Evaluation of WS-* Standards Based Interoperability of SOA Products for the Hungarian e-Government Infrastructure

Balázs Simon, Zoltán László, Balázs Goldschmidt, Károly Kondorosi, Péter Risztics, and László Bacsa

Budapest University of Technology and Economics,
Department of Control Engineering and Information Technology
Magyar tudósok körútja 2,
1117 Budapest, Hungary
{sbalazs,laszlo,balage,kondor}@iit.bme.hu,
{risztics,bacsa}@ik.bme.hu

Abstract. The proposed architecture of the Hungarian e-Government Framework, mandating the functional co-operation of independent organizations, puts special emphasis on interoperability. WS-* standards have been created to reach uniformity and interoperability in the common middleware tasks for web services such as security, reliable messaging and transactions. These standards, however, while existing for some time, have implementations slightly different in quality. In order to assess implementations, thorough tests should be performed, and relevant test cases ought to be accepted. For selecting mature SOA products for e-Government application, a methodology of such an assessment is needed. We have defined a flexible and extensible test bed and a set of test cases for SOA products considering three aspects: compliance with standards, interoperability and development support.

Keywords: Web services, testing; interoperability, WS-* standards, e-Government.

1 Introduction

Similarly to numerous other countries all over the world, Hungary has its own strategy for e-government development [10]. Although Hungary has middle-ranked position in the level of e-government services [11], strategic studies and assessments showed that one of the primary deficiencies is the lack of interoperable, multi- and cross-organizational back-office functionality.

Several interoperability frameworks have been accepted by national, or union-level governments or organizations: e-Government Interoperability Framework (eGIF) in UK, [8], Federal Enterprise Architecture (FEA) in USA, [12], Standards and Architectures for e-Government Applications (SAGA) in Germany [13], European Interoperability Framework (EIF) in EU [7] etc. For interoperable cross-sector collaboration the concept of Seamless e-Government has been introduced to describe the ideal model of delivering public services [9].

A similar effort has been started in Hungary by establishing the Hungarian e-Government Framework (HeGF) [14]. The Framework proposes a SOA-based e-Government Service Bus for the implementation of the integrated back-office services. The architecture specifies three layers: process-level layer, service-level layer and message-level layer. The process-level layer orchestrates cross-organizational activities and services. The service-level layer defines interfaces, manages the basic operations, handles security, federated identity and management aspects. It is based on WS-* standards, and a wide variety of products promising conformity to them. The message layer is based on a message oriented middleware to provide reliability.

Early laboratory pilots showed the difficulties of the integration of heterogeneous components on the basis of WS-* standards. In some cases products did not follow the standards, in others poor documentation caused difficulties. Two questions arose at this point: a) are the SOA products mature enough for e-government use; b) how to select the best product at a future tender.

The rest of this paper describes a methodology developed to evaluate the interoperable behavior of SOA products and the quality of the WS-* standards implementations. Our goal was only the evaluation of the proposed methodology itself, not pre-selection of product, or making a ranking at this stage. The test cases copied from the architecture specification in the HeGF are listed in Table 1.

After presenting related work in section II, several SOA products selected for test are introduced in section III. Section IV enlists the tested WS-* standards. Section V describes the test cases. Section VI specifies the test environment. Section VII presents the test results. Section VIII concludes the paper and describes future work.

Table 1. Test cases from HeGF

Requirement		Corresponding test case
Message format	M	HTTP, SOAP 1.2
Exception handling	M	HTTP, SOAP 1.2 faults
Addressing	M	HTTP, SOAP 1.2, WS-A 1.0
Asynchronous messages	M	HTTP, SOAP 1.2, WS-A 1.0, async
Message level security	M	HTTP, SOAP 1.2, WS-A 1.0, WS-SC
Transport level security	M	HTTPS, SOAP 1.2
Binary transmission	R	HTTP, SOAP 1.2, MTOM
Reliable messaging	O	HTTP, SOAP 1.2, WS-A 1.0, WS-RM
Short-term transactions	O	HTTP, SOAP 1.2, WS-A 1.0, WS-AT
Message format	O	HTTP, SOAP 1.1
Addressing	O	HTTP, SOAP 1.1, WS-A 1.0
Addressing	O	HTTP, SOAP 1.1, WS-A 2004/08
Asynchronous messages	O	HTTP, SOAP 1.1, WS-A 1.0
Binary transmission	O	HTTP, SOAP 1.1, MTOM
Short-term transactions	O	HTTP, SOAP 1.1, WS-A 1.0, WS-AT

