School of Computing Science,
University of Newcastle upon Tyne

Assessing the Risk and Value of Adopting Emerging and Unstable Web Services Specifications

Savas Parastatidis and Jim Webber

Technical Report Series

CS-TR-851

July 2004

Published by the University of Newcastle upon Tyne,
School of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK.

Assessing the Risk and Value of Adopting Emerging and Unstable Web Services Specifications

Savas Parastatidis & Jim Webber
School of Computing Science, University of Newcastle upon Tyne, UK
{Savas.Parastatidis, Jim.Webber}@newcastle.ac.uk

Abstract

With the proliferation of Web Services specifications, the question of which of the multiplicity of technologies to adopt arises. While sometimes the decision can be based purely on matching requirements to protocol capabilities, a harder consideration is whether the chosen specifications will enable wide interoperability with other systems, or lead to an architectural dead-end. This paper presents a risk assessment of several contemporary WS- protocols, enabling architects and developers to make sane choices for service interoperability.*

1. Introduction

Over the last few years, the commercial, e-Science, and Grid communities have been, and continue to be, presented with a significant number of Web Services (WS) specifications, in many cases in overlapping technology domains or in outright competition. The early hype surrounding the introduction of WS coupled with plethora of industrially-led standardisation efforts have created a feeling of fear, uncertainty and doubt (FUD) about the selection of technology for WS projects. The sheer number of specifications and the mixed signals coming from industry due to competing specifications leave application architects with an impression that there is no single clear vision for Web Services technologies.

User communities have realised the importance of Web Services for their distributed, large-scale, cross-organisation applications. However, in many cases these communities are not particularly interested in focusing on the investigation of the emerging infrastructure-specific WS technologies. The existence of stable, widely accepted, and interoperable tooling to support the underlying infrastructure is of major importance to them, not the middleware layer per se.

Web Services is a relatively young technology with few true standards in place. Seemingly every month a new specification is published, but few will actually reach stable maturity with wide industry acceptance and the benefits that this brings: interoperability, tool support, user education, and multiple vendor implementations. Given these issues, a real problem for developers is choosing which specifications to use in their implementations. If a specification is chosen too early in its lifecycle, then developers may suffer from lack of tool support and instability due to changes incurred as the specification evolves through the standardisation process. In the worst case, a specification may never be widely adopted, and so will wither and die along with any deployments that chose to support it.

Unfortunately early-adopters (e.g., Grid and e-Science communities) have found that they cannot afford to wait until the entire WS stack is finalised and production-quality tooling is made available. Often decisions have to be made to adopt emerging, but not yet stable or standardised, technologies in favour of implementing equivalent functionality from scratch[1]. There is, however, a certain level of risk associated with such a decision due to the potential interoperability, the instability of tools, and the potential for future changes to the specification.

The questions that arise are "what are the issues that middleware and service architects should consider when making the decision on adopting a Web Services specification?" and "how can the inherent risks be minimised?" This paper does not seek to address the notion of an "acceptable level of risk" since this is domain-specific. However, this paper does propose an approach to assessing the risk and value associated with the adoption of a particular Web Service specification for meeting the requirements of typical applications.

2. The Web Services stack

The Web Services stack is the conceptual architecture which middleware developers use to contextualise the

[1] While we concede that the Grid community has chosen to implement its own infrastructure in the past and continues to do so [OGSI and WSRF refs], we believe this to be a short-lived phenomenon. We hope this paper will play its part in avoiding such duplication of effort in future.

various WS protocols in relation to one-another. The canonical version of this stack is presented in Figure 1.

