Interoperability Description of Web Services Based Application Servers

Pawel L. Kaczmarek

Interoperability Description of Web Services Based Application Servers

Paweł L. Kaczmarek

Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology
Narutowicza 11/12, 80-233 Gdańsk, Poland
pkacz@eti.pg.gda.pl
http://www.eti.pg.gda.pl

Abstract. Web services standards were designed to enable interoperability of heterogeneous application servers in the Service Oriented Architecture. Although the standards proved to be highly successful, there are still difficulties in effective services integration. The paper presents a methodology that enables description of application servers interoperability in order to improve the service integration process. The methodology proposes a systematic classification of Web services standards, versions and configuration options, and uses the classification for interoperability rating. Concrete integrations are rated by developers that specify integration scope, configuration complexity and required expertise level. The methodology was implemented in a web system that enables definition of standards and configuration options as well as rating of integrations. As a part of the research, interoperability experiments were executed and registered in the system.

Keywords: Interoperability, Distributed programming, Web services, System integration and implementation

1 Introduction

Service Oriented Architecture (SOA) assumes that complex systems are developed by integration of existing services, which improves development cost and time. The approach, however, requires resolution of interoperability issues, as services are run in heterogeneous environments and supplying potentially incompatible interfaces. Web services (WS) [3] standards were designed to improve interoperability and were widely adopted in the industry environment. The WS technology covers now over fifty standards and standard like concepts [9] with SOAP and WSDL as basic ones. Specialized WS standards (WS-*) have been published concerning, among others: security, reliability and transactions.

Despite the general success of Web services, there are still difficulties in effective integration of components. There exist a proliferation of standards, their version and alternative configuration options anticipated by standards. The options enable high application configurability, but raise interoperability difficulties

in case of configuration mismatches. Additionally, software vendors add product specific extensions [4] as a result of existence of open points in standards or intent to lock user into a proprietary technology.

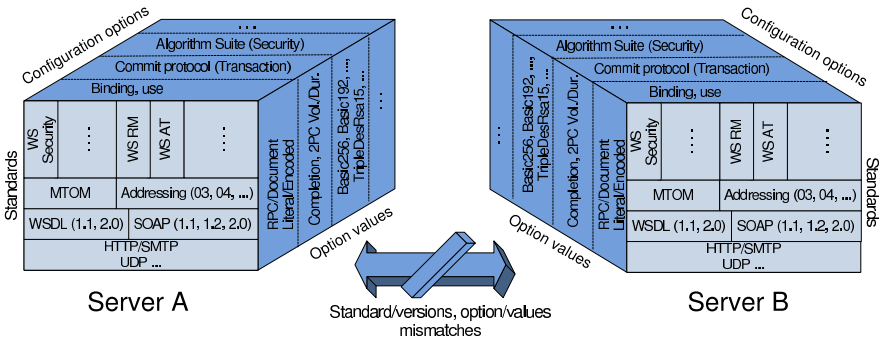

Fig. 1. General cooperation model between heterogeneous application servers

The WS Interoperability (WS-I) Organization [16] has been established to resolve imprecisions in WS-* specifications. The organization has issued additional interoperability profiles that improve interoperability by giving detailed guidelines and restrictions for data interchange. Interoperability profiles, however, cover only selected areas of the wide WS-* technology.

Although significant improvements in WS-* interoperability have been made, developers are still faced with problems in integration of heterogeneous services:

- Application servers, that host services, may implement a different selection of standards or vendor-specific extensions, which results in interface incompatibilities (see Fig. 1).
- High development skills are required in the field of standards, application servers and their configuration options to integrate services effectively. A non-expert developer may fail to integrate services in a concrete case because of insufficient knowledge, although the integration is objectively possible.

Considering the existing difficulties, it is necessary to design a methodology that will enable description of application servers interoperability and, additionally, to perform experimental interoperability verification of concrete configurations. The paper presents a methodology that proposes a systematic classification of WS-* concepts, and uses the classification in experimental verification of interoperability.