M=Mandatory, R=Recommended, O=Optional,
WS-A=WS-Addressing, WS-SC=WS-SecureConversation,
WS-RM=WS-ReliableMessaging, WS-AT=WS-AtomicTransaction.

2 Related Work

2.1 WS-I Basic Profile

The various WS-* standards provide too many options from which the implementers can choose what to implement. This freedom makes interoperability much harder since different vendors may choose different options to implement. Therefore the Web Services Interoperability (WS-I) Organization [1] was formed by a wide range of companies and standards development organizations to provide best practices called profiles for a selected set of standards. They also define test cases and create testing tools to verify the various implementations against these profiles. Software vendors participating in WS-I usually implement the test cases in their own products.

WS-I defines profiles for the most important WS-* standards. Basic Profile covers SOAP, WSDL, WS-Addressing and MTOM. Basic Security Profile aims WS-Security with different Security Token Profiles including SAML. Reliable Security Profile deals with WS-ReliableMessaging and WS-SecureConversation. WS-Coordination and WS-AtomicTransaction, however, are not yet included in any profiles.

The advantage of WS-I is that it covers a lot of issues regarding WS-* standards, it resolves ambiguities, it defines a lot of test cases and it also implements them. The source codes are available for public access; they can be downloaded from the WS-I web site. All the major software vendors participate in the WS-I Organization, thus the profiles defined are a results of a consensus and are expected to be supported in their products as well.

The disadvantage of WS-I is that its profiles are a result of a slow agreement process, therefore it always lags behind the newest versions of the WS-* standards. The implementations of the test cases are not up-to-date; they cannot keep up with the acquisitions in the market and the rapid evolution of the products. The test cases focus mainly on verifying the conformance to the profiles and are not derived from real customer needs.

2.2 Interop Events

While Windows Communication Foundation (WCF, codename Indigo) was being developed, Microsoft organized a series of events called Interop Plug-Fests for SOA vendors to implement a set of test cases by every participant and then execute the tests between each other. In the previous years numerous Interop Plug-Fests have been held and the web services endpoints of WCF are still available [2]. The close cooperation of Microsoft and Sun Microsystems has led to a very high level of interoperability between WCF and Metro, the web services stack of Sun.

The advantage of these Interop Plug-Fests is that there were very detailed predefined test cases for the selected WS-* standards and the executed tests resulted in immediate feedbacks to the vendors. The test specifications are still available for download. Unfortunately, most of the web pages about these Plug-Fests are no longer available, the evolution of the products is no longer followed and also the source codes cannot be downloaded.

2.3 Web Services Test Forum

Web Services Test Forum (WSTF) [3] is an open community founded by a couple of software vendors to provide test scenarios and a multi-vendor testing environment. Customers can also join the community to suggest test cases based on their needs. After accepting the test cases members of the community can implement them and provide web services endpoints to the public.

The advantage of WSTF is that it is less formal than a standards body; therefore, it is more flexible. Members of WSTF do not have to wait for the standards development organizations to complete the standards or the final version of SOA products to be released to start implementing the test cases. The source code is also accessible for the community. The current test clients provide a user interface only, no automated tests are defined. Although some test cases are already available for the various WS-* standards, not all of them are implemented yet, since the community was formed at the end of 2008. Unfortunately, Microsoft and Sun Microsystems (although acquired by the community member: Oracle) were not among the founders and Microsoft still has not joined the community yet.

Another similar initiative to WSTF is the Apache Stonehenge project [4] formed earlier than WSTF mainly by open-source vendors (Apache, WSO2, Red Hat), but Microsoft is also a participant and they also welcome other software vendors.

