concept. Then everything listed in Formula 2 acquires the following semantics: a concept with this interpretation means:

- $\text{Concept} \cap \text{Relation}$ - a lot of concepts of interconnection (i.e., relationship);
- $\text{Concept} \cap \text{Term}$ - many concepts of terms (term);
- $\forall \text{hasPartOf.Concept}$ - many concepts that are part of this concept;
- $\text{Property} \cap \exists \text{hasProperty.Concept}$ - many properties that are property of the concept.

A full description of the semantic web service model in the formulations of the description logic is given in the table 2.

Table 3. Metamodel of SWS in formulations of syntax description logic

Concept		Role	
EndPoint	ServiceName	hasParameters	hasInteface
Specification	Definition	hasInstance	hasDescription
Binding	Parameters	hasInput	hasSpecification
Methods	InputParameters	hasOutput	endPoint
Operation	OutputParametes	hasResult	hasOperation
Interface	ContactInformation	hasPrecondition	hasMethods
Service	Value	hasCode	hasName
Taxonomy	Code	isPartOf	
Result	ServiceClassifacation	hasPelation	
Text	ServiceParametersName	hasObject	
Name			

Thus, the formation of ALC logic concepts for the SWS model is as follows:

$$\begin{gathered} Service \cap Methods \mid Definiton \cap \text{Text} \mid \\ \forall \text{hasParameters.Service} \mid \text{Value} \cap \exists \text{hasParameters.Service} \end{gathered} \quad (3)$$

The semantic interpretation of formula 3 is as follows:

- $\text{Service} \cap \text{Methods}$ - many service methods;
- $\text{Definiton} \cap \text{Text}$ - many text descriptions;
- $\forall \text{hasParameters.Service}$ - many concepts that are service parameters;
- $\text{Value} \cap \exists \text{hasParameters.Service}$ - many values that are service parameters.

Thus, a semantic web service is a pair of ontology and web service, which are described above by the presented models. Each formal model of descriptive logic is easily ported to OWL.

6 Processing knowledge of semantic web service

Formal models of ontology and semantic web service are aimed at supporting the process of automated deployment of intelligent applications in heterogeneous environments and allows:

- hide the technological heterogeneity that is characteristic of many heterogeneous web services;
- hide the semantic heterogeneity inherent in the used heterogeneous domain ontologies in order to semantically annotate the data of semantic web services.

The following is the process of finding the web services the user needs. In this case, the user can specify both the functions or properties of the service itself, as well as the characteristics associated with the end result of the functioning of the service.

At the same time, the user can apply the terms from any ontologies available to him that describe the DKBS space and the corresponding subject area (with reference to the ontologies themselves). But in the description of the semantic web service, terms from other ontologies related to close domains can be used.

Therefore, the problem of comparing ontologies arises. In the general case, this problem is extremely complex and laborious, but if there are a number of restrictions on the comparable ontologies, it can be solved in an acceptable time.

For the intellectualization of data and the processing of knowledge stored in ontologies, logical inference blocks (reasoners) are used, which play a key role in knowledge-oriented intelligent systems.

There are many implementations of logical inference processors (reasoning engine) for OWL ontologies that differ in capabilities, applications, and quality of task execution. The generalized analysis allows us to divide them into three groups depending on the implementation method:

1. Table Dl-processors. Traditionally, they were developed first to solve such problems. They have low performance, but are able to make conclusions on complex ontologies with many non-trivial constructions. This class includes resonators Pellet, RacerPro, FacT ++, as well as Hermit and SHER [41, 42, 43].
2. Disjunctive Datalog processors. They transform the ontology into a disjunctive Datalog program and use the deductive database technique and the resolution rule. Such processors have satisfactory performance when using some optimizations, but do not support certain OWL-designs, in particular cardinal restrictions and ratings. This group belongs to KAON2. [44].
3. Rules processors. Use rule processing systems for inference on ontologies. They have high performance, but can only process simple ontologies, devoid of many

important structures. Representatives of this group: Sesame / OWLIM, Jena, Owljesskb. [45].

There are two approaches to the implementation of inference: based on the rules (using the algorithms forwardchaining and / or backward-chaining) and based on the semantic scoreboard (semantic tableau). Based on the rules, Semantics, SDK and Owlim are implemented, and on the basis of the semantic scoreboard - Pellet.

