Acquiring Evolving Technologies: Web Services Standards

Harry L. Levinson
Liam O'Brien

February 2006

Acquisition Support Program

Technical Note
CMU/SEI-2006-TN-001

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

Acknowledgements

The authors would like to thank Paulo Merson, Mary Ann Lapham, John Foreman, Ted Marz, Bud Hammons, and Michael Bandor of the Carnegie Mellon® Software Engineering Institute (SEI) for their technical reviews. Their thoughtful comments greatly improved the quality of this report. The authors wish to thank Bob Ferguson and Linda Levine for sharing their knowledge and expertise. Thanks also to Susan Kushner for her excellent editorial support.

Abstract

Software development projects rarely are started or proceed without risks involving the technologies used. Typically, many facets of a project such as system functionality and tool support depend on the availability of a specific technology. This dependency poses risks: the required technology can disappear within the project's life cycle or a promised technology may not be available when it's required.

A popular software technology today, Web services standards, is a widely supported approach to implementing a service-oriented architecture. Because Web services standards promise system interoperability and flexibility to large projects, commercial and government organizations are including it as the cornerstone of future computer-based systems. In fact, many systems currently being architected and designed assume the availability of products built upon a stable and effective set of Web services standards. This assumption presents project stakeholders with a large technology availability risk.

This technical note discusses some of the challenges of using Web services standards and presents the results generated by an assessment tool used to track the appropriateness of using this technology. The appendix includes an example built using the authors' opinions about the current level of appropriateness of using Web services standards in a typical, large software-intensive project.

1 Introduction

> "All our lauded technological progress—our very civilization—is like the axe in the hand of the pathological criminal."
>
> —Attributed to Albert Einstein

Addressing and managing evolving technology in software development is a challenge and can even seem to be an impossible job when nothing stays the same over time. In this report, the evolution of technology is viewed from two perspectives. First, software projects change over time due to modified requirements, fluctuating constraints, and altered designs due to implementation decisions. Second, technology selected for the project will change, usually for reasons beyond the control of the project. For these reasons, software architects, engineers, and project managers struggle with the need to use an evolving technology while trying to deliver a project on schedule and within budget.

An assessment tool can be used to better understand the implications of using an evolving technology within the bounds of a project that is itself likely to change. This report presents the results generated by an assessment tool the authors created for tracking certain aspects of an evolving technology, Web services standards.

1.1 Making Decisions

Each of us needs to make decisions when confronted with choices. For instance, deciding how to get from point A to point B could be daunting if one were to consider all of the available modes of transportation. Your long list of options could include the automobile, bus, airplane, train, bicycle, walking, and any combination thereof. In addition, the decision requires wrestling with conflicting factors such as how fast do I need to get to point B, how much will it cost, what is my desired level of comfort, does my choice impact the environment, are there benefits to personal health, is the mode of transportation enjoyable and convenient, just to name a few.

The decision-making process has been investigated from many different angles. This is evident in the number of textbooks that discuss decision-making. In the acquisition of software products today, tools, methods, and even regulations exist in an attempt to improve the overall quality of software-intensive systems by addressing various areas in the management of new technology. Deciding when it is beneficial to use new software technology is a common issue throughout the software development and acquisition communities. The following sections discuss why it is important to have processes and tools in place to help manage information used to make technology decisions.

1.1.1 The Challenge of Using COTS Components

The use of commercial off-the-shelf (COTS) software components is prevalent throughout software development organizations today. In theory, the reason for selecting a COTS software component is to use a proven solution, thus reducing the overall schedule and effort for a project, while improving quality. In practice, this is often a difficult goal to achieve. As discussed in this report, selecting a COTS component is only the first step in the life cycle of both the project and the technology. Many methods and approaches are available to help projects evaluate and select components that will likely integrate successfully into the desired project [SEI 05, Section "Procuring Interoperable Components"]. Many of these methods and approaches also discuss that the selection criteria for COTS components should go beyond cost considerations. For example, evaluating products based on system attributes such as performance, security, reliability and maintainability improves the chances for a successful project.

In addition to these selection issues, dealing with evolving technology presents an additional challenge:

> Building solutions based on incorporating pre-existing components is different from typical custom development in that the components are not designed to meet a project-defined specification. COTS components are built to satisfy the needs of a market segment. Therefore, an understanding of the components' functionality and how it is likely to change over time must be used to modify the requirements and end-user business processes as appropriate, and to drive the resulting architecture [Albert 02].

This quote points out one of the many challenges facing practitioners. Many approaches stress that monitoring the appropriateness of the selected COTS component throughout a product's life cycle is necessary. Thus, the need for a tool to help monitor evolving technology is evident.

1.1.2 Technology Readiness Assessments

Current Department of Defense (DoD) acquisition directives and instructions require that Technology Readiness Assessments (TRAs) be conducted several times during the life cycle of a product acquisition [DoD 03a, DoD 03b]. A TRA examines program concepts, technology requirements, and demonstrated technology capabilities in order to determine technological maturity. Maturity is described through a "recommended technology readiness level (TRL) (or some equivalent assessment) for each critical technology."

> The use of TRLs enables consistent, uniform, *[sic]* discussions of technical maturity across different types of technologies. Decision authorities will consider the recommended TRLs (or some equivalent assessment methodology, e.g., Willoughby templates) when assessing program risk. TRLs are a measure of technical maturity. They do not discuss the

probability of occurrence (i.e., the likelihood of attaining required maturity) or the impact of not achieving technology maturity [DAU 04, Section 10.5.2].

The DoD's *Technology Readiness Assessment (TRA) Deskbook* describes in detail how to identify the critical technology for a project and evaluate the TRL for that technology [DoD 05]. By design, TRLs assign a single value to make it easier to select a single technology from competing technologies by creating a single common denominator. Usually when selecting a software technology, a difficult and sometimes frustrating task is managing the various competing attributes of the whole decision. Smith discusses several "orthogonally related" attributes that should be considered when making a decision to utilize a software technology [Smith 04]. These consist of the following four attributes:

1. Requirements: How well the functional and non-functional requirements can be allocated to a solution
2. Environmental Fidelity: How closely the selected technology has been operated in the solution's environment
3. Technology Criticality: How dependent the solution is on the selected technology
4. Product Aging: The lifespan of the technology related to the lifespan of the solution and also the maturity of the technology in the marketplace

This report discusses how using a subset of these attributes helps facilitate the decision-making process.

1.2 The Challenge of Assessing Evolving Technology

These examples of software reuse and TRA processes show how important it is to gather information about a technology and then reason and even experiment to determine its appropriateness for use. In addition, these processes require that information be gathered several times during the life cycle of a product to reevaluate the technology's appropriateness. Even for complex technology, understanding the functional features is fairly straightforward. However, to make effective choices, decision makers usually need a way to make the unique characteristics of the technology more understandable. Using a tool to summarize and track these unique characteristics is one way to make this information more understandable and usable when assessing new technologies. These tools are usually built using text documents, spreadsheets, or databases to make the information available and understandable to the decision-makers.

In Section 2, we will explore some of the decisions that need to be made in large software projects using Web services standards. Section 3 describes the assessment tool used to generate the results presented in the appendix of this report. This tool was designed to track the appropriateness of Web services standards in the areas of requirements and maturity for use in large software systems. The results contained in the appendix are intended to be a starting point for project managers and software architects to help them make difficult

project-level architectural design decisions early in a project. Note, however, that they reflect a snapshot of an evolving technology as of November 2005. In an attempt to satisfy stakeholders' changing needs and expectations, assessment tools should be modified and updated frequently to meet the evolving needs of the project and the current state of the technology.

2 The Challenge of Assessing Web Services Standards

To assess the appropriateness of a technology for use within a project requires an understanding of the project's goals and how the selected technology will evolve. This section provides some insights into the challenge of assessing technology in general and Web services standards in particular. In order to better reason about the appropriateness of using Web services standards on a large project and to better relate the methods presented in this technical note to a real-world situation, we first introduce a notional project. We then look at *quality attributes*, which is one of the many software architectural concepts critical for creating successful products. Last, we discuss how Web services standards are created and evolve.