2.4 Other

Senthil et al. [5] examined how WS-I Basic Profile (WS-I BP) 1.0 addresses interoperability issues with the core web services standards (SOAP 1.1 and WSDL 1.1). They found that the efforts point to the right direction, however, there are some limitations, too. The main argument they brought up is that WS-I BP does not deal with such data types as float, decimal, date and time, and this can result in precision loss in interoperability scenarios.

Kuppuraju et al. [6] identified various aspects on how to test interoperability of SOA products based on a case study. They raised the issues but did not provide any solution: testing tools and test report generation are mentioned only as a future work. The main issues are compliance tests against WS-I profiles, integration tests for business processes, and performance tests. They also identified WS-* standards as a key to interoperability.

3 SOA Landscape

This section compares the set of products we selected for testing, but this set is far from complete since there are many more SOA products. The proposed test environment, however, is flexible and mature enough to extend the range of the current study to incorporate further products.

Table II. compares the selected products based on the following aspects: name, vendor name, application server name, Integrated Development Environment (IDE), web service API, web service stack implementation, supported programming languages, configuring WS-* protocols.

Other well-known SOA products subject of further investigation include FUSE from Iona based on CXF, the WSO2 SOA Platform based on Axis2, ActiveVOS from Active Endpoints, Intalio BPM from Intalio and also TIBCO Service Bus and Sonic ESB.

Table 2. Comparison of SOA products

name	vendor name	application server	IDE	WS API	WS stack	program language	configuration
WCF	Microsoft	IIS	Visual Studio	WCF	WCF	any .NET	custom XML
GlassFishESB	Sun	GlassFish	Netbeans	JAX-WS	Metro	Java	WS-Policy
RAD 7	IBM	WAS 7	RAD 7 (Eclipse based)	JAX-WS		Java	WS-Policy
WebLogic Suite	Oracle	WebLogic Server	JDeveloper	JAX-WS		Java	WS-Policy
JBoss	RedHat	JBoss AS	Eclipse	JAX-WS	Native (RedHat); Metro (Sun); CXF (Iona)	Java	custom XML; WS-Policy
Axis2	Apache	Tomcat	Eclipse	JAX-WS	Axis2 (WSO2)	Java	custom XML

Abbreviations: WCF = Windows Communication Foundation, IIS = Internet Information Services, RAD = Rational Application Developer, WAS = WebSphere Application Server, AS = Application Server.

4 WS-* Standards

This section gives a short overview of WS-* standards specified in the requirements of the Hungarian e-Government Infrastructure.

WS-Addressing (WS-A) raises addressing and routing specifications to message level thus makes them independent of the actual transport layer. The Message Transmission Optimization Mechanism (MTOM) defines how large binary data can be efficiently transmitted as part of a SOAP message. WS-ReliableMessaging (WS-RM) can minimize the impact of network communication problems. It can guarantee exactly-once message delivery and preserving the order of the messages. WS-Coordination and WS-AtomicTransaction (WS-AT) make specifying and committing transactions possible.

WS-Security is responsible for signing and encrypting parts of a SOAP message, and also for transmitting authorization tokens. WS-SecureConversation (WS-SC) is designed to support excessive encrypted message-exchange by maintaining a security context (similarly to SSL). WS-Trust defines means for issuing, renewing, exchanging and revoking security tokens by a Security Token Service (STS) (similarly to Kerberos) and makes federated authorization across security domains also possible mostly through SAML (Security Assertion Markup Language) assertions.

5 Test Aspects and Test Cases

In order to conduct testing three basic tasks were defined; each designed to be capable of assessing the existence or absence of functionalities selected for testing. For each task the functionalities checked and the relevant standards are listed. For compliance and interoperability testing both the input and the expected output parameters have been specified before actual testing was done.