Using the methodology, concrete integrations of application servers are described by specifying used standards, versions and configuration options. Then, integrations are rated concerning their scope (success rate), complexity and required development effort. The information is supplied by developers for developers. The supplying developer has already attempted to integrate servers, either

successfully or unsuccessfully, and registers the ratings. The consuming developer uses the information to improve the development process by refraining from configurations that are known to fail and applying those that are known to be relatively simple and successful. The benefit of the methodology is simplification of service integration by supplying information about development simplicity and interoperability of application servers in different configurations, and consequently components deployed on those servers. This reduces the workload as developers do not have to perform a detailed verification of various configurations.

The designed methodology was implemented as a web application that is available for use and loaded with information about servers and standards.

The rest of the paper is organized as follows. The next section presents a classification of WS-* concepts: standards, their categories, configuration options and option values. Sect. 3 presents experience-based rating of interoperability using the classification of WS-* concepts. Sect. 4 presents implementation work. Sect. 5 overviews existing solutions in WS-* interoperability and methods of interoperability rating. Finally, Sect. 6 concludes the paper.

2 Classification of WS-* Concepts

A systematic description of existing concepts for the Web services-based technology is the first step necessary for interoperability analysis of application servers. We focus on common entities that are encountered in practice in system development and are identified in [1] [9]. We distinguish the following main concepts to perform the classification:

- **Standard** represents a technical or de facto standard used in WS application integration. Each standard is described by a name, version, issuing organization. Let STD denote the set of considered standards and $std \in STD$ denote a single standard.
- **Standard version** represents versions of standards. It is assumed that at least one version is defined for each standard. Let VER, ver denote the set of considered versions and a single version, respectively.
- **Option** describes any alternative in configuration. They may be assigned to a version of a standard or may be standard independent, in which case are referred to as of application scope. Let OPT, opt denote the set of options and a single option, respectively.
- **Option value** Each option has an associated list of concrete option values. Values are of an enumerate type and depend on the option. Let VAL, val denote available values and a single value, respectively.

Definition 1. *Versions of standard (STD_VER) is a relation between STD and VER: $STD\_VER \subset STD \times VER$ such that $STD\_VER(std_i, ver_j)$ is standard std_i in version ver_j.*

Definition 2. *Options of standard version (STD_VER_OPT) is a relation between STD_VER and OPT: $STD\_VER\_OPT \subset STD\_VER \times OPT$ such that if $STD\_VER\_OPT(std_i, ver_j, opt_k)$, then standard std_i in version ver_j anticipates a configuration option opt_k.*

Definition 3. *Values for options (OPT_VAL) is a relation between OPT and VAL: $OPT\_VAL \subset OPT \times VAL$ such that if $OPT\_VAL(opt_i, val_j)$, then opt_i accepts val_j for its value.*

It is additionally specified, whether an option is required or not and may be neglected. Options may be shared between standards. Typically, different versions of the same standard define similar options. Fig 2 shows as part the described concepts and their relationships.

Table 1 shows exemplary options and their alternative values. Lists of (uncategorized) Web services options may be found in [1] [3], while [2] presents general options of applications. Naturally, some of the defined options may be insignificant in the context of a particular application.

Table 1. Exemplary options and option values of Web services standards

Configuration option	Alternative values	Description	Concerned standards or application
Transport protocol	HTTP, JMS, Jabber, SMTP/POP3, TCP	Specifies low-level transport protocol (typically HTTP)	SOAP
Protection Order	EncryptBeforeSigning, SignBeforeEncrypting	Indicates the order in which integrity and confidentiality are applied to the message	WS-Security Policy
Transaction commit protocol	Completion, Two-Phase Commit (Volatile), Two-Phase Commit (Durable)	Determines the behavior of coordinators and participants when presented with protocol messages or internal events	WS-Atomic Transaction
Floating point data	integer, floating point	Specifies whether floating point data is used	Application scope

Additionally, we use the following concepts that concern the execution environment:

- **Application server** represents a runtime platform that hosts applications. Let AS, as denote the set of servers and a single server, respectively.
- **Related library** supplies additional software modules for the server such as: communication libraries, IDE tools or lower layers runtime platforms. Let RL, rl denote the set of related libraries and a single library, respectively.