2.1 Language Translation Services Project

The notional project, Language Translation Services (LTS), is a commercial software system envisioned to provide thousands of services worldwide, with thousands of users who have different levels of system needs. Users of this system want to translate one or more words between languages. Each service in the system is designed to accept from 1 to 1000 words in one language and to return a message that contains words translated into another language. To encourage worldwide development and use, each service is limited to a single originating language and a single target language. The data communication network is sufficient to enable the required communications, but because of the distance messages travel and high network traffic, response time can be slow. Because of the need to interoperate with other systems and to encourage software reuse, the stakeholders have decided to use Web services standards as a key design principle.

For example, the following scenario can be used to reason about a few of the decisions that need to be made for LTS.

> The first part of a translation transaction requires a transfer of 1000 English words from an LTS application to an LTS service in less than 5 seconds with a .0001% or less likelihood of unauthorized viewing of the data within 50 years.

To help a system designer make tradeoff decisions, determining answers to the following questions from an architectural and implementation perspective represents large steps toward formulating a system design:

- How can performance between an LTS application and service across the worldwide network be predicted and monitored?

- How can the information be encrypted so that both the LTS application and service can decode it?
- What does the LTS application need to do to guarantee that the exact same information arrives at desired LTS service?
- Can an LTS service trust that the received message is actually from an authorized LTS application?

Before we can create an assessment tool, we need to better understand the quality attributes of a system such as LTS. Also, it would be useful to understand the mechanisms of Web services standards development. The following sections discuss quality attributes and Web services standards development and how they relate to the LTS example.

2.2 Quality Attributes

Software architecture is an important phase of the software development life cycle. There are many processes and technical concepts that are employed to create and document a software architecture. One architectural concept called *quality attributes* is used in this report to help with our assessment activity. In the software architecture field, quality attributes are sometimes referred to as "non-functional requirements" or the "-ilities."

For example, we can extract some quality attributes that are relevant to this system from what we know about the notional LTS project:

- Reliability: the ability to make sure the message actually gets to the correct system
- Performance: the requirement to move 1000 bytes of data in less than 5 seconds
- Security: make it highly unlikely that an unauthorized entity can gain access to the data.

Why is it important to consider a system's quality attributes? Early decisions in the architectural process have an impact on the subsequent quality attributes of the system. As pointed out in *Software Architecture in Practice,* defining quality attributes is a crucial activity:

1. Architecture is critical to the realization of many qualities of interest in a system, and these qualities should be designed in and can be evaluated at the architectural level.
2. Architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality, but this foundation will be to no avail if attention is not paid to the details [Bass 03, p. 72].

Another characteristic of quality attributes is that they normally compete within a system for dominance. Increasing the prominence of one quality attribute usually decreases the prominence of one or more other quality attributes. These tradeoffs, inherent in every design, are decisions that an architect should share with all stakeholders throughout the life cycle of the project.

Although there are many factors to a project's success, understanding the desired system quality attributes is one of the key influences. In the beginning of the software life cycle, architecture is usually considered at a high level of abstraction, but as Bass and colleagues point out, high-level decisions need to be backed up by detailed work [Bass 03]. Focusing on quality attributes helps the stakeholders become more aware of the ways in which tradeoffs affect how the overall system works.

2.3 Web Services Standards

Web services technology is being used industry-wide to implement interoperable service-oriented architectures (SOAs). This technology comprises a set of evolving standards that tries to address many of the goals and challenges of the overall SOA approach. Some organizations that want to lower the cost of development and maintenance for software systems, while at the same time becoming more flexible in terms of capabilities, consider Web services standards as a possible solution. A big reason that SOAs are storming the software solution space is their key quality attributes such as interoperability, extensibility, and modifiability [O'Brien 05].

When trying to predict the future state of Web services standards, it helps to understand the current process of defining and implementing them for use in solutions. While this process can be fragile, clumsy, and frustrating, it is the method used worldwide to develop an SOA that interoperates across multiple private and commercial implementations.

A key goal of Web services standards is to support interoperable machine-to-machine interaction over a network. This is accomplished today by using Extensible Markup Language (XML)-based messaging such as Web Services Description Language (WSDL), the Simple Object Access Protocol (SOAP), and the Universal Description, Discovery, and Integration (UDDI). These, as well as additional standards, are managed by a consortium of industry members. The process for developing standards is open and evolutionary and as a result, the creation of new standards and subsequent revisions is unpredictable in both content and timing.

Many organizations are working to establish open standards, but there are three that are key to the evolution of Web services standards. Each of these three organizations encourages individual and organizational membership and support from both the commercial and academic communities. Members meet frequently to evolve standards through defined processes for creation of drafts, public review, and approval of final standards.

One of the key organizations that develops Web standards is the World Wide Web Consortium (W3C[1]) founded by Tim Berners-Lee, the inventor of the World Wide Web. Starting with the Hypertext Transfer Protocol (HTTP) and working its way up to XML, SOAP, and other standards, this organization is made up of many committees whose goals are

[1] For more information about W3C, visit http://www.w3.org.

to create and maintain Web standards that the W3C calls "recommendations." Another group, the Organization for the Advancement of Structured Information Standards (OASIS[2]), is dedicated to creating the infrastructure and implementation of Web services standards. The other organization called the Web Services Interoperability Organization (WS-I[3]) delivers practical guidance, best practices, and resources for developing interoperable Web services solutions. All three of these organizations rely on the international software engineering community including commercial companies, universities, and individuals to commit the knowledge and finances that allow them to operate.

At the time of this writing, Web services standards have a significant number of prominent proponents including Microsoft, IBM, Oracle, and BEA, in addition to the open source community that demonstrates its support through many initiatives, such as an Apache Software Foundation Web services project called Axis.[4] In addition, many smaller companies, Sonic, Actional, and Systinet to name a few, have built their business plans by relying on Web services standards. There are hundreds of other companies large and small that create software components built on interoperable standards and recommendations. Many of these companies develop products that enable applications to be built by integrating components built on Web services standards at the application level. The goal of using Web services standards is to build a system by installing products released by different companies and to allow the individual components to work together seamlessly.

The amount of activity in the Web services standards arena and wide industry support lead one to believe that this technology will be significant to the software development industry for many years. One of the current problems is that the implementation of Web services standards is slow and, at times, marked by fits and starts, causing many adoption headaches. Understanding the capabilities of each standard and tracking their evolution is an activity that project stakeholders need to do effectively during the life cycle of a project. The next section describes a tool we created that helps organize and present information by relating the quality attributes of a system with many of the more popular Web services standards.

[2] For more information about OASIS, visit http://www.oasis-open.org.
[3] For more information about WS-I, visit http://www.ws-i.org.
[4] For more information about the Axis project, visit http://ws.apache.org/axis.

3 Assessing the Appropriateness of Web Services Standards

As discussed previously, it is important to make decisions about the appropriateness of a technology based on the quality attributes of the system. In the notional LTS project, the applications and services are based on Web services standards, thus creating a potential technology risk to the project. This risk is present due to evolution in the project's implementation and changes in Web services standards. The following sections describe the outcome of the evaluation of this risk by showing how we assessed the appropriateness of Web services standards with regard to impact and maturity of the Web services technologies in a typical application.

3.1 Assessing Appropriateness

Below are a few situations that might be relevant to a solution using Web services standards, such as the LTS project. Remember that these can occur throughout the product life cycle in different phases and at unpredictable times.