5.1 Test Cases for Compliance

Calculator

The aim of this task is to test compliance with basic protocols and simple fault handling. A calculator application has to be created with the operations: addition, subtraction, multiplication and division. The tested standards are:

- SOAP 1.1 and SOAP 1.2 over HTTP
- SOAP 1.2 over HTTPS
- Fault handling with SOAP 1.2: when dividing by zero, MathFault exception is to be thrown.
- Ws-Addressing 1.0 and Ws-Addressing August 2004
- Ws-ReliableMessaging with SOAP 1.2: order of messages preserved; session properly closed.
- Ws-SecureConversation with SOAP 1.2: message level encryption and digital signature is to be applied, based on Basic256 (AES-256) algorithm. Authenticate both sides with X.509 certificates.
- WS-Trust, SAML: the different operations require different access rights provided by SAML tokens issued by a STS. (test case not yet implemented)

Asynchronous calculator

The aim of this task is the asynchronous version of the Calculator. The tested standards are:

- *WS-Addressing 1.0 with SOAP 1.1 and SOAP 1.2:* the server has to retrieve the addressing headers and use dynamic addressing when calling back the client.

Upload

This test is to check MTOM encoding compliance, by sending a 1MB file to the server. The tested standards are:

- MTOM with SOAP 1.1 and SOAP 1.2

Bank

The aim of the test is to check compliance with transaction standards. The task is to access a database through a web service. The server is a bank which provides services for modifying the balance of an account and getting the account's status. If the account number is non-existent, or during withdrawal the amount is greater than the balance, a specific BankFault exception is to be thrown. For repeatability automated SQL scripts have to be created for setting up the database. The tested standards are:

- *WS-AtomicTransaction and WS-Coordination over SOAP 1.1 and SOAP 1.2:* checking commit, rollback and exceptions. At the end of each transaction the correct amounts have to be found in the database.

5.2 Interoperability

To each service endpoint a corresponding client has to be created that tests this specific service. Clients are also web services and all have the same interface containing a single tester operation accepting the URL of the service to be called. This tester operation executes a functional test on the service observing the correct behavior, handling the expected faults and checking for unexpected exceptions resulting from protocol implementation mismatches. The return value of the operation indicates the success of the test. This method makes it possible to pair each client and each service from all the products corresponding to a given test case, and thus automatic tests can be run to check interoperability.

5.3 Development Support

This aspect refers to how products support development of web services. Different products provide different ways of WS-* protocol configuration. The task was to summarize and evaluate these possibilities.

6 Testing Environment

The testing environment was predefined and every product had to be installed and tested accordingly. This section summarizes the environment and the main problems which had to be solved.

The testing environment was built on five high-performance computers each of them capable of hosting multiple virtual machines. Each SOA product had to be installed on a separate virtual machine to avoid collisions with the others. The primary cause of collisions is that the different application servers use the same HTTP port, although in most cases these ports are reconfigurable.

For security tests X.509 certificates had to be issued for the services, clients and STSs. The certificates were generated as self-signing certificates using OpenSSL. Then they were installed in Windows with special access rights for IIS to access the private keys. The JDK had to be upgraded with the Unlimited Strength Jurisdiction Policy Files to be able to use longer keys for security. The public certificates were imported into the trust-stores of the Java products using keytool from the JDK. To import private certificates into key-stores a separate tool named pkcs12import had to be downloaded. To configure a transaction coordinator for WCF some special packages had to be installed in Windows. Also the WS-AT coordinators required the public certificates of the coordinators to be installed into the other products' trust-stores.

Predefined forms were specified for each task and each test. These forms had to be filled for every implementation. Additional forms were supplied for installation instructions and development problems.

In order to automate tests the clients also had to be created as web services, all of them providing the same interface having a single operation accepting the URL of the service to call. A simple testing tool has been created to pair each client with each service for a given test-case, and the results have been summarized in a table for each test-case.