Definition 4. *Has related library(HAS_RL) is a relation between AS and RL: $HAS\_RL \subset AS \times RL$ such that if $HAS\_RL(as_i, rl_j)$ then as_i is integrated with rl_j.*

3 Description of Experience-Based Rating of Interoperability

Typically, application servers supply necessary communication libraries and runtime environments that implement WS-* standards, while concrete services are deployed on the servers and communicate through the WS interface. The rating is performed for concrete integrations (denoted *integr*) that are specified by the following elements:

1. The two application servers that are integrated. They represent runtime platforms that host applications (in this case WS accessible services). Related libraries may be specified, which covers additional software such as: WS-* communication libraries, IDE tools or lower layer runtime platforms.
2. The configuration (denoted $config$) in which the two application servers interoperate. The configuration specifies the following elements:
   - Standards and standard versions. A configuration typically uses many standards that supply a compound functionality.
   - Configuration options and selection of option values. Analogously to standards selection, an integration may specify many options and their values. If an option is not specified, the option is assumed as insignificant.

Different integrations may be defined for two application servers representing different selection of the concepts.

Definition 5. *Configuration $CONFIG$ is a relation between STD_VER and OPT_VAL, such that $CONFIG \subset STD\_VER^n \times OPT\_VAL^m$.*

Definition 6. *Integration ($INTEGR$) is a relation between AS, RL and $CONFIG$ defined as follows:*

- $INTEGR \subset AS \times RL^p \times AS \times RL^r \times CONFIG$, *where*
- *for each* $int_i \in INTEGR$ *there is* $int_i = (as_{i1}, RL_{i1}, as_{i2}, RL_{i2}, conf_i)$ *and the following conditions are met:*
  1. $as_{i1} \in AS$ *(server − side),* $as_{i2} \in AS$ *(client − side)*
  2. $RL_{i1} \subseteq RL$ *and* $\forall rl \in RL_{i1} : rl \in HAS\_RL(as_{i1})$
  3. $RL_{i2} \subseteq RL$ *and* $\forall rl \in RL_{i2} : rl \in HAS\_RL(as_{i2})$
  4. $conf_i \in CONFIG$

Typically, interoperability rating methods define a single metric that specifies the level of interoperability [5]. We extend the approach and propose the following attributes:

- Scope. The developer specifies the level of reached interoperability, whether the integration is fully or partially functional, or failed.
- Simplicity. The developer specified the level of attempted complexity, which indicates the type of configuration and development efforts made. It is not reasonable to assume that an integration is impossible just because a developer failed to establish it, as she/he may have made a programming mistake or lacks knowledge regarding the technology.

– Helper attributes. The attributes may specify other issues such as QoS including performance, documentation, reliability.

Table 2 shows some of the proposed attributes and their description.

Table 2. Concerned interoperability attributes

Interoperability attribute / Description	Value-meaning
Scope The ability to exchange information (on the protocol compatibility level) Range [0 .. 5]	0 - none, no integration 1 - uncertain, system is unstable and may fail in some conditions 2 - limited, works, but imposes some limitations 3 - non-standard, works, but does not conform to specifications or presents low QoS (e.g. performance) 4 - operational, works, but there are known internal problems (e.g. a system handles a thrown exception) 5 - complete, works without known problems
Simplicity (¬ Complexity) The difficulty of necessary configuration and development work to establish the integration Range [0 .. 5]	0 - low, requiring changes of undocumented features (high configuration difficulty) 1 - medium – , custom workarounds required or major configuration of non-standard features 2 - medium, minor configuration of non-standard features (custom changes) 3 - medium + , major configuration of standard features 4 - high – , minor changes of non-GUI available options or use of advanced GUI features 5 - high, GUI changes of standard options
Connector / mediator module Range [0, 5]	0 - yes, required 5 - no, not required
Documentation The amount of available doc Range [0, 3, 5]	0 - no appropriate documentation 3 - general documentation 5 - detailed description of integration

Interoperability in each integration is rated by developers that have already attempted to establish it. The rating is organized analogously to most existing interoperability rating methods [5], that is, natural numbers are assigned for different levels. We use the 0 - 5 rates with 0 denoting the worst result (the least desired situation) and 5 denoting the the best result (the most desired situation) for an attribute. Fig.2 shows the information structure that describes application servers, integrations and ratings.