- Changing expectations overlap with changing Web services standards.
  - Example: Bandwidth increases in the underlying network lead users to expect improved performance from the system, but at the same time, standards have increased the number of bytes needed to send the same information.
- A design decision to use a specific standard affects one or more quality attributes.
  - Example: The application used a specific standard to transfer messages reliably between two points. The standard is changed to include an extra set of messages to guarantee accuracy, thus affecting overall performance.
- A specific standard changes for reasons beyond the project's scope, yet it affects system functionality.
  - Example: A compression standard was added to allow for efficient transmission over millions of miles for space exploration. This may have a positive or negative effect on projects that are deployed on earth.

In addition to assessing and tracking the appropriateness by using functional requirements or environmental constraints, evaluating each standard against a selected group of quality attributes and tracking the results will help us make appropriateness decisions throughout the LTS life cycle. For the LTS project, we assessed and tracked two dimensions of appropriateness of Web services standards: the impact they have on the system quality attributes and the maturity of the standards as related to the system quality attributes.

3.2 Selecting Relationships to Assess

The focus of the report by O'Brien and colleagues is to indicate the impact that an SOA approach has on a group of quality attributes of an application [O'Brien 05]. An application using Web services standards usually consists of a combination of individual standards, but the use of each standard has the potential to impact each quality attribute of an application or service in different ways. By understanding how each standard affects the quality attributes of the system, the architects, engineers, and project managers can make better assessments about how to use software based on the Web services standards. Another dimension of this assessment is the maturity of a technology. As discussed earlier, the process to create and evolve each Web services standard is volatile and currently many of the standards are changing.

However, over time the impact and maturity dimensions will change. This occurs because the Web services standards, the project requirements, the architecture, and the implementation evolve. As each standard evolves, changes will be made that may affect the impact that it has on each of the quality attributes. For example, a security standard that originally seemed to have no impact on system modifiability could be changed to restrict future architectural changes. Or the lack of features within a standard can make maintaining systems that rely on it more difficult.

When looking at a standard's maturity, it may seem obvious that the maturity increases as time goes on or that monitoring the maturity of the standard may seem unnecessary after it has been thought to reach a mature state. In reality, both of these assumptions are incorrect. A poorly conceived standard implemented in many products may have more and more features added to it, causing it to become unstable. Additionally, as the Web services standards improve overall, user expectations increase, thus requiring expanded support to specific standards.

3.3 Developing an Assessment Tool

The impact a Web services standard has on a quality attribute and the maturity of a standard are significant contributors to the project's risk and subsequent mitigation strategies. While there are other factors to consider such as the availability and quality of Web services, COTS products, and the training and skill level of available staff, we have selected impact and maturity relationships to track as input to help architects, engineers, and project managers make appropriateness assessments. As pointed out in this technical note, there are many reasons for the assessments to be conducted multiple times during a product's life cycle.

The proposed assessment tool is not complicated, although the number of standards and quality attributes to track is large. For each standard, 13 different quality attributes are evaluated in two different ways. First, the impact that the standard has in relation to each one of these quality attributes is rated. The second relationship is an evaluation of maturity, or

the likelihood that the standard will change in relation to the specific attribute. This determination can be made in various ways ranging from analytical to empirical.

We started to track these relationships in a spreadsheet. Making the results understandable and meaningful became difficult as the number of Web services standards increased. The spreadsheet was organized into six pages, with the standards grouped according to their main function. The spreadsheet format was effective, but it was hard to keep track of why each value was selected. We decided to expand the tool into a database containing six different tables. In this way, the information could be grouped and presented in various reports allowing the data to be visualized and analyzed differently.

Between August and November 2005, we evaluated Web services standards at a high level and entered information into the database assessment tool. The results are contained in the appendix of this report. There are several notes of caution to users of these results.

- The presented results were prepared to test the usefulness and validity of the assessment process and the tool.
- The assessment value selected for each cell was determined by our studying the associated Web services standard and making a "best guess" as to its impact and maturity.
- Additionally, the results include only our opinions as of November 2005; further analysis and validation through experimentation would be required to develop more accurate assessment.

3.4 Selecting a Rating Criteria

Since the intent of this exercise was to evaluate the tool, a simple three-level rating scale was selected. For the impact dimension the three levels are defined as follows:

Positive	The standard tends to support the quality attribute.
Minimal	The standard has little or no affect on the quality attribute.
Negative	The standard tends to degrade the quality attribute.

For example, a standard that implements security related features would be assessed as "positive" in relation to the security quality attribute.

The values of "Mature," "Adolescent," and "Immature" were selected to more closely relate to the maturity dimension. In addition, since the results were being viewed in a table, using different values allows the reader to more clearly determine which dimension an individual cell represents.

Mature	The standard is widely used and is not expected to change as related to the quality attribute.
Adolescent	The standard is in low use or may change as related to the quality attribute.
Immature	The standard is not in significant use or is likely to change as related to the quality attribute.

Keep in mind that a standard may be maturing in relation to certain quality attributes but because significant change is expected to happen it may be less mature in other quality attribute areas.

As a summary for each standard, we calculated an overall impact and maturity rating based on the results for all of the quality attributes. For each rating, we assigned a numeric value. The average of these values, which falls between -1 and 1, is shown at the bottom of each column. A negative average indicates an overall negative impact or low maturity; an average above zero indicates a positive impact or more mature overall assessment. Because this scale is very coarse and the relationships between the dimension and quality attribute are complex, this overall rating should be used only as a rough indication of overall impact or maturity.

3.5 Assessment Example

The example below displays ratings for one of the 13 quality attributes, Security, for the Web services standard, Security Assertion Markup Language (SAML), assessed in terms of impact and maturity in relation to our notional LTS project. This particular standard is maintained by an OASIS committee. Since this standard is directly related to the security quality attribute, the impact value we assigned is "Positive." The development of Version 1.0 of this standard began in 2001 and was adopted in November 2002. However, after three years of wide adoption, OASIS and others are actively working on Version 2.0 of this standard. For this reason, we assigned a maturity rating of "Adolescent."

	Impact	*Maturity*
Security	**Positive** Standardize passing of security information	**Adolescent** Ver. 1.0 is mature but ver. 2.0 released recently (2005)

For each quality attribute, we applied similar reasoning to assign one of the three ratings for the impact and maturity assessments. As shown in the appendix, after rating all of the relationships for this standard, an overall rating of 0.46 was calculated for impact and 0.00 for maturity. Since the overall impact rating is a positive number, it indicates that SAML has a positive impact to the overall capabilities of LTS. Because there was recent release of SAML, each maturity relationship was rated at "Adolescent" (sometimes for different reasons) to achieve the overall rating of 0.00. This value indicates that the LTS stakeholders

should monitor the project's security design decisions along with the new SAML changes as the new release becomes part of the LTS project.

The appendix contains example results for 38 Web services standards, assessed for impact and maturity, based on 13 quality attributes.

4 Conclusion

This technical note demonstrates one way of systematically assessing the appropriateness of using a popular but evolving technology, Web services standards. By focusing on the project's quality attributes, another dimension to technology assessments can be added to help software architects, engineers, and project managers make complex decisions. We chose the popular Web services standards technology as an example in the hope that the results of our examination will be useful to active projects.

Use this assessment tool and the associated process as a beginning and tailor it to meet the needs of applications and services that use Web services standards. The goal is to make informed decisions and track those decisions on a regular basis. Remember the 'axe' mentioned by Einstein; technology assumptions change frequently so the decisions based on these assumptions need to be reviewed regularly.

Appendix A Appropriateness Assessment Results

The information presented in this appendix was prepared by the authors in November 2005 and is presented as a baseline analysis of Web services standards. The reference project was a typical project using Web services standards such as the LTS project described in the report. The modification and expansion of the appropriateness assessment results presented in this appendix is required for effective use in your project. The assessment tool you use should be tailored to the specific needs of a project by

- selecting which quality attributes to track based on your project's requirements
- selecting which standards are tracked to meet project requirements
- tracking selected commercial Web services products to determine the appropriateness of the solution

One last caution is that this technical note does not address how you should make decisions such as gathering the information for each comparison or how to make system level decisions based on this tool. There are many ways to do this, ranging from plain old guessing, informal opinion gathering and synthesis, or a more structured approach like Wideband Delphi. The method you choose will vary, depending on your project's needs.