7 Results and Evaluation

In order to validate the testing environment, including product-dependent components, forms, the automated testing program and testing methodology a series of tests have been performed. The test-cases mentioned in section V were implemented in the selected products. Both the client and service of each test case were realized as web services. The results of the tests based on the testing method described in sub-section V.B. are grouped into the following categories:

- **Passed:** the products participating in the test support the related standards and the result conforms to the expectations
- **Failed:** the products participating in the test support the related standards, but the cooperation between the parties failed for some reason: the client and the service were unable to produce the expected result
- **Not supported:** according to the documentation of the tested version of the product the given function is not supported
- **Not tested:** this feature was not supported or was undocumented in the tested version of the product, but according to the documentation of a newer version, the functionality is now supported

7.1 Compliance

In the first test session both the client and the service came from the same SOA product. This kind of configuration makes it possible to check compliance to the selected functionalities. There were 15 test cases for each of the 6 SOA products. From the 90

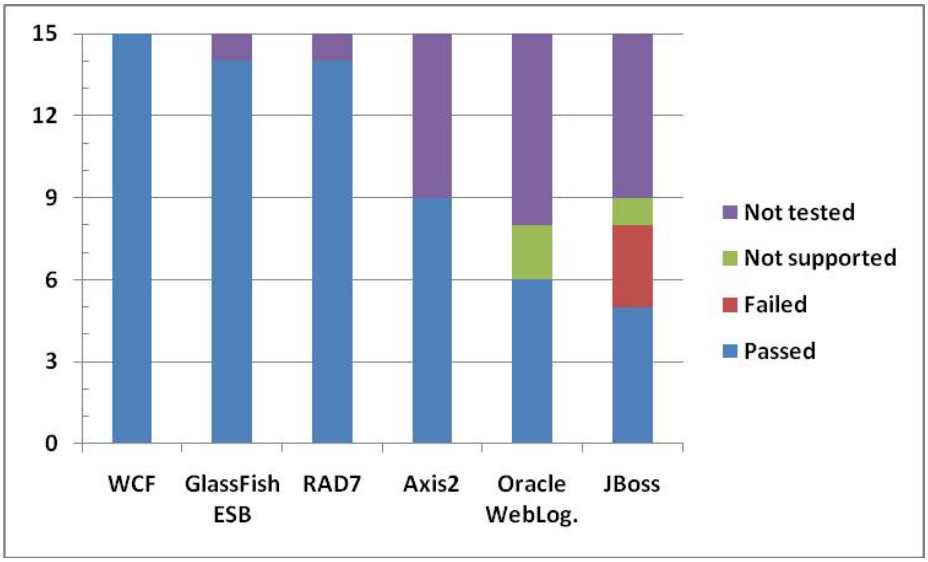

Fig. 1. Number of tests passed, failed, unsupported and untested grouped by products (products tested with themselves)

test runs 63 have passed, and only 3 have failed. The number of unsupported test cases was also 3. The relatively high number (21) of the untested results demonstrates that the SOA products are evolving rapidly.

It can be seen from Fig. 1. that WCF passed all the tests. GlassFish ESB and RAD7 also perform very well. The reason for the many untested results of the other three products is that they lacked detailed documentation at the time of the testing. Since then new versions have been released of them and also their documentations have gone through improvements, therefore, the tests have to be implemented and executed again.