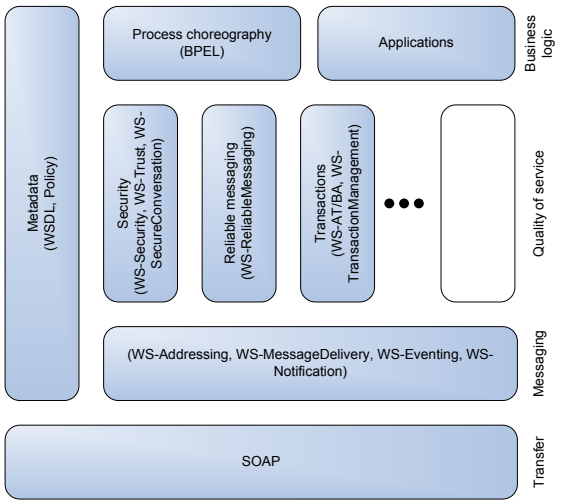

Figure 1. The Web Services stack (adapted from [9])

The transfer protocol for Web Services is SOAP [18]. Though it might be contentious in some communities, it is SOAP above all else which underlies everything else in Web Services. To reflect this, in the stack shown in Figure 1, SOAP is shown at the level of transfer protocol and assumes there will be some (arbitrary) mechanism which can be used to transport SOAP messages (e.g., HTTP, SMTP, etc.).

Layered on top of SOAP are the messaging and addressing protocols (e.g., WS-Addressing, WS-Eventing, etc.) which describe useful mechanisms and patterns for exchanging information that higher level protocols can use. The metadata and policy element of the stack governs the way in which information about the protocols is described. In particular, WSDL describes the message exchanges that a service is willing to participate in while a policy may define constrains on the content of the permissible messages (e.g. to support non-repudiation or encrypted message exchanges).

The middle layer of the stack houses the various quality-of-service protocols that underpin enterprise computing. Sometimes called the "WS-*" protocols, these specifications address important aspects of dependable systems by adding wire-level security (WS-Security), reliable messaging (either WS-ReliableMessaging or WS-Reliability), and transactions (WS-Atomic Transaction/Business Activity, WS-Transaction Management, or BTP) to the basic message-oriented architecture.

At the uppermost layer in the stack, we have the process orchestration specifications like BPEL, and applications with consume Web Services, utilising the features provided in the lower layers.

While the protocols presented here are sufficient as a basis for the majority of applications, they are in actuality a subset of the plethora of WS protocols that have been proposed [5, 7-9, 15, 16].

3. Consideration when choosing Web Services specifications

The choice of which WS specifications to adopt should be ideally be driven by the requirements of the problem domain at hand. Unfortunately given the state of flux in the Web Services standards space, often architects are forced to make educated guesses on particular protocols, or to follow the lead of one or other of the emerging camps (typically, though not exclusively, characterised as Microsoft/IBM/BEA versus Sun/Oracle). While "standards wars" still ensue, there have been notable efforts such as SOAP, WSDL, and WS-Security which have achieved broad consensus across the rival factions.

However both educated guesses and technology "brand loyalty" may be tempered by considering the stability of a given set of specifications, their market penetration, tool support, training materials, and equally importantly the collective experience that a development team has with particular technologies. In this section we outline the risks that we perceive as being the most significant to WS developers.

3.1. Stability of infrastructure

Web Services specifications like SOAP 1.1 [17] and WSDL 1.1 [19] provide the basis for the higher-level WS infrastructure and are largely stable. Industry is investing significant effort in defining interoperability profiles that govern the use of these specifications (i.e., WS-Interoperability Profile 1.0a [20]). It will take time until similar profiles are available for the rest of the WS stack in Figure 1.

The process of understanding all the issues with emerging specifications and producing a standard is time consuming and fraught with difficulty. For example, the incubation period for a W3C or an OASIS standard can be significant (of the order of years) and involve numerous parties with their own motivations, interpretations, and intents. Any development project choosing to base today's implementation on tomorrow's standards is a risky proposition which jeopardises interoperability or increases the cost of maintenance.

3.2. Interoperability and adoption

To facilitate interoperability between deployed services, it is necessary to ensure the widest adoption of a common infrastructure. For example, the WS-I Profile 1.0a and WS-Security [13] are supported by all the major software vendors in the WS area (e.g., Microsoft, IBM, Sun, Oracle, etc.) and there are open source implementations [4].

There is a risk in adopting a specification for which interoperability between tools from different vendors is not certain and/or the specification itself is not widely endorsed.

3.3. Composability

Problems may also arise from adopting specifications that do not fit well with the other protocols in the Web Services stack and the loosely-coupled framework as advocated by the service-oriented architecture style. For example, WS-RF [14] follows a resource-oriented approach to building distributed systems and, hence, it cannot be used without modifications or extensions to implementations of other specifications like BPEL [11].