How to Read the Results

The results are presented alphabetically according to the standard's name. At the top of each page a line of text indicates the managing organization and the version and date of the standard's documentation that was used for the analysis. Each page contains two data columns. The first column represents the impact that the standard has relative to each individual quality attribute. A simple three-level scale was selected to indicate a positive, minimal, or negative impact in this relationship.

Positive	The standard tends to support the quality attribute.
Minimal	The standard has little or no affect on the quality attribute.
Negative	The standard tends to degrade the quality attribute.

The second column represents the maturity of the standard in relation to each quality attribute.

Mature	The standard is widely used and is not expected to change as related to the quality attribute.
Adolescent	The standard is in low use or may change as related to the quality attribute.
Immature	The standard is not in significant use or is likely to change as related to the quality attribute.

Each page in this appendix contains the assessment results for a single standard with regard to impact and maturity as they relate to each of the 13 quality attributes. Below each rating is a brief comment that indicates the reason for the rating.

To get an idea of the overall impact or maturity for each standard, a number between -1 and 1 is shown at the bottom of each column. For each individual result we assigned a numeric value of 1, 0, or -1 and then averaged these values for the whole column. For the impact column, the average is a rough indication of how the standard may negatively or positively impact the system. For the maturity column, the average is a rough indication of how mature the standard is in relation to the system's quality attributes. Remember that the results presented here were not derived from detailed analysis or an actual project's architecture.

WS Standard: **Asynchronous Service Access Protocol (ASAP)**

Organization: **OASIS, Ver: v1.0 5/05**

	Impact	*Maturity*
Adaptability	Positive More flexibility in integrating services and processes	Adolescent Although new, probably won't change much for this QA
Auditability	Negative Difficult to audit asynchronous services	Immature Anticipate change for this QA
Availability	Minimal Not key QA	Adolescent Although new, probably won't change much for this QA
Extensibility	Positive Allows for integration of processes	Adolescent Although new, probably won't change much for this QA
Interoperability	Positive Allows for better interoperability with longer running services	Immature Anticipate change for this QA
Modifiability	Minimal Not key QA	Adolescent Although new, probably won't change much for this QA
Operability and Deployability	Minimal Allows for asynchronous service to be integrated	Adolescent Although new, probably won't change much for this QA
Performance	Negative Asynchronous services can negatively affect performance	Immature Anticipate change for this QA
Reliability	Minimal Does not affect the reliability of the service	Adolescent Although new, probably won't change much for this QA
Scalability	Negative Asynchronous service is hard to predict as system grows	Immature Anticipate change for this QA
Security	Minimal Not key QA	Adolescent Although new, probably won't change much for this QA
Testability	Negative Difficulty in testing asynchronous services	Immature Anticipate change for this QA
Usability	Minimal Allows for monitors and controls that may provide better interactions with users	Immature Anticipate change for this QA

Impact Average: **-0.08** ***Maturity Average:*** **-0.46**

WS Standard: **Security Assertion Markup Language (SAML)**

Organization: **OASIS, Ver: v2.0 3/05**

	Impact	*Maturity*
Adaptability	Positive	Adolescent
	Not bound to specific transportation or communication protocols	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Auditability	Minimal	Adolescent
	Not key QA	Although not key QA, may change over time
Availability	Minimal	Adolescent
	Not key QA	Although not key QA, may change over time
Extensibility	Positive	Adolescent
	Allows for additional fields within messages	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Interoperability	Positive	Adolescent
	Standardizes passing of security information	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Modifiability	Positive	Adolescent
	Underlying system can change without need for changing security	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Operability and Deployability	Minimal	Adolescent
	Not key QA	Although not key QA, may change over time
Performance	Negative	Adolescent
	More messages and information need to be passed	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Reliability	Minimal	Adolescent
	Not key QA	Although not key QA, may change over time
Scalability	Positive	Adolescent
	Can handle increased usage	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Security	Positive	Adolescent
	Standardize passing of security information	Ver. 1.0 is mature but ver. 2.0 released recently (2005)
Testability	Minimal	Adolescent
	Not key QA	Although not key QA, may change over time
Usability	Positive	Adolescent
	Supports authentication and authorization	Ver. 1.0 is mature but ver. 2.0 released recently (2005)

Impact Average: **0.46** ***Maturity Average:*** **0.00**

WS Standard: **Service Provisioning Markup Language (SPML)**

Organization: **OASIS, Ver: v2.0cd 9/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature 2nd version of SPML just released
Auditability	Negative More items will need auditing	Immature 2nd version of SPML just released
Availability	Minimal Not key QA	Adolescent Although released recently, unlikely to change relative to this QA
Extensibility	Positive Can handle multiple types of resources	Immature 2nd version of SPML just released
Interoperability	Positive Provides a standard for handling provisioning across systems	Immature 2nd version of SPML just released
Modifiability	Minimal Not key QA	Adolescent Although released recently, unlikely to change relative to this QA
Operability and Deployability	Positive Provides standards for users and system access entitlements which can be automated	Adolescent Although released recently, unlikely to change relative to this QA
Performance	Negative More messages to interpret	Immature 2nd version of SPML just released
Reliability	Minimal Not key QA	Adolescent Although released recently, unlikely to change relative to this QA
Scalability	Positive Allows for extending the number of users or systems that need access entitlements	Immature 2nd version of SPML just released
Security	Positive Provides standards for handling user and system access entitlements	Immature 2nd version of SPML just released
Testability	Negative Difficult in testing the different resource handling scenarios	Immature 2nd version of SPML just released
Usability	Minimal Not key QA	Adolescent Although released recently, unlikely to change relative to this QA

Impact Average: **0.15** ***Maturity Average:*** **-0.62**

WS Standard: **Simple Object Access Protocol (SOAP)**

Organization: **W3C, Ver: v1.2d 6/03**

	Impact	*Maturity*
Adaptability	Positive Fields can be changed. Passes through firewalls	Adolescent Anticipate growth related to this QA
Auditability	Minimal Not key QA	Mature Many products designed using SOAP
Availability	Minimal Not key QA	Mature Many products designed using SOAP
Extensibility	Positive Easily add fields and formatting	Adolescent Anticipate growth related to this QA
Interoperability	Positive Designed for Interoperability	Mature Many products designed using SOAP
Modifiability	Minimal Not key QA	Mature Many products designed using SOAP
Operability and Deployability	Minimal Not key QA	Mature Many products designed using SOAP
Performance	Negative Size of message	Mature Many products designed using SOAP
Reliability	Minimal Not key QA	Mature Many products designed using SOAP
Scalability	Positive Messages can grow as big as needed	Adolescent Anticipate growth related to this QA
Security	Minimal Not key QA	Mature Many products designed using SOAP
Testability	Minimal Not key QA	Mature Many products designed using SOAP
Usability	Negative Size of message and need for tools	Mature Many products designed using SOAP

Impact Average: **0.15** ***Maturity Average:*** **0.77**

WS Standard: **SOAP MTOM and/or XOP and/or SWA**

Organization: **W3C, Ver: v0.0r 1/05**

	Impact	*Maturity*
Adaptability	Positive Fields can be changed in the message	Immature SWA dying, waiting for MTOM/XOP
Auditability	Negative May be difficult to audit optimized messages	Immature SWA dying, waiting for MTOM/XOP
Availability	Minimal Not key QA	Adolescent Either method won't be affected much
Extensibility	Positive Easily add fields and formatting to messages	Immature SWA dying, waiting for MTOM/XOP
Interoperability	Positive Defines rules that must be followed	Immature SWA dying, waiting for MTOM/XOP
Modifiability	Positive Underlying applications can change	Adolescent Either method won't be affected much
Operability and Deployability	Negative Not all actors in an SOA may be using MTOM	Immature SWA dying, waiting for MTOM/XOP
Performance	Positive Designed to optimize transmission of messages	Immature SWA dying, waiting for MTOM/XOP
Reliability	Minimal Not key QA	Adolescent Either method won't be affected much
Scalability	Positive Messages can grow but reduces size of messages	Immature SWA dying, waiting for MTOM/XOP
Security	Negative Optimizations can be changed by intermediaries	Immature SWA dying, waiting for MTOM/XOP
Testability	Negative Difficulty in testing optimizations	Immature SWA dying, waiting for MTOM/XOP
Usability	Minimal Not key QA	Adolescent Either method won't be affected much