Although user experience seems an appropriate form of rating, it may happen that unreliable opinions will occur. Therefore, the method implements mechanisms to increase rating reliability: user roles based on reliability level, user expertise level and multiple opinion gathering. Three user reliability levels are distinguished [0..2]: guests (0), registered users (1) and trusted users (2). A guest is an unknown, unregistered person. A registered user may add ratings, define

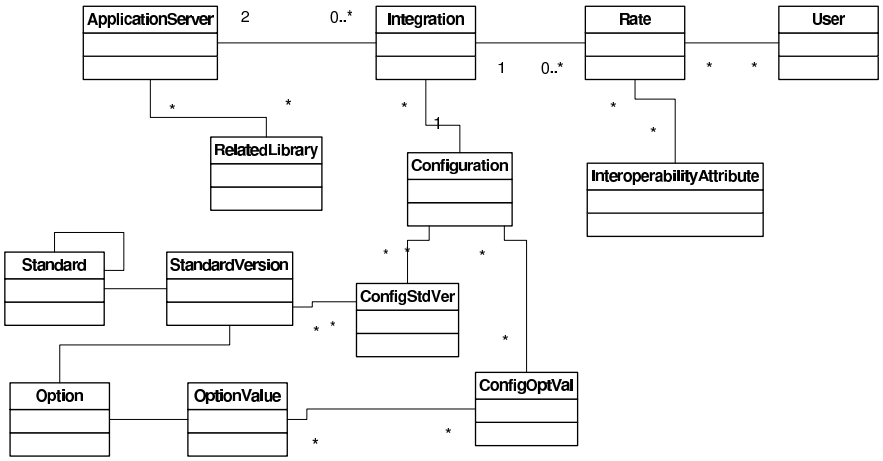

Fig. 2. Information structure of integration rating

servers and WS-* concepts. A trusted user is a registered user that has been verified for its identity. Additionally, all registered users should specify their expertise level using the following scale [0..3]: novice (0) - have studied a technology theoretically or worked with manuals, semi-advanced (1) - worked approximately 6 months in a technology, advanced (2) - worked more than 6 months in a technology, expert (3) - knows undocumented features of a technology.

As a simple example, consider a .NET v 3.5 / Development Serever client that invokes a Web service running on Glassfish v3 / JavaEE6 / NetBeans IDE 6.8 using WS-ReliableMessaging. Two detailed configurations are analyzed: (i) without exact ordering of messages and (ii) with the ordering. The ordering may be specified using the NetBeans GUI "Deliver Messages In Exact Order" option. The first configuration works correctly and the invocation is successful. The interoperability is rated as complete (5) and the simplicity is high (5). The second configuration (with message ordering), however, results in an error in a typical case. If the option is enabled, the .NET environment generates error "Could not find default endpoint element that references contract". A negative rating (0) will be given to the integration, the simplicity remains high (5). In a more advanced scenario, the developer may workaround the problem and modify manually the web.config file and specifying ordered="true" in the reliableSession node. The procedure is required because of a GUI failure, which is definitely a non-standard behavior. The configuration is rated as: complete (5) and medium simplicity (2).

4 Implementation and Evaluation

The designed methodology was implemented in the Application Servers Interoperability system that supplies the following functionality:

- Configuration of information about WS-* concepts: standards, options and values together with their relations,
- Definition of interoperability attributes and ratings,
- Management of information about experience based rating of application servers integrations,
- User registration, user data management and role-based functionality access,
- Viewing information about ratings.