3.4. Tooling

The existence of production quality tooling from multiple vendors and the open source community is of high importance to building stable middleware platforms and applications. Without such tooling the cost/benefit of developing WS-based software may be significantly reduced and the risk of vendor lock-in increases.

3.5. Experience in building WS applications

Many existing, successful commercial and scientific projects [1, 3, 10] have already built on top of WS-I Profile 1.0a. Further, in the commercial world there are examples of web services such as those from Amazon and Google [10] that demonstrate that even the simplest WS specification can be used for building production quality, business-critical services – providing access to huge amounts of dynamically updatable data with high throughput.

The dissemination of the experience gained from building such services is critical to the success of any WS-based project. The early adoption of emerging WS specifications carries the danger of not having a good understanding of all the issues surrounding the use of the technology in production deployments, and hence both increases the lead time for a project and its costs.

3.6. Documentation and Training

There is a wealth of written material on building applications from existing and stable WS technologies. Alongside this there is already a large, trained workforce. There is somewhat less material that covers advanced Web Services concepts, and fewer still are the developers who are familiar with those concepts. Thus the costs in terms of training of deploying a higher-level WS specification in a production environment are generally higher than building on lower-level specifications.

4. Production vs. experimental deployments

Of course, it is important for the community to experiment with new and emerging specifications such as WS-RF and WS-CAF [12]. A “two track” approach could be adopted where there are sufficient resources to dedicate to such an approach, using “production” and “experimental” builds of a service. In the production track only the stable or “low risk” versions of infrastructure and high-level services would be adopted. All aspects of the “production track” will have been through a rigorous standardisation and interoperability testing process with support from the wider Web Services community or a calculated risk has been taken to adopt and deploy a non-stable technology.

Investigation of new infrastructure solutions and proof-of-concept implementations could be run as part of the experimental track in projects that are able to absorb the risks associated with adopting specifications that may be volatile, are not well supported by tooling, and may not become standardised. This experimentation is vital in determining the value of a specification before it is adopted in production deployments. Experiences would be fed back into any standardisation efforts that are underway in order to steer their evolution and to the production deployments.

Obviously there is a need for a framework to be put in place for determining which specifications should be considered of low/high risk or value. Risk/value-based profiles of specifications could be created based on an analysis of the risk/value associated with the adoption of those specifications that will help middleware developers to make decisions always in combination with the particular requirements of a project/deployment.

5. Assessment methodology

Three approaches were considered for evaluating the suitability of Web Service specifications for a production system: stability/maturity spectrum, stability vs. functionality graph, and a straightforward point scoring system.

5.1. Stability/maturity spectrum

The first of the approaches uses a stability/maturity spectrum. The stability/maturity of a specification can be determined from its lifetime, incubation period in a standards organisation, its wide acceptance, interoperability testing, its inclusion in one of the WS-I profiles, and so forth. An example of a stability/maturity spectrum of some of the current WS specifications is shown in Figure 2

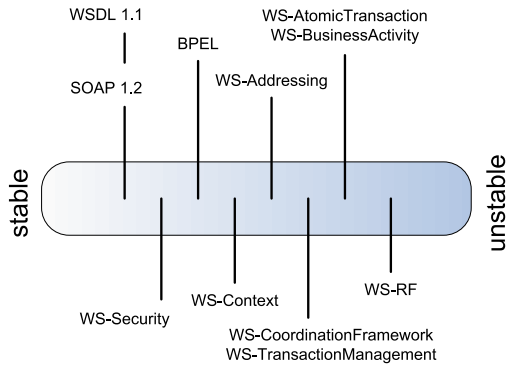

Figure 2. A spectrum for the adoption WS specifications

In the example of Figure 2, it is assumed that the stability of a specification has been determined through a combination of factors, like "number of years since it has been released and is used in experimental environments", "number of years in the standardisation process", "support by tooling", "support by software vendors/open source community", etc.

The main drawback with this approach is that it does not clearly illustrate the reasons a particular WS specification occupies a specific position on the spectrum. Its main advantage is that it provides simple means for WS application architects to determine the relative risk/stability of a particular WS specification in relation to others.