Impact Average: **0.15** ***Maturity Average:*** **-0.69**

WS Standard: **Universal Description Discovery & Integration (UDDI)**

Organization: **OASIS, Ver: v3.0 3/05**

	Impact	*Maturity*
Adaptability	Positive	Mature
	Provides structures for defining multiple taxonomies	Third version, should be stable for this QA
Auditability	Minimal	Adolescent
	Not key QA	Anticipate improvements for this QA.
Availability	Minimal	Mature
	Does not guarantee the services will be available - just lists who is providing them	Third version, should be stable for this QA
Extensibility	Positive	Mature
	UDDI registries can be extended	Third version, should be stable for this QA
Interoperability	Positive	Mature
	Part of the foundational infrastructure for interoperable services	Third version, should be stable for this QA
Modifiability	Minimal	Mature
	Not key QA	Third version, should be stable for this QA
Operability and Deployability	Positive	Mature
	Allows various mechanisms for the publishers to add entries and users to access them	Third version, should be stable for this QA
Performance	Negative	Adolescent
	Not clear what the performance of the UDDI registry is	Anticipate improvements for this QA.
Reliability	Minimal	Mature
	Does not guarantee reliability of the underlying services	Third version, should be stable for this QA
Scalability	Positive	Adolescent
	Can handle increasing numbers of services	Anticipate improvements for this QA.
Security	Minimal	Adolescent
	Needs additional security mechanisms to be in place	Anticipate improvements for this QA.
Testability	Minimal	Adolescent
	Not key QA	Anticipate improvements for this QA.
Usability	Positive	Mature
	Allows searching for a particular service	Third version, should be stable for this QA

Impact Average: **0.38** ***Maturity Average:*** **0.62**

WS Standard: **Web Service Transfer (WS-Transfer)**

Organization: **Other, Ver: v0.0 9/04**

	Impact	*Maturity*
Adaptability	Positive Allows for change in a resource's representation	Adolescent Although not widely implemented, standard is simple
Auditability	Negative May be difficult to track use of resources for audit purposes	Immature Important QA so it might change
Availability	Minimal Positively or negatively affect the resources available to a service	Adolescent Although not widely implemented, standard is simple
Extensibility	Positive Allows for change in a resource's representation	Adolescent Although not widely implemented, standard is simple
Interoperability	Minimal Not key QA	Adolescent Although not widely implemented, standard is simple
Modifiability	Positive Allows for dynamic change of resource specifications	Adolescent Although not widely implemented, standard is simple
Operability and Deployability	Minimal Allows for deletion and reestablishment of resources	Adolescent Although not widely implemented, standard is simple
Performance	Negative Removal of resources can impact performance	Immature Performance is important so standard might change
Reliability	Minimal Not key QA	Adolescent Although not widely implemented, standard is simple
Scalability	Minimal Not key QA	Adolescent Although not widely implemented, standard is simple
Security	Negative Allows for manipulation of a server's resources and change in resource specification	Immature Security may force changes relative to this QA
Testability	Negative May be difficult to test the various resource scenarios	Immature Testing is difficult across services
Usability	Positive Allows for changes in resources which can have a positive impact on user	Adolescent Although not widely implemented, protocol is simple

Impact Average: **0.00** ***Maturity Average:*** **-0.31**

WS Standard: **Web Services Atomic Transaction (WS-AtomicTransaction)**

Organization: **W3C, Ver: v1.0 8/05**

	Impact	*Maturity*
Adaptability	Positive Allows more complex transactions to be built	Adolescent Recently submitted but all the major players support this standard
Auditability	Negative Difficult to audit potential failures	Immature Key QA so anticipate changes
Availability	Minimal Not key QA	Adolescent Recently submitted but all the major players support this standard
Extensibility	Positive Allows more complex transactions to be built	Immature Key QA so anticipate changes
Interoperability	Positive Existing transaction systems can interoperate across HW and SW vendors	Immature Key QA so anticipate changes
Modifiability	Minimal Not key QA	Adolescent Recently submitted but all the major players support this standard
Operability and Deployability	Minimal Provide consistent failure and recovery semantics	Adolescent Recently submitted but all the major players support this standard
Performance	Negative Does not guarantee performance of entire transaction	Adolescent Recently submitted but all the major players support this standard
Reliability	Positive With other standards, guarantees consistent transactions	Immature Key QA so anticipate changes
Scalability	Minimal Not key QA	Adolescent Recently submitted but all the major players support this standard
Security	Minimal Not key QA	Adolescent Recently submitted but all the major players support this standard
Testability	Negative Difficulty to test various transaction failure scenarios	Immature Key QA so anticipate changes
Usability	Positive With other standards, guarantees consistent transactions	Adolescent Recently submitted but all the major players support this standard

Impact Average: **0.15** ***Maturity Average:*** **-0.38**

WS Standard: **Web Services Business Activity Framework (WS-BusinessActivity)**

Organization: **Other, Ver: v1.0 8/05**

	Impact	*Maturity*
Adaptability	Positive Can handle changing business process interoperation	Immature 3rd version in a couple of years. Not submitted yet.
Auditability	Negative More items need to be setup for auditing	Immature 3rd version in a couple of years. Not submitted yet.
Availability	Minimal Not key QA	Adolescent Although not submitted, has strong backing and this QA probably won't change
Extensibility	Positive Can handle multiple business processes	Immature 3rd version in a couple of years. Not submitted yet.
Interoperability	Positive Provides standards for business process to interoperate across different vendor implementations	Immature 3rd version in a couple of years. Not submitted yet.
Modifiability	Minimal Not key QA	Adolescent Although not submitted, has strong backing and this QA probably won't change
Operability and Deployability	Minimal Not key QA	Adolescent Although not submitted, has strong backing and this QA probably won't change
Performance	Negative More coordination of the business processes, storing of state and metadata	Immature 3rd version in a couple of years. Not submitted yet.
Reliability	Positive Defines coordination type for handling exceptions	Adolescent Although not submitted, has strong backing and this QA probably won't change
Scalability	Minimal Not key QA	Adolescent Although not submitted, has strong backing and this QA probably won't change
Security	Negative Trust boundaries have to be established	Immature 3rd version in a couple of years. Not submitted yet.
Testability	Minimal Not key QA	Adolescent Although not submitted, has strong backing and this QA probably won't change
Usability	Positive Provides mechanisms for handling exceptions in business processes	Immature 3rd version in a couple of years. Not submitted yet.