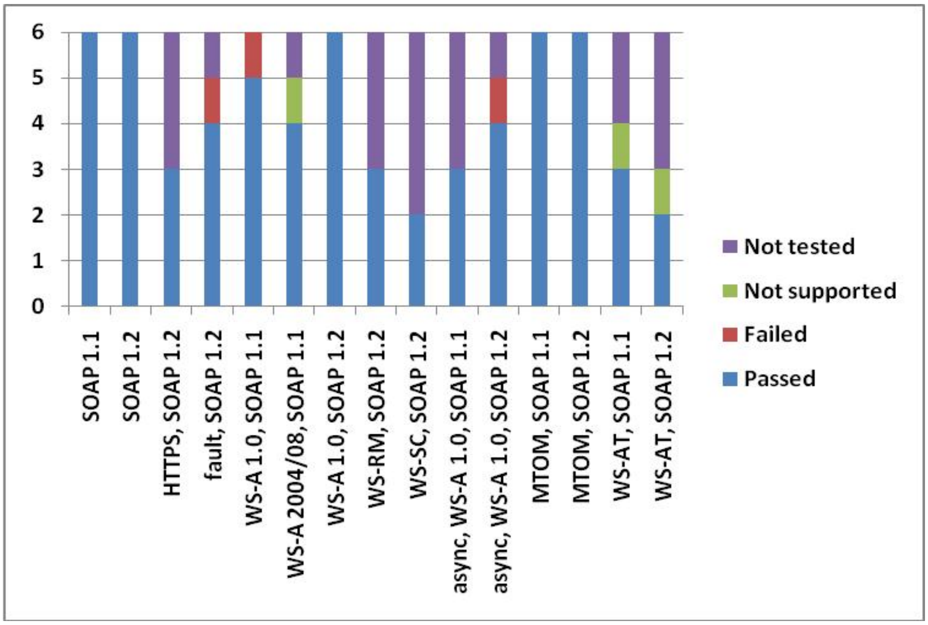

Fig. 2. Number of tests passed, failed, unsupported and untested grouped by test cases (products tested with themselves)

From Fig. 2. it can be inferred that the most supported standards are SOAP 1.1 and 1.2, WS-Addressing 1.0, and MTOM. WS-SC and WS-AT do not perform very well; they had only 2 successful runs each.

7.2 Interoperability

In the second test session the test cases were executed for each client-service pair of the SOA products (including themselves). This configuration can be used to check interoperability between different products. Having 15 test cases for 36 client-service pairs the total sum of tests is 540.

From Fig. 3. it can be seen that the results are very similar to the ones before, but more tests have failed. This means, that although the products perform well with themselves, there are still problems when communicating with the others. Another

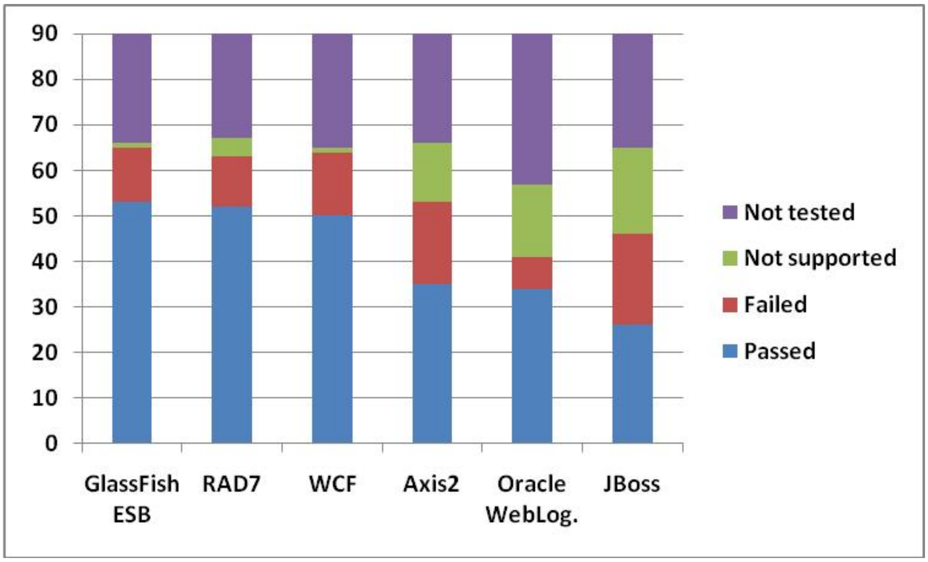

Fig. 3. Number of tests passed, failed, unsupported and untested grouped by products as services (products tested with each other)