The system is available online at:
http://www.as-interoperability.eti.pg.gda.pl

Information management covers typical operations of adding, deleting and modifying data concerning WS-* standards and application servers. Application servers and standards are described giving their name, version, vendor and descriptive information. The system enables user management including user registration and role assignment. The system is designed in a typical layered architecture, covering the database and data access layers, the web services layer and the web layer that supplies the end user interface.

The system is implemented in the C# language in .NET Framework 3.5 and uses the MS SQL 2005 database server to store information. The running system is deployed on Microsoft IIS server. WCF API for Web services was used to implement communication. The system also uses a few third party packages including: log4net, AjaxControlToolkit, LinqKit.

4.1 System Usability

AS-Interoperability supplies information that aims at improving the development process of new systems, and reduce development cost and time. During system design, a developer uses interoperability information to verify if considered application servers are capable of cooperating in concrete cases. The information is vital for making design decisions and selection of appropriate servers to host the system.

Interoperability information may be gathered using both academic, industrial and open community sources. Currently, registered data were collected by cooperation with students and academics. Community related cooperation is expected to supply widest range of results. This kind of cooperation has been successfully applied in many open-source projects. Also, existing results of test cases are an important source of interoperability information for the system.

Fig. 3 shows an exemplary screenshot of the system.

4.2 Exemplary Test Results

We recorded selected information about standards, servers and interoperability ratings that had been performed during our research as presented in [11]. Test results consider, amongst others, the following application servers: JBoss 5.1.0.GA, Apache Geronimo 2.2.0, Apache Axis2 1.5.1 hosted on Jetty or Tomcat web containers, IBM WebSphere CE 2.1.1.3 and Microsoft IIS 5.1 with

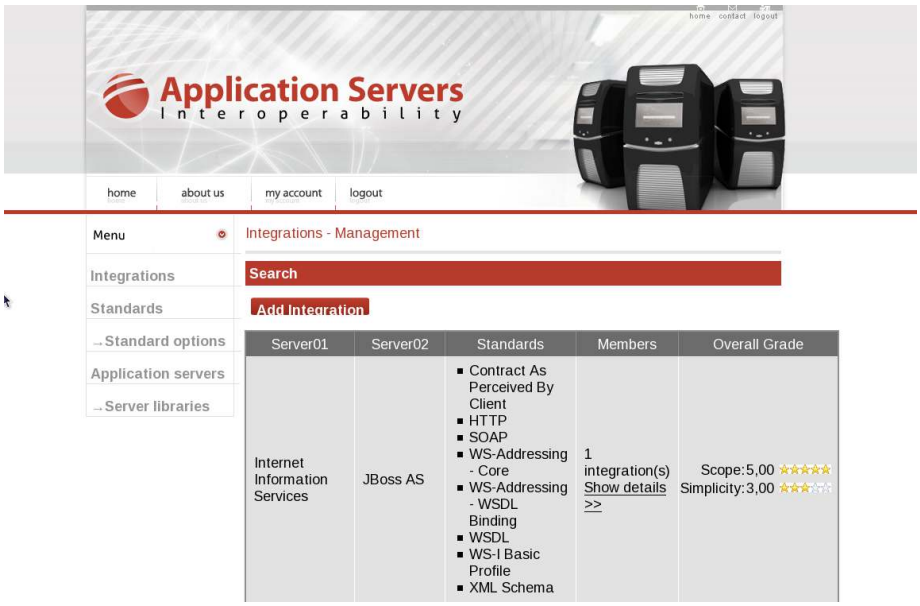

Fig. 3. Exemplary screenshot of the system for interoperability description

WCF 4.0. Performed experiments covered selected configurations in the standards: SOAP, WSDL, WS-Addressing, WS-Reliable Messaging, WS-Policy, WS-AtomicTransactions and WS-I BasicProfile conformance. Additionally, runtime environments of the Enterprise Service Bus architecture were tested, including: Mule ESB, Sun Open ESB and Micosoft WWF.