Impact Average: **0.15** ***Maturity Average:*** **-0.54**

WS Standard: **Web Services Business Process Execution Language (WSBPEL)**

Organization: **OASIS, Ver: v2.0cd 8/05**

	Impact	*Maturity*
Adaptability	Positive Describes various mechanisms for defining business processes	Adolescent Has wide support but is actively being changed
Auditability	Negative More items will need to be audited with little support provided	Immature This QA is important and needs work
Availability	Minimal Not key QA	Adolescent Has wide support but is actively being changed
Extensibility	Positive New processes can be added using the standard	Adolescent Has wide support but is actively being changed
Interoperability	Positive Allows for coordination and sharing of information between web services	Adolescent Has wide support but is actively being changed
Modifiability	Minimal Not key QA	Adolescent Has wide support but is actively being changed
Operability and Deployability	Minimal Not key QA	Adolescent Has wide support but is actively being changed
Performance	Negative More messages required to support the process	Immature This QA is important and needs work
Reliability	Minimal Does nothing to ensure the reliability of the underlying services	Adolescent Has wide support but is actively being changed
Scalability	Minimal Not key QA	Adolescent Has wide support but is actively being changed
Security	Negative Does not ensure security level of the underlying services	Immature This QA is important and needs work
Testability	Minimal Not key QA	Adolescent Has wide support but is actively being changed
Usability	Positive The level of automation of business processes can be increased by development of tools	Immature This QA is important and needs work

Impact Average: **0.08** ***Maturity Average:*** **-0.31**

WS Standard: **Web Services Choreography Description Language (WS-CDL)**

Organization: **W3C, Ver: v0.0wd 9/05**

	Impact	*Maturity*
Adaptability	Positive An organization can change underlying implementation provided it does not change the Choreography	Immature Still in draft, key QA so anticipate change
Auditability	Negative More items need to be audited	Immature Still in draft, key QA so anticipate change
Availability	Minimal Not key QA	Immature Still in draft but still anticipate change
Extensibility	Positive An organization can change underlying implementation of its part of the Choreography	Immature Still in draft, key QA so anticipate change
Interoperability	Positive Provides for interoperability between organizations through standards	Immature Still in draft, key QA so anticipate change
Modifiability	Minimal Not key QA	Immature Still in draft but still anticipate change
Operability and Deployability	Minimal Not key QA	Immature Still in draft but still anticipate change
Performance	Negative More message traffic	Immature Still in draft, key QA so anticipate change
Reliability	Minimal Does not guarantee reliability of underlying services	Immature Still in draft, key QA so anticipate change
Scalability	Minimal Not key QA	Immature Still in draft, key QA so anticipate change
Security	Negative More places where security can be affected	Immature Still in draft, key QA so anticipate change
Testability	Minimal Not key QA	Immature Still in draft but still anticipate change
Usability	Minimal Not key QA	Immature Still in draft but still anticipate change

Impact Average: **0.00** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Context (WS-Context)**

Organization: **Other, Ver: v1.0d 10/05**

	Impact	*Maturity*
Adaptability	Positive	Immature
	Allows support for newly emerging standards such as workflow and transactions	Recent draft, key QA so anticipate change
Auditability	Negative	Immature
	Difficult in auditing which services affect a shared context	Recent draft, key QA so anticipate change
Availability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Extensibility	Positive	Immature
	Allows new services and applications to be added	Recent draft, key QA so anticipate change
Interoperability	Positive	Immature
	Allows for multiple services to share a common context	Recent draft, key QA so anticipate change
Modifiability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Operability and Deployability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Performance	Negative	Immature
	More message traffic and requires and context resource manager	Recent draft, key QA so anticipate change
Reliability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Scalability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Security	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Testability	Minimal	Immature
	Not key QA	Recent draft but still anticipate change
Usability	Positive	Immature
	Allows for sharing of a context across multiple services	Recent draft, key QA so anticipate change

Impact Average: **0.15** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Coordination (WS-Coordination)**

Organization: **Other, Ver: v1.0 8/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Although new, this QA probably won't change
Auditability	Minimal Not key QA	Adolescent Although new, this QA probably won't change
Availability	Minimal Not key QA	Adolescent Although new, this QA probably won't change
Extensibility	Positive Allows for the publication of coordination protocols and definition of extension elements	Immature Recently changed, products starting to use this standard
Interoperability	Positive Allows for specifying various coordination behaviors	Immature Recently changed, products starting to use this standard
Modifiability	Minimal Not key QA	Adolescent Although new, this QA probably won't change
Operability and Deployability	Positive Allows for control of the coordination between applications and services	Adolescent Although new, this QA probably won't change
Performance	Negative More time needed to establish and work through coordination protocols	Adolescent Although new, this QA probably won't change
Reliability	Positive Establishes a coordination protocol between	Immature Recently changed, products starting to use this standard
Scalability	Positive Allows for different coordination protocols	Immature Recently changed, products starting to use this standard
Security	Negative More areas where security can be affected and needs trusted coordinator	Immature Recently changed, products starting to use this standard
Testability	Negative More scenarios to be tested based on the choice of different coordination protocols	Immature Recently changed, products starting to use this standard
Usability	Positive Provides for different coordination protocols between applications	Immature Recently changed, products starting to use this standard

Impact Average: **0.23** ***Maturity Average:*** **-0.54**

WS Standard: **Web Services Coordination Framework (WS-CF)**

Organization: **W3C, Ver: v1.0 7/03**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature Part of WS-CAF but probably won't change in relationship to this QA
Auditability	Minimal Not key QA	Immature Part of WS-CAF but probably won't change in relationship to this QA
Availability	Minimal Not key QA	Immature Part of WS-CAF but probably won't change in relationship to this QA
Extensibility	Positive Allows for static and dynamic tailoring to fit any context	Immature Part of WS-CAF which is actively being changed
Interoperability	Positive Defines a generic coordination service that applications and services can use	Immature Part of WS-CAF which is actively being changed
Modifiability	Minimal Not key QA	Immature Part of WS-CAF but probably won't change in relationship to this QA
Operability and Deployability	Positive Help to achieve coordination between applications and services	Immature Part of WS-CAF which is actively being changed
Performance	Negative More message traffic	Immature Part of WS-CAF which is actively being changed
Reliability	Positive Once coordination is established provides more reliable communication	Immature Part of WS-CAF which is actively being changed
Scalability	Positive Allows for different coordination protocols	Immature Part of WS-CAF which is actively being changed
Security	Negative More areas where security could be affected	Immature Part of WS-CAF which is actively being changed
Testability	Minimal Not key QA	Immature Part of WS-CAF which is actively being changed
Usability	Positive Allows for better coordination between services and applications	Immature Part of WS-CAF which is actively being changed

Impact Average: **0.31** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Description Language (WSDL)**

Organization: **W3C, Ver: v2.0d 8/05**

	Impact	*Maturity*
Adaptability	Positive	Mature
	Service description in WSDL can be adapted to meet changing needs	One of the first standards, widely implemented
Auditability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Availability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Extensibility	Positive	Adolescent
	Service description in WSDL can be extended as the service interface changes	May change related to this QA
Interoperability	Positive	Adolescent
	Allows for the definition of services across multiple environments	May change related to this QA
Modifiability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Operability and Deployability	Positive	Mature
	A key piece of infrastructure for operation of services	One of the first standards, widely implemented
Performance	Negative	Adolescent
	Messages have to packed and unpacked	May change related to this QA
Reliability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Scalability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Security	Minimal	Adolescent
	Not key QA	May change related to this QA
Testability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented
Usability	Minimal	Mature
	Not key QA	One of the first standards, widely implemented

Impact Average: **0.23** ***Maturity Average:*** **0.69**

WS Standard: **Web Services Distributed Management (WSDM)**

Organization: **OASIS, Ver: v1.0 3/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Released recently and anticipate change
Auditability	Positive Limits the way that IT resources can be managed and thus the audit trail	Immature Key QA, anticipate change
Availability	Positive Provides for monitoring and enforcing a service level agreement	Immature Key QA, anticipate change
Extensibility	Minimal Not key QA	Adolescent Released recently and anticipate change
Interoperability	Positive Provides for management of IT resources using web services and use of WS standards	Immature Key QA, anticipate change
Modifiability	Minimal Not key QA	Adolescent Released recently and anticipate change
Operability and Deployability	Positive Provides for monitoring and enforcing a service level agreement	Immature Key QA, anticipate change
Performance	Minimal Not key QA	Adolescent Released recently (2005). This area could change as needed
Reliability	Positive Provides for monitoring and enforcing a service level agreement	Adolescent Released recently (2005). This area could change as needed
Scalability	Positive Can handle a number of IT resources	Immature Key QA, anticipate change
Security	Positive Limits the way that IT resources can be managed	Adolescent Released recently (2005). This area could change as needed
Testability	Minimal Not key QA	Immature Key QA, anticipate change
Usability	Minimal Not key QA	Immature Key QA, anticipate change