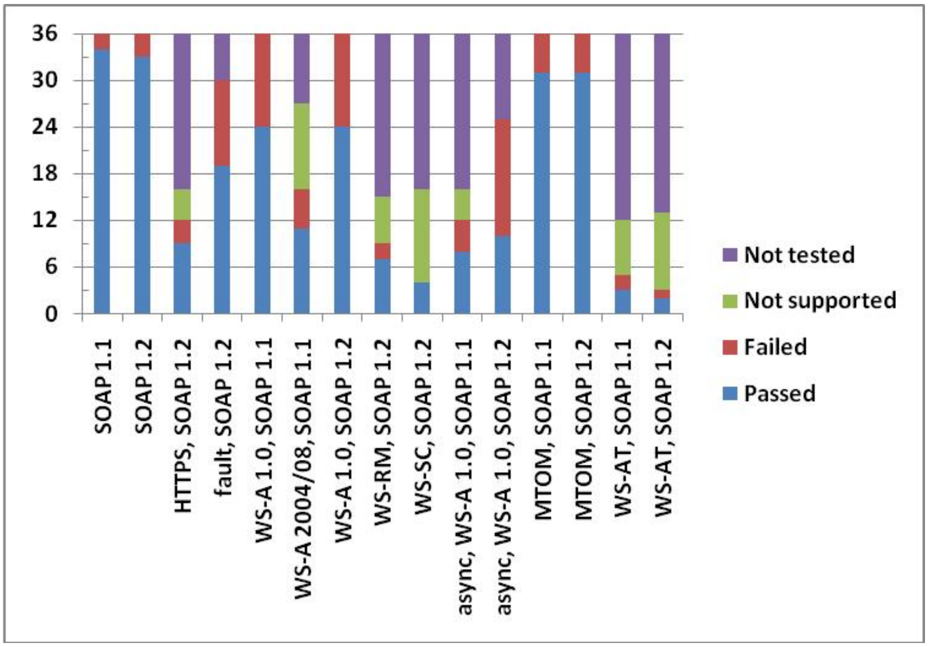

Fig. 4. Number of tests passed, failed, unsupported and untested grouped by test cases (products tested with each other)

interesting thing to note is that GlassFish ESB became the top one and WCF slid down to the third place. The reason for this is that GlassFish ESB is more permissive with the protocols, e.g. if a web service call having multiple MIME parts arrives, it will still be accepted even if MTOM is not enabled. WCF on the other hand is much stricter, and rejects every call that does not conform to the specified configuration.

Fig. 4. shows the results grouped by the test cases. It can be noted that the most and least supported standards are the same as before.

7.3 Development Support

For maintainable and interoperable development it is essential to have support for generating client proxies and service implementations from a WSDL. WCF has a tool named SvcUtil.exe, which generates service contracts as well as application configurations. JDK contains a similar tool named wsimport that does the same (less the configuration files) in the Java world. In the case of Metro the WSDL containing the bindings and policies serves directly as configuration file, too. Other JAX-WS API implementations usually rely also on wsimport, however, in most cases the configuration has to be done manually due to lack of built-in tool support.

WCF and JAX-WS implementations automatically generate WSDL-s for the deployed service endpoints. Authors have found that WS-Policy support is essential for interoperability since more complicated standards like WS-SecureConversation require many parameters, and setting them manually in a custom configuration to match the required values is very difficult and often results in unexpected errors. Some older SOA products lacked WS-Policy support, but the current versions of the examined products all perform very well regarding this aspect.

The different products provide different ways of WS-* standards configuration. These were mentioned during the introduction of the SOA products. The two main methods are either the direct usage of WS-Policy or using a custom XML configuration file. In the former case it is useful to have pre-defined policies or graphical support for policy generation. In the latter case a tool is needed to convert between the custom configuration and WS-Policies.

It is advisable to keep the program code independent of the applied protocols; therefore a separate configuration is useful. In most cases this can be achieved. Unfortunately, JAX-WS raises some protocols to programming level: the SOAP version, MTOM and WS-Addressing features are all selected by Java annotations, however, in some cases these can be overridden in configuration files.

JAX-WS provides a portable API for web services in the Java world, however, configuration of WS-* standards is out of scope resulting in vendor-dependent configuration solutions. This also makes interoperability harder as it is difficult to find the exact match for a specific configuration in another product.

7.4 Evaluation

The applied testing methodology is very similar to the one used in WSTF, but our testing environment supports automated tests, too. The test cases are not intended to formally check conformance to specific standards. The focus is mainly on achieving interoperability based on typical application patterns. From the implementations and documentations of these patterns new applications can be easily created. The test cases cover all the service level requirements of the Hungarian e-Government.