Today, the Pellet system implements the most expressive descriptive logic using a high-performance logic inference (tableau-based algorithm), which is used to process ontologies described in the OWL DL language.

7 Conclusion

We presented the language features and constructs of ontology of WS descriptions. Step by step, we created semantic description to describe SWS. For instance, given that WSDL is also used to describe WS, understanding the relationship between WSDL and OWL is important for Semantic Web developers. We first built two models (structural and formal) to describe the ontology and web service. Examples of semantic interpretation of each model are given. Presented existing tools for further work with the obtained ontologies.

The advantages of using ontologies for WS:

1. Automatic discovery of SWS: Finding the desired service can be hard, especially when the service requester does not know of the existence of the service provided. However, to make SWS a real success, a way to discover the requested service should be provided; also, it has to be discovered automatically, with great accuracy and efficiency.
2. Automatic composition of the necessary services. Clearly, a software agent should be able to find all the necessary services and invoke them in the correct order to accomplish the business goal.

References

1. Ruzhi Xu, Peiguang Lin, Cheng Liu: Research on Distributed Knowledge Base System Architecture for Knowledge Sharing of Virtual Organization. Atlantis Press, pp. 349-357 (2010)
2. Fensel et. Al.: Implementing Semantic Web Services, Springer-Verlag Berlin Heidelberg (2008)
3. Portier B.: COA terminology overview, Part 1: Service, architecture, governance, and business terms (2007)
4. Klush, Matthias; Kapahnke, Patrick; Schulte, Stefan; Lecue, Freddy; Bernstein, Abraham (2016). Semantic web service search: a brief survey. Künstliche Intelligenz (KI), 30(2):139-147. DOI: https://doi.org/10.1007/s13218-015-0415-7
5. Zeppenfeld Klaus, Patrick Finger: SOA und WebServices, Springer-Verlag Berlin Heidelberg (2009)

6. Kulykovska, N. A.; Timenko, A., V; Ilyashenko, M. B.; Kirichek, G. G.: Distributed Knowledge Base System. PROBLEMELE ENERGETICII REGIONALE vol. 1-1 (40): 79-90 (2019)
7. Allemang D, Hendler J.: Semantic web for the working ontologist modeling in RDF, RDFS and OWL, Elsevier Inc. (2008)
8. Demirkan. H., Kauffman. R. J., Vayghan. J.A., Fill. H.G., Karagiannis. D., Maglio. P.: Service-oriented technology and management: Perspectives on research and practice for the coming decade. Electronic Commerce Research and Applications 7: 356–376 (2008)
9. Erl,T.: Service-Oriented Architecture Concepts , Technology , and Design ,Prentice Hall: 2-14 (2005)
10. Jack C.P. Cheng, Kincho H. Law, Hans Bjornsson, Albert Jones c, Ram Sriram: A service oriented framework for construction supply chain integration , Automation in Construction 19: 245–260 (2010).
11. Rudi Studer, Stephan Grimm, Andreas Abecker: Semantic Web Services, Springer-Verlag Berlin Heidelberg (2007)
12. Jeff Friesen: Java XML and JSON, Apress (2016)
13. Helen Hye-Young Paik, Angel Lagares Lemos, Moshe Chai Barukh, Boualem Benatallah, Aarthi Natarajan: Web Service Implementation and Composition Techniques, Springer International Publishing (2017)
14. Liang-Jie Zhang: Web Services, Springer-Verlag Berlin Heidelberg (2004)
15. James McGovern, Oliver Sims,nAshish Jain, Mark Little: Enterprise Service Oriented Architectures, Springer Netherlands (2006)
16. Klaus Zeppenfeld, Patrick Finger: SOA und WebServices, Springer-Verlag Berlin Heidelberg (2009)
17. Guang Chen, Tonghai Jiang, Meng Wang, Xinyu Tang, Wenfei Ji: Modeling and reasoning of IoT architecture in semantic ontology dimension. Computer Communications Volume 153: 580-594 (2020)
18. Matthias Klusch, Patrick Kapahnke, Stefan Schulte, Freddy Lecue & Abraham Bernstein: Semantic Web Service Search: A Brief Survey, KI - Künstliche Intelligenz vol. 30: 139–147 (2016)
19. 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.: OWL-S: Semantic Markup for Web Services. (2004) www.w3.org/Submission/OWL-S/
20. Fensel, D.; Lausen, H.; Polleres, A: Enabling Semantic Web Services, Springer (2010)
21. Farrell, J.; Lausen, H.: Semantic Annotations for WSDL and XML Schema. (2007) www.w3.org/TR/sawsdl/
22. Kona, S.; Bansal, A.; Simon, L.; Mallya, A.; Gupta, G.; Hite, T.D.: USDL: A Service-Semantics Description Language for Automatic Service Discovery and Composition, Web Services Research (2009)
23. Lampe, U.; Schulte, S;: Self-Adaptive Semantic Matchmaking Using COV4SWS.KOM and LOG4SWS.KOM. In: B. Blake et al. (eds.) Semantic Web Services, Chapter 9, Springer (2012)
24. Pedrinaci, C.; Leidig, T.: Linked USDL Core (2011) www.linked-usdl.org/ns/usdl-core
25. Klusch, M.: The S3 Contest: Performance Evaluation of Semantic Service Matchmakers. In: Blake, M.B.; Cabral, L.; Koenig-Ries, B.; Kuester, U.; Martin, D. (Eds.): Semantic Web Services: Advancement through Evaluation; Springer (2012)
26. Gomadam, K.; Ranabahu, A.; Sheth, A.: SA-REST: Semantic Annotation of Web Resources. (2010) www.w3.org/Submission/2010/SUBM-SA-REST20100405/
27. Klusch, M.: Semantic Web Service Coordination. In: [56], Chapter 4. (2008)