Generally, application server present high interoperability in basic WS-* standards, such as SOAP and WSDL. The standards are well established and implemented by virtually all servers in different versions, which gives background for effective integration. The integration, however, face various difficulties if extended standards are used.

As a part of the research, we performed experiments of transactional Web services invocation between the .NET and Java based servers. The experiments were performed by an experienced developer up to simplicity level 2 and 3 (major or non-standard configuration changes). Generally, integration of homogeneous servers was successful. We consider homogeneous servers as two instances of the same product, such as two IBM WebSphere AS 7.5 installations or two Microsoft IIS/WCF installations. However, the developer was not able to establish transactional integration of heterogeneous servers.

Integration of Sun Glassfish and Microsoft WWF failed because of difficulties in the WS-Coordination protocol in both client-server directions. Experiments covered major configuration changes (not supported by GUI interfaces) and installation of additional packages, which gives simplicity level 2. The developer was not able to successfully repeat interoperability tests that are supplied by

server vendors. The developer was also not able to establish transactional Web services integration between Microsoft .NET/WWF and IBM WebSphere AS. The experiment covered modification of advanced options available in GUI interfaces, which gives simplicity level 3.

Another group of registered results concern Workflow and Enterprise Service Bus runtime platforms as WS related technologies [12]. The workflow engines were used to invoke services supplied by the application servers in selected configurations.

Sun Open ESB and Microsoft IIS integration works correctly, although there exist difficulties in WSDL compatibility during development. The difficulties can be overcome by minor manual modification of the WSDL file (which gives simplicity level 3). If, however, the service is not compatible with WS-I Basic Profile, there are errors during execution and further adjustments of the WSDL file are necessary regarding binding and portType (which gives simplicity level 2). Similar difficulties are encountered during integration of OpenESB and Apache Axis2 independently from the used SOAP version. The integration is possible, but requires simplicity level of 2. Integration of OpenESB and Glassfish works straightforwardly giving simplicity level 5. The case should be considered actually as an integration of homogeneous environments as both servers are developed by the same community. Generally, workflow/ESB systems enable integration of heterogeneous servers, but detailed knowledge is necessary to identify and repair non-standard failures during development and operation.

5 Related Work

Interoperability has been researched for decades as an essential aspect of information systems. Although researched thoroughly, there exist open issues in computer systems interoperability due to changes in communication technology and design approaches.

Several methods and metrics for interoperability rating have been proposed. The work [5] presents an overview of approximately fifteen methods of interoperability rating and about thirty interoperability definitions. Most rating methods describe interoperability by defining levels of interoperability that depend on the scope of integration. General purpose methods include, among others: SoIM, LISI, LCI. Existing approaches usually address a wide range of issues including: syntax, semantics, data compatibility or resource sharing. Tools that simplify interoperability analysis have been proposed for various design levels ranging from the protocol level [2] to the enterprise architecture level[15]. [2] describes interoperability attributes of commercial-off-the-shelf (COTS) components in software development. The attributes are classified in four categories: general, interface, internal assumption, and dependency. Authors also present an assessment tool that performs analysis of potential integration mismatches. [15] presents a model for enterprise service interoperability analysis. The model is based on factors and factor dependencies that influence interoperability, such as service description, orchestration language, or semantic compatibility. Design

time and runtime interoperability are explicitly distinguished. This work differs in that it focuses on detailed analysis of service integration using WS, which concerns classification of WS-* concepts and multi-attribute rating. The analysis may be located within level 1 (Connected) in the LISI metric or level 2 (Data/Object) in the LCI metric [5].

SOA and WS-* specific issues in interoperability have been researched by both academic and industrial bodies [10] [16] [13]. Books and tutorials discuss integration techniques of .NET and Java based systems [14]. Additionally, application server vendors supply their interoperability guidelines and specification of implemented standards [9]. [4] discusses the problem of systems that are incompatible despite their standard-compliance. The work focuses on the process of standard design and implementation, explicitly refraining from interoperability which has a malevolent background. Despite the proliferation of SOA interoperability research, there is no work known to the author that presents a general methodology of systematic classification of WS-* related concepts and rating of interoperability attributes.