5.2. Functionality vs. stability

Service designers often need to find the right balance between adopting stable WS specifications and having to reinvent and implement functionality that is already available but in a less stable form. In such cases, it is useful to compare the stability of specifications with the value they offer to the system to be developed/deployed.

The fundamental WS specifications, SOAP and WSDL, are the building blocks on top of which other specifications are built and are both stable and provide significant functionality. If service architects require infrastructure-specific functionality in the areas of transactions, notification, workflow, etc., they need to judge which of the available alternatives specifications can be adopted or whether the functionality will be implemented in an application-specific manner, with all the dangers that both approaches entail.

A functionality vs. stability graph attempts to capture the relationship between a specification's value to the particular implementation/deployment scenario and the instability it may be introducing into the system. An example of how WS specifications can be positioned into a functionality vs. stability graph is shown in Figure 3.

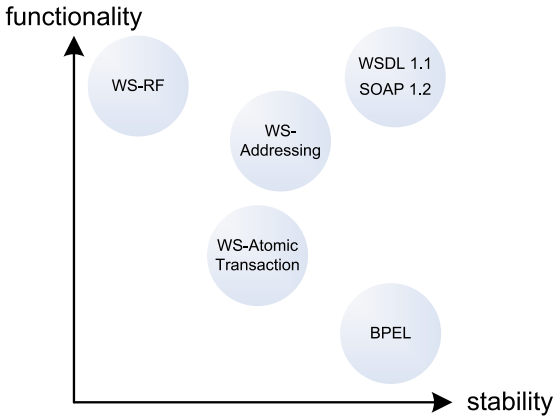

Figure 3. A functionality vs. stability graph

In the example of Figure 3, WS-Addressing is required for the message-level addressing requirements of the system, while BPEL, although more stable that WS-Addressing, does not add great value in terms of functionality to the particular project/deployment.

5.3. A point system to determine risk/value of adoption

A point-based system combines the two previous approaches, adding discrete values to enable direct comparison and allowing project-specific extensions to be introduced and used during the assessment process.

A list of properties or characteristics is created and a value between 0 and 3 is assigned to them for each of the

specifications considered for adoption. The range of values for each characteristic is the equivalent to a stability/maturity spectrum. The value is chosen according to Table 1.

Table 1. Points used when assessing risk

Value	The specification with regards to a characteristic is considered to be of
0	"no risk"
1	"low risk"
2	"medium risk"
3	"high risk"

However, merely assigning a value for the risk associated with the adoption of a specification may not reflect the requirements of a deployment. In many situations, the importance of a characteristic cannot be determined merely by its risk assessment. For example, the importance of not having to reinvent the functionality offered by a specification or the value to the application of having tool support by major commercial technology providers or the open source community has to be reflected. Furthermore, the wide adoption of a specification which has yet to be submitted to a standards body may be of greater importance in a particular scenario. Hence, it is necessary to introduce a second weighting, as shown in Table 2, for such situations.

Table 2. Points used when assessing value

Value	The specification with regards to a characteristic is considered to be of
0	"high value"
1	"medium value"
2	"low value"
3	"no value"

Service architects can choose to determine the resulting risk/value of adopting a particular specification or assess the accumulated risk/value for the all the specifications that are going to be used in the implementation of a service. In either case, the aim is to achieve as low values as possible.

Obviously, determining how many points a characteristic is to be given is somewhat subjective and certainly deployment-specific and may depend on particular deployment scenarios. The relevance of a characteristic in a given deployment scenario is itself a variable and should also be considered and perhaps, even excluded from particular assessments.

The proposed mechanism is meant to provide direction for service architects and is not meant as a definitive evaluation of existing WS specifications. Therefore, values may change between assessments and/or characteristics may be dropped/included according to their relevance.

6. Deriving characteristics

This paper proposes a list of potential characteristics that can be considered and gives values to them for existing and emerging Web Services specifications. The points represent the opinions and experiences of the authors. An explanation of the method is given as an illustration on how service architects could use the proposed approach when evaluating the risk/value of WS specification adoption.