Impact Average: **0.54** ***Maturity Average:*** **-0.54**

WS Standard: **Web Services Dynamic Discovery (WS-Discovery)**

Organization: **Other, Ver: v0.0 4/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature Not key QA but still in draft
Auditability	Minimal Not key QA	Immature Not key QA but still in draft
Availability	Positive Dynamically locates service by type but does not provide information on the service's availability	Immature Key QA and still in draft
Extensibility	Positive Provides extensibility for more sophisticated and unanticipated scenarios	Immature Key QA and still in draft
Interoperability	Positive Allows for discovery of service with a minimum of networking support	Immature Key QA and still in draft
Modifiability	Minimal Not key QA	Immature Not key QA but still in draft
Operability and Deployability	Minimal Not key QA	Immature Not key QA but still in draft
Performance	Negative Not clear how long it takes to dynamically discover services	Immature Difficult QA and still in draft
Reliability	Minimal Not key QA	Immature Not key QA but still in draft
Scalability	Positive Allows for scaling to a large number of endpoints	Immature Key QA and still in draft
Security	Minimal Not key QA: needs other standards	Immature Not key QA but still in draft
Testability	Negative Difficult to test dynamic discovery situations	Immature Difficult QA and still in draft
Usability	Minimal Not key QA	Immature Not key QA but still in draft

Impact Average: **0.15** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Enumeration (WS-Enumeration)**

Organization: **Other, Ver: v0.0 9/04**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Not implemented widely but unlike to change relative to this QA
Auditability	Negative Difficult to audit how large data sets are handled	Immature Year old and not implemented widely
Availability	Minimal Not key QA	Adolescent Not implemented widely but unlike to change relative to this QA
Extensibility	Positive Allows for more information to be passed in a standard way	Immature Year old and not implemented widely
Interoperability	Positive Allows for better management of large shared data sets	Immature Year old and not implemented widely
Modifiability	Minimal Not key QA	Adolescent Not implemented widely but unlike to change relative to this QA
Operability and Deployability	Minimal Not key QA	Adolescent Not implemented widely but unlike to change relative to this QA
Performance	Minimal Not key QA	Immature Always looking for performance improvements
Reliability	Minimal Not key QA	Adolescent Not implemented widely but unlike to change relative to this QA
Scalability	Positive Allows for handling larger data sets	Immature Year old and not implemented widely
Security	Negative More places for security to be impacted	Immature Year old and not implemented widely
Testability	Negative Difficult to test different enumerations and to find one that works well	Immature Year old and not implemented widely
Usability	Positive Better handling of data sets	Immature Year old and not implemented widely

Impact Average: **0.08** ***Maturity Average:*** **-0.62**

WS Standard: **Web Services Eventing (WS-Eventing)**

Organization: **Other, Ver: v0.0 9/04**

	Impact	*Maturity*
Adaptability	Positive Enables change in underlying mechanisms	Immature Battling with WS-Notification and last version is 2004
Auditability	Negative More items that may need to be audited	Immature Battling with WS-Notification and last version is 2004
Availability	Negative Does nothing to guarantee underlying events	Immature Battling with WS-Notification and last version is 2004
Extensibility	Positive Allows for more sophisticated and unanticipated subscription scenarios	Immature Battling with WS-Notification and last version is 2004
Interoperability	Positive Does not rely on a particular mechanism / defines a standard for notification	Immature Battling with WS-Notification and last version is 2004
Modifiability	Minimal Not key QA	Immature Battling with WS-Notification and last version is 2004
Operability and Deployability	Positive Allows subscriber define the way messages are delivered	Immature Battling with WS-Notification and last version is 2004
Performance	Negative More message between providers and users	Immature Battling with WS-Notification and last version is 2004
Reliability	Negative Does nothing to guarantee reliability of underlying events	Immature Battling with WS-Notification and last version is 2004
Scalability	Positive Standard way to specify subscription and notification	Immature Battling with WS-Notification and last version is 2004
Security	Negative Need to leverage other specifications	Immature Battling with WS-Notification and last version is 2004
Testability	Negative More specifications/scenarios/mechanisms that need to be tested	Immature Battling with WS-Notification and last version is 2004
Usability	Positive Standard way to specify subscription and notification	Immature Battling with WS-Notification and last version is 2004

Impact Average: **0.00** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Federation Language (WS-Federation)**

Organization: **Other, Ver: v1.0 7/03**

	Impact	*Maturity*
Adaptability	Positive Service Users are required to know more about what security mechanisms providers are using.	Immature Not implemented widely and it interacts with 3 other standards
Auditability	Negative More information and scenarios to audit	Immature Not implemented widely and it interacts with 3 other standards
Availability	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA
Extensibility	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA
Interoperability	Positive Allows for multiple system to interact	Immature Not implemented widely and it interacts with 3 other standards
Modifiability	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA
Operability and Deployability	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA
Performance	Negative More messages between users and providers	Immature Not implemented widely and it interacts with 3 other standards
Reliability	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA
Scalability	Positive Can handle multiple systems	Immature Not implemented widely and it interacts with 3 other standards
Security	Positive Allows for a variety of security mechanisms to be used	Immature Not implemented widely and it interacts with 3 other standards
Testability	Negative Difficult to test scenarios for how systems will be federated	Immature Not implemented widely and it interacts with 3 other standards
Usability	Minimal Not key QA	Adolescent Not implemented widely but unlikely to be modified relative to this QA

Impact Average: **0.08** ***Maturity Average:*** **-0.54**

WS Standard: **Web Services for Remote Portlets (WSRP)**

Organization: **OASIS, Ver: v2.0d 10/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Auditability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Availability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Extensibility	Positive New interfaces and portlets can be added	Immature Key QA, anticipate change
Interoperability	Positive Provides well-defined interfaces for pluggable presentation-oriented web services	Immature Key QA, anticipate change
Modifiability	Positive Built using existing standards	Adolescent Although in draft, this QA not likely to change
Operability and Deployability	Positive Allows integration of new portlets in a portal without the need for custom coding or deployment activities	Immature Key QA, anticipate change
Performance	Negative Allows end-user to interact directly with service	Immature Key QA, anticipate change
Reliability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Scalability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Security	Negative Allows more interfaces and services to be used with more areas for security to be affected	Immature Key QA, anticipate change
Testability	Minimal Not key QA	Adolescent Although in draft, this QA not likely to change
Usability	Positive Directly targeted to end-user presentation web services	Adolescent Although in draft, this QA not likely to change

Impact Average: **0.23** ***Maturity Average:*** **-0.38**

WS Standard: **Web Services Inspection Language (WS-Inspection)**

Organization: **Other, Ver: v1.0 11/01**

	Impact	*Maturity*
Adaptability	Positive Can allow the users to pick and choose which descriptions they want to use	Immature Key QA but no activity since 2001
Auditability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Availability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Extensibility	Positive Can add new repositories of descriptions as they become available	Immature Key QA but no activity since 2001
Interoperability	Positive Provides mechanisms for referencing and utilizing existing repositories of service descriptions	Immature Key QA but no activity since 2001
Modifiability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Operability and Deployability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Performance	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Reliability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Scalability	Minimal Not key QA	Immature Key QA but no activity since 2001
Security	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Testability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001
Usability	Minimal Not key QA	Adolescent Not key QA and also no activity since 2001