In some cases, test suits for standards are supplied by the issuing bodies, for example SOAP 1.2 tests [8] and WS-I standards tests [7]. Typically, the tests do not present detailed description of the source code used and required configuration changes, which makes it difficult to reproduce the test, especially by a non-expert developer. [6] presents a general method of conformance testing of parallel languages. This work does not intend to execute comprehensive tests of application servers or standards, but rather supplies a methodology for systematic description of test results, and selection of application servers and standards.

6 Conclusions and Future Work

The paper presented a methodology for description of application servers interoperability in various configurations of WS-* standards. The methodology was implemented in a web system that enables interchange of information concerning WS-* interoperability including information about implemented standards and configuration options by application servers. Developers benefit from the system by using the available information to make more effective design decisions concerning selection of application servers and standards.

The main scope of future work is to load the system with information concerning application servers interoperability. It is planed to engage a community of software developers and system integrators that will share their experiences. Using the registered information, general characteristics of interoperability may be derived on various levels of abstraction. The characteristics may further be used for assessment of WS-* runtime platforms.

Acknowledgments. This work was supported in part by the Polish Ministry of Science and Higher Education under research project N N519 172337. The author thanks students from the Faculty of ETI for their development work on

the system for WS-* interoperability description and performing various interoperability tests.

References

1. Web Services Stack Comparison, Apache. http://wiki.apache.org/ws/StackComparison (2010)
2. Bhuta, J., Boehm, B.: Attribute-based cots product interoperability assessment. In: Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems. pp. 163–171 (2007)
3. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Services Architecture, Working Group Note. W3C, http://www.w3.org/TR/ws-arch/ (2004)
4. Egyedi, T.M.: Standard-compliant, but incompatible?! Computer Standards & Interfaces 29(6), 605–613 (2007)
5. Ford, T., Colombi, J., Graham, S., Jacques, D.: A survey on interoperability measurement. In: 12th ICCRTS Adapting C2 to the 21st Century
6. Garstecki, Ł., Kaczmarek, P., Krawczyk, H., Wiszniewski, B., de Kargommeaux, J.C.: Testing for conformance of parallel programming pattern languages. Intr. Conf. on Parallel Processing and Applied Mathematics, LNCS 2328, 323–330 (2006)
7. Greene, S., Lauzon, D., Stobie, K.: Basic Profile 1.1 Test Assertions Version 1.1, Final Material. Web Services Interoperability Consortium (2005)
8. Hurley, O., Haas, H., Karmarkar, A., Mischkinsky, J., Thompson, L., Martin, R., Jones, M.: SOAP Version 1.2 Specification Assertions and Test Collection (Second Edition). http://www.w3.org/TR/2007/REC-soap12-testcollection-20070427/ (2007)
9. IBM, http://www.ibm.com/developerworks/webservices/standards/: Standards and Web services (2010)
10. International Federation for Information Processing (IFIP), Johannes Kepler University Linz: TC5 - Information Technology Applications Work Group 5.8 Enterprise Interoperability
11. Kaczmarek, P.L., Nowakowski, M.: A developer's view of application servers interoperability. 9th Intr. Conf. on Parallel Processing and Applied Mathematics, Part II, LNCS 7204 (in print) (2011)
12. Kaczmarek, P.L., Wierzbowski, P.: Dependable integration of esb and web services systems (in polish). In: 6. Konferencja Technologie Informacyjne. Poland (2008)
13. Microsoft Corporation, http://www.microsoft.com/interop/: Microsoft Interoperability (2010)
14. Moroney, L., Lai, R., Fisher, M.: Interoperability between java ee technology and .net applications. In: JavaOne Conference (2006)
15. Ullberg, J., Lagerström, R., Johnson, P.: A framework for service interoperability analysis using enterprise architecture models. In: IEEE SCC (2). pp. 99–107 (2008)
16. Web Services Interoperability Consortium: Interoperability: Ensuring the Success of Web Services (2004)