6.1. Standards bodies

The most obvious characteristics that an architect may consider during the assessment process are related to the status of the specifications as standards or whether they have being submitted into the standardisation process. The main standards body are W3C and OASIS. The WS-I organisation, which is supported by all the major software vendors, is also of great importance.

6.2. Architecture related

Unarguably, standards are of great value to any project, especially when there are good quality tools to support them. Interoperability between deployed services is also of great value. However, service architects may have to take pragmatic decisions given the current landscape of WS specifications. They may have to choose between adopting a WS specification early vs. having to re-implement its functionality in a project-specific manner.

In some situations making a decision like that may be easier than others. For example, in the business process space the OASIS BPEL specification is an obvious candidate for a project with relevant requirements even though it has not been standardised yet. Of course, there is a risk associated with its early adoption but, at least, a project will not have to reinvent similar functionality and, as a result, spend valuable resources. Another example is WS-Addressing, which although has not been submitted to a standards body has wide support, there is only one (similar) alternative, and it is supported by industry and open source toolkits. Furthermore, there has been a long period of hardening the specification through experimentation by a closed group of few companies [2].

There are situations, though, where making such a decision is not so obvious. For example, in the Web Services transactions space, there are three competing approaches:

OASIS BTP, WS-Transactions Management, and WS-Atomic Transaction with WS-Business Activity. The BTP is a (committee-level) standard and has implementations available, but is not widely supported by the tooling from the big industrial software vendors. The second has been submitted to OASIS and it is supported by Oracle, Sun, Iona, etc. but the technical committee has not started working on it yet because of dependencies on other specifications (i.e., WS-Context and WS-Coordination Framework). Furthermore, tooling support it has not been released yet. Finally, the third has not been submitted to a standards body yet but it is supported by both Microsoft and IBM and has been demonstrated through concept applications [6].

Service architects have a difficult choice to make if their deployments require support for transactions. Determining the risk and value points is a difficult choice. However, they can minimise the impact of the differences between the available specifications by designing their system around the concepts of transactions in a service-oriented architecture. When the industry settles and agrees on a single, interoperable specification, the transition will not be as disruptive as if a project-specific solution was to be deployed.

There are situations, however, where an even harder architectural-specific decision needs to be made. For example, WS-RF follows a resource-oriented conceptual model that results in composability problems with existing Web Services specifications (e.g., BPEL). The service architect will have to devise characteristics for the assessment that will capture the architectural differences and the potential disruption to the whole system that may result from the failure of widespread adoption of WS-RF.

6.3. Vendor backing related

The support of one, a group, or all the vendors may be a factor that may influence the risk/value assessment of a specification. There may be cases where an assessment may value the support by Microsoft and IBM for a particular specification is higher than the backing of an alternative by Oracle and Sun. Of course, the converse may also stand.

Also, a particular assessment may also consider disagreement between traditional partners on a specification as a negative factor in its adoption (e.g., Microsoft and IBM having competing specifications in the eventing/notification area).

6.4. Vendor/open source community tooling related

The preference towards vendor or open source community tooling (or both) may also be reflected in the risk/value assessment of specifications. The quality of these tools (experimental, technology previews, production quality/stable) are factors that may influence the evaluation process and the ultimate decision on whether a specification should be adopted.

7. Web Services specifications risk matrix

Table 4 presents an example assessment of some of the current Web Services specifications according to the methodology described in this paper. The points assigned represent the views of the authors and are not meant to be a normative evaluation of the specifications.

For some of the characteristics in Table 4 only two of the possible values are used: 0 or 3. For example, we give a “no risk” value to all the specifications that are standardised and a “high risk” value to the rest. Some of the specifications in the latter category are already in a standardisation process, which is the reason the “specification in standardisation process” characteristic is considered. We have chosen to award those specifications that have been longer in a standards body with a lower risk factor because of the increasing stability that this often suggests. We also value the existence of industry and open source tooling and we award production quality over experimental releases.

In the project-specific section we assume an application that has security, notification, addressing, and business process/service orchestration requirements. In this particular scenario we consider the support for a specification by both Microsoft and IBM being of great value to our project since they are currently seen as leaders in the WS domain and we know that our implementation platforms are predominantly Microsoft and IBM; we award value points accordingly (0: greatest value – 4: no value).