28. Baader F. (editor), et al.: The Description Logic Handbook,Cambridge University Press (2003)
29. Gandon F.: Ontology engineering: A Survey and a Return of experience, INSTITUT DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (2002)
30. Wooldridge, M.J., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2): 115–152 (1995)
31. Gontar N. A., Kudermetov R. K.: Working out ontology of systems engineering of space system, Radio Electronics, Computer Science, Control 2: 131-137 (2011)
32. Kirichek G., Harkusha V., Timenko A., Kulykovska N. System for detecting network anomalies using a hybrid of an uncontrolled and controlled neural network. CS&SE@SW 2019: 2nd Student Workshop on Computer Science and Software Engineering : 138 – 148 (2019)
33. Cardoso J, Hepp M., Lytras M. (Eds.): The Semantic Web. Real-World Applications from Industry(2008)
34. Cardoso Jorge and Amit P. Sheth.: Semantic Web Services, Processes and Applications (2006)
35. Kulykovska N.A, Timenko A.V.: A Structure of Semantic Service in a Distributed Knowledge Based System. Computer Modeling and Intelligent Systems: Proceedings of the Second International Workshop on Computer Modeling and Intelligent Systems (CMIS-2019), In: CEUR Workshop Proceedings 2353, 533-544 (2019)
36. Kirichek, G., Tymoshenko, V., Rudkovskyi, O., Hrushko, S.: Decentralized System for Run Services. In: Second International Workshop on Computer Modeling and Intelli-gent Systems (CMIS-2019). In: CEUR Workshop Proceedings 2353, 860–872 (2019)
37. Allemang D.:P Semantic web for the working ontologist. Modeling in RDF, RDFS and OWL, Morgan Kaufmann Publishers (2008).
38. Yu, Liyang.: Introduction to Semantic Web and Semantic Web services, Springer (2007)
39. Leuf, Bo.: Technology Analyst, Sweden. The Semantic Web: crafting infrastructure for agency (2006)
40. Medjahed B. and Bouguettaya A.: Service Composition for the Semantic Web, Springer Science+Business Media (2011)
41. Cary Pennington: Introduction to Web Services, Computer Science (2007) DOI:10.4018/978-1-59904-045-5.ch007
42. Kishore, Rajiv, Ramesh, Ram (Eds.): Ontologies. A Handbook of Principles, Concepts and Applications in Information Systems, Springer US (2007)
43. Wang, H., Horridge, M., Rector, A.L., Drummond, N., Seidenberg, J.: Debugging OWL-DL Ontologies: A Heuristic Approach. vol. 3729: 745–757 (2005)
44. Ebrahimipour, Vahid, Yacout, Soumaya (Eds.): Ontology Modeling in Physical Asset Integrity Management, Springer International Publishing (2015)
45. Tamma, V., Dragoni, M., Goncalves, R.S., Ławrynowicz, A. (Eds.): Ontology Engineering. 12th International Experiences and Directions Workshop on OWL, OWLED 2015 (2015)