When a new version of a product was released during the testing phase, we immediately switched to that one so that we could always have the most current results. The tests ended at the end of 2008. Newer versions of the products released since then have to be retested, but it would take much less effort than the original tests. Some of the products were already mature enough in 2008 to pass most of the test cases.

Implementing the test cases helped us to learn the peculiarities of the selected products, and now we have a broader view of the different development methods. We have the virtual machines running the products, the source codes of each test case and nearly 400 pages of documentation. Based on this documentation the test environment and all the test cases can be reproduced.

8 Conclusions and Future Work

When selecting mature SOA products for e-Government application, a methodology of assessment, including test-case specifications and a flexible, automated testing environment is needed. This paper has shown a test bed suitable to assess interoperability of SOA products. The test cases are reproducible and the testing environment is flexible enough for adding a new product and having it tested with all the others. The automated tests make collecting the results easier. We also evaluated our results of tests on products of several major vendors.

The test results published in this paper only demonstrate the suitability of the testing framework for assessing interoperability based on WS-* standards. Our intention was not yet the ranking the tested SOA products, although, we have found that some SOA products are mature enough to fulfill the HeGF requirements. We would like to introduce further test aspects such as performance and stress tests.

The tested SOA products use different configuration methods. Based on a product-independent model, a code generator tool could be used to produce directly interoperable configurations. The construction of a meta-model and a tool has been started and some of its functions are already under test. This tool is also for generating common administration and management components, and also functional test cases for e-Government services. The platform-independent models of these services and the code generators producing the required components could be part of a service registry to make development easier.

As it was shown in section II, WSTF has a similar testing methodology. We have the most development experience in WCF and GlassFish ESB, which seem to be a shortage of their profile. Cooperation with them could be mutually beneficial.

References

[1] WS-I Basic Profile, `http://www.ws-i.org/` (accessed: June 11, 2009)

[2] Microsoft, Web Services Interoperability Plug-Fest, `http://www.mssoapinterop.org/ilab/` (accessed: June 11, 2009)

[3] Web Services Test Forum, `http://www.wstf.org/` (accessed: June 11, 2009)

[4] Apache, Project Stonehenge,
http://wiki.apache.org/incubator/StonehengeProposal
(last access: June 11, 2009)

[5] Senthil Kumar, K.M., Das, A.S., Padmanabhuni, S.: WS-I Basic Profile: a practitioner's view. In: Proc. IEEE International Conference on Web Services, pp. 17–24 (2004)

[6] Kuppuraju, S., Kumar, A., Kumari, G.P.: Case Study to Verify the Interoperability of a Service Oriented Architecture Stack. In: Proc. IEEE International Conference on Services Computing SCC 2007, pp. 678–679 (2007)

[7] European Interoperability Framework, http://ec.europa.eu/idabc/en/document/7728 (accessed: June 11, 2009)

[8] Saekow, A., Boonmee, C.: Towards a Practical Approach for Electronic Government Interoperability Framework (e-GIF). In: Proc. 42nd Hawaii International Conference on System Sciences HICSS 2009, pp. 1–9 (2009)

[9] Estevez, E., Janowski, T.: Government-Enterprise Ecosystem Gateway (G-EEG) for Seamless e-Government. In: Proc. 40th Annual Hawaii International Conference on System Sciences HICSS 2007, pp. 101–110 (2007)

[10] E-public administration, Strategy (2010), http://www.ekk.gov.hu/hu/ekk/strategia/egovstrategy.pdf (accessed: June 14, 2009)

[11] United Nations e-Government Survey 2008, From e-Government to Connected Governance, United Nations, New York (2008)

[12] US Government, Federal Enterprise Architecture,
http://www.whitehouse.gov/omb/e-gov/fea/ (accessed: June 14, 2009)

[13] German Government, Standards and Architectures for e-Government Applications (SAGA) 4.0 (March 2008), http://www.kbst.bund.de/saga (accessed: June 14, 2009)

[14] E_Közgazgatási Követelménytár (in Hungarian),
http://kovetelmenytar.complex.hu/ (accessed: June 14, 2009)