Given the assumptions we have chosen for the hypothetical scenario, the summary of risk/value points for each specification allows us to compare their relative importance to the project. The results indicate that SOAP 1.1, WSDL 1.1, and UDDI are very safe to adopt. Also, all the security-related specifications are relative risk-free. After these specifications and given the assumptions we have made, WS-Addressing and BPEL are the next safe choices, falling in the “low” to “medium” risk/value category.

The points given to all characteristics for a particular specification are aggregated and the average is calculated. Each specification is assigned a risk/value result according to the values of Table 3

Table 3. Summary values

Average	The specification with regards to all characteristics is considered to be of
2.75-3.0	"high risk/no value"
2.25-2.75	"medium-high risk/no-low value"
1.75-2.25	"medium risk/low value"
1.25-1.75	"low-medium risk/medium-low value"
0.75-1.25	"low risk/medium value"
0.25-0.75	"no-low risk/medium-high value"
0-0.25	"no risk/high-value"

Of course, given different assumptions the results may be more favourable to different specifications.

8. Conclusions and future work

As there is no central design authority (such as the role that the OMG played for CORBA), the job of selecting appropriate WS specifications is challenging. However for services designed for immediate public consumption, we note that the "safest" approach is to adopt existing, stable Web Services specifications. Only over time as new specifications become stable, are standardised, and become widely supported by industry and the open source community, should they incorporated and used in production services.

We do not preclude situations where deployments need to take calculated risks by adopting emerging specifications that provide a particular quality of service in order to avoid having to implement the offered functionality from scratch. However, a balance between incorporated functionality and risk has to exist. Our proposal simply provides decision-support tool for helping to determine the right balance for a project.

9. References

[1] "GEODISE." http://www.geodise.org/.

[2] "Microsoft/IBM/BEA supported interoperability workshops." http://msdn.microsoft.com/webservices/community/workshops/.

[3] "myGrid." http://www.mygrid.org.uk.

[4] Apache, "Web Services Functionality Extensions (WS-FX)." http://ws.apache.org/ws-fx/.

[5] BEA, "Web Services Specifications Index." http://dev2dev.bea.com/technologies/webservices/standards.jsp.

[6] D. F. Ferguson, T. Storey, B. Lovering, and J. Shewchuk, "Secure, Reliable, Transacted Web Services: Architecture and Composition." http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwebsrv/html/wsoverview.asp, 2003.

[7] HP, "Web Services Specifications Index." http://devresource.hp.com/drc/specifications/wsrf/.

[8] IBM, "Web Services Specifications Index." http://www-106.ibm.com/developerworks/views/webservices/standards.jsp.

[9] Microsoft, "Web Services Specifications Index." http://msdn.microsoft.com/webservices/understanding/specs/.

[10] A. Nakhimovsky and T. Myers, Google, Amazon, And Beyond: Creating And Consuming Web Services: APRESS, 2003.

[11] OASIS, "OASIS Web Services Business Process Execution Language." http://www.oasis-open.org/committees/wsbpel.

[12] OASIS, "Web Services Composite Application Framework (WS-CAF)." http://www.iona.com/devcenter/standards/WS-CAF.

[13] OASIS, "Web Services Security (WS-Security)." http://www.oasis-open.org/committees/wss.

[14] OASIS, "Web Services Resource Framework (WS-RF)." http://www.oasis-open.org/committees/wsrf, 2004.

[15] ORACLE, "Web Services Specifications Index." http://otn.oracle.com/tech/webservices/htdocs/spec/.

[16] SUN, "Web Services Specifications Index." http://developers.sun.com/techtopics/webservices/standards.html.

[17] W3C, "Simple Object Access Protocol (SOAP) 1.1." http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[18] W3C, "SOAP Version 1.2 Part 1: Messaging Framework." http://www.w3.org/TR/soap12-part1.

[19] W3C, "Web Services Description Language (WSDL)." http://www.w3.org/2002/ws/desc.

[20] WS-I, "Web Services Interoperability (WS-I) Interoperability Profile 1.0a." http://www.ws-i.org.

Table 4. Risk/value assessment matrix for current Web Services specifications

General	SOAP 1.1	SOAP 1.2	WSDL 1.1	WSDL 2.0	UDDI v3.0	WS-Security	WS-SecureConversation	WS-Trust	WS-Federation	WS-ReliableMessaging (MS, IBM, BEA)	WS-ReliableMessaging (Sun and others)	WS-MetadataExchange	WS-Discovery	WS-Addressing	WS-MessageDelivery	WS-Context	WS-CoordinationFramework	WS-TransactionsManagement	WS-Coordination	WS-AtomicTransaction	WS-BusinessActivity	WS-Eventing	WS-Notification	WS-ResourceFramework	WS-DistributedManagement	Business Process Execution Language (BPEL)	WS-Choreography	Business Transaction Protocol (BTP)	WS-Policy	SAML	XACML	MOTM (check this)
Is a W3C/OASIS standard	0	0	0	3	0	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	3	0	0	3
Is in the W3C/OASIS standardisation process			0	0	0	0	3	3	3	3	1	3	3	3	2	0	1	2	3	3	3	3	0	2	0	0	0		3	0	0	0
It is part of an WS-I profile	0	2	0	3	0	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
There is one or more open source toolkits	0	2	0	3	0	0	1	3	3	3	3	3	3	0	3	3	3	3	3	3	3	3	2	2	3	2	3	3	3	1	1	
There is one or more industry toolkits available	0	2	0	3	0	0	1	2	3	3	3	3	3	0	3	3	3	3	3	3	3	3	2	2	3	1	3	0	1	2	2	
It introduces composability problems	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
Specification Assessment	**0**	**9**	**0**	**12**	**0**	**1**	**11**	**14**	**15**	**15**	**13**	**15**	**15**	**9**	**14**	**12**	**13**	**14**	**15**	**15**	**15**	**15**	**10**	**15**	**12**	**9**	**12**	**6**	**13**	**6**	**6**	**6**
Project specific																																
Value added to the project	0	3	0	2	1	0	0	3	3	2	2	1	3	0	0	0	3	3	3	3	3	0	0	3	3	0	0	3	0	0	0	0
It is supported by Microsoft AND IBM	0	0	0	0	0	0	0	0	0	0	3	0	0	0	3	3	3	3	0	0	0	0	3	3	3	0	3	3	0	0	0	0
It is supported by Microsoft	0	0	0	0	0	0	0	0	0	0	3	0	0	0	3	3	3	3	0	0	0	0	3	3	3	0	3	3	0	0	0	0
It is supported by IBM	0	0	0	0	0	0	0	0	0	0	3	0	0	0	3	3	3	3	0	0	0	3	0	0	0	0	3	3	0	0	0	0
It is supported by Sun	0	0	0	0	0	0	2	2	3	3	0	3	3	3	0	0	0	0	3	3	3	3	3	3	3	3	0	0	3	0	0	0
It is supported by Oracle	0	0	0	0	0	0	2	2	3	3	0	3	3	3	0	0	0	0	3	3	3	3	3	3	0	3	0	0	3	0	0	0
It is supported by all vendors	0	0	0	0	0	0	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	0
Project-specific Assessment	**0**	**12**	**0**	**14**	**1**	**1**	**17**	**23**	**27**	**26**	**27**	**25**	**27**	**18**	**26**	**24**	**28**	**29**	**27**	**27**	**27**	**27**	**25**	**33**	**27**	**18**	**24**	**21**	**22**	**6**	**6**	**6**
	NR/HV	LR/MV	NR/HV	LR/MV	NR/HV	NR/HV	LR-MR/LV-MV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	LR-MR/LV-MV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR/LV	MR-HR/NV-LV	MR/LV	LR-MR/LV-MV	MR/LV	LR-MR/LV-MV	LR-MR/LV-MV	NR-LR/MV-HV	NR-LR/MV-HV	NR-LR/MV-HV

HR=High Risk, MR=Medium Risk, LR=Low Risk, NR=No Risk, HV=High Value, MV=Medium Value, LV=Low Value, NV=No Value