Impact Average: **0.23** ***Maturity Average:*** **-0.31**

WS Standard: **Web Services Metadata Exchange (WS-MetadataExchange)**

Organization: **Other, Ver: v0.0 09/04**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Unlikely to change relative to this QA but still not clearly specified
Auditability	Minimal Not key QA	Immature Not key QA, not clearly specified
Availability	Minimal Not key QA	Adolescent Unlikely to change relative to this QA but still not clearly specified
Extensibility	Positive Allows for different types of metadata about a service to be retrieved	Immature Key QA, spec not submitted yet
Interoperability	Positive Allow for exchange of metadata between services and various users	Immature Key QA, spec not submitted yet
Modifiability	Minimal Not key QA	Immature Not key QA, not clearly specified
Operability and Deployability	Minimal Not key QA	Immature Not key QA, not clearly specified
Performance	Minimal Not key QA	Immature Not key QA, not clearly specified
Reliability	Minimal Not key QA	Adolescent Unlikely to change relative to this QA but still not clearly specified
Scalability	Minimal Not key QA	Immature Not key QA, not clearly specified
Security	Minimal May have security implications if all metadata about a service can be retrieved	Immature Not key QA, not clearly specified
Testability	Minimal Not key QA	Immature Not key QA, not clearly specified
Usability	Minimal Not key QA	Adolescent Unlikely to change relative to this QA but still not clearly specified

Impact Average: **0.15** ***Maturity Average:*** **-0.69**

WS Standard: **Web Services Notification (WSN)**

Organization: **OASIS, Ver: v1.3d 7/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature Battling with WS-Eventing and last version is 2004
Auditability	Negative Another piece to audit	Immature Battling with WS-Eventing and last version is 2004
Availability	Minimal Not key QA	Immature Battling with WS-Eventing and last version is 2004
Extensibility	Minimal Not key QA	Immature Battling with WS-Eventing and last version is 2004
Interoperability	Positive Standardizes how notifications are handled	Immature Battling with WS-Eventing and last version is 2004
Modifiability	Minimal Not key QA	Immature Battling with WS-Eventing and last version is 2004
Operability and Deployability	Positive Allows for standard way for notifying interested parties on topics	Immature Battling with WS-Eventing and last version is 2004
Performance	Negative Increase in number of messages	Immature Battling with WS-Eventing and last version is 2004
Reliability	Negative Lots of actors in an SOA have to be using the standard	Immature Battling with WS-Eventing and last version is 2004
Scalability	Positive Use standards across an SOA	Immature Battling with WS-Eventing and last version is 2004
Security	Negative More places for security to be impacted	Immature Battling with WS-Eventing and last version is 2004
Testability	Negative Adds additional items that need to be tested	Immature Battling with WS-Eventing and last version is 2004
Usability	Positive Standardizes notification on topics	Immature Battling with WS-Eventing and last version is 2004

Impact Average: **-0.08** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Policy Attachment (WS-PolicyAttachment)**

Organization: **Other, Ver: v0.0 9/04**

	Impact	*Maturity*
Adaptability	Positive The attachment of policies to service can be altered	Immature Key QA but not submitted yet
Auditability	Minimal Not key QA	Immature Not key QA but likely to change to improve auditing
Availability	Minimal Not key QA	Adolescent Not key QA, probably won't change
Extensibility	Positive Allows for multiple policies to be attached to a service	Immature Key QA but not submitted yet
Interoperability	Positive Defines mechanisms for associating policies with services	Immature Key QA but not submitted yet
Modifiability	Positive The set of policies attached to a service can be changed	Immature Key QA but not submitted yet
Operability and Deployability	Minimal Not key QA	Adolescent Not key QA, probably won't change
Performance	Negative May have a performance hit if multiple policies are attached to a service and the effective policy needs to be identified	Adolescent Not key QA, probably won't change
Reliability	Minimal Not key QA	Adolescent Not key QA, probably won't change
Scalability	Minimal Not key QA	Adolescent Not key QA, probably won't change
Security	Positive Allows for a security policy to be associated with a service	Adolescent Base standard, probably won't change
Testability	Negative Difficult to test all of the policies attached to a service and how they are handled	Immature Not key QA but likely to change to improve testing
Usability	Minimal Not key QA	Adolescent Not key QA, probably won't change

Impact Average: **0.23** ***Maturity Average:*** **-0.46**

WS Standard: **Web Services Policy Framework (WS-Policy)**

Organization: **Other, Ver: v0.0 9/04**

	Impact	*Maturity*
Adaptability	Positive Policies can be adapted based on changes in the services	Immature Key QA but not submitted yet
Auditability	Minimal Not key QA	Adolescent Although not submitted, unlikely to change relative to this QA
Availability	Minimal Not key QA	Adolescent Although not submitted, unlikely to change relative to this QA
Extensibility	Positive Policies can extended when new capabilities are added	Adolescent Although not submitted yet, designed to be extensible
Interoperability	Positive Provides for a standard way of defining capabilities, requirements and characteristics of services	Immature Key QA but not submitted yet
Modifiability	Positive The underlying policies can be changed easily	Immature Key QA but not submitted yet
Operability and Deployability	Positive Allows for the description of capabilities, requirements and characteristics of services	Immature Key QA but not submitted yet
Performance	Negative Possibly more message traffic between a service provider and user	Adolescent Unlikely to change to improve performance
Reliability	Minimal Not key QA	Adolescent Although not submitted, unlikely to change relative to this QA
Scalability	Minimal Not key QA	Adolescent Although not submitted, unlikely to change relative to this QA
Security	Positive Can be used to define security policy and dynamically interpreted	Adolescent Base standard that seems extensible enough
Testability	Negative Testing that a service meets stated policies may be difficult	Immature Not key QA but improvement needed for testing
Usability	Minimal Not key QA	Adolescent Although not submitted, unlikely to change relative to this QA

Impact Average: **0.31** ***Maturity Average:*** **-0.38**

WS Standard: **Web Services Reliable Messaging (WS-Reliability)**

Organization: **OASIS, Ver: v1.1 11/04**

	Impact	*Maturity*
Adaptability	Positive Different network transportation technologies can be used	Immature Battling with WS-ReliableMessaging, major companies on both sides
Auditability	Minimal Not key QA	Immature Battling with WS-ReliableMessaging, major companies on both sides
Availability	Positive Overcomes network and software component failures	Immature Battling with WS-ReliableMessaging, major companies on both sides
Extensibility	Minimal Not key QA	Immature Battling with WS-ReliableMessaging, major companies on both sides
Interoperability	Minimal Not key QA	Immature Battling with WS-ReliableMessaging, major companies on both sides
Modifiability	Minimal Not key QA	Immature Battling with WS-ReliableMessaging, major companies on both sides
Operability and Deployability	Positive Overcomes problems with failures	Immature Battling with WS-ReliableMessaging, major companies on both sides
Performance	Negative Increases size of messages	Immature Battling with WS-ReliableMessaging, major companies on both sides
Reliability	Positive Key QA - provides reliable messaging	Immature Battling with WS-ReliableMessaging, major companies on both sides
Scalability	Minimal Not key QA	Immature Battling with WS-ReliableMessaging, major companies on both sides
Security	Minimal Acknowledgement of message reaching destination	Immature Battling with WS-ReliableMessaging, major companies on both sides
Testability	Negative Difficulties in testing failure scenarios	Immature Battling with WS-ReliableMessaging, major companies on both sides
Usability	Positive Overcomes problems with failures	Immature Battling with WS-ReliableMessaging, major companies on both sides

Impact Average: **0.23** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Reliable Messaging Protocol (WS-ReliableMessaging)**

Organization: **OASIS, Ver: v1.0 2/05**

	Impact	*Maturity*
Adaptability	Positive Different network transport technologies can be used	Immature Battling with WS-Reliability, major companies on both sides
Auditability	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Availability	Positive Overcomes problems with failures	Immature Battling with WS-Reliability, major companies on both sides
Extensibility	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Interoperability	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Modifiability	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Operability and Deployability	Positive Overcomes problems with failures	Immature Battling with WS-Reliability, major companies on both sides
Performance	Negative Increases size of messages	Immature Battling with WS-Reliability, major companies on both sides
Reliability	Positive Overcomes failures in networks and software components	Immature Battling with WS-Reliability, major companies on both sides
Scalability	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Security	Minimal Not key QA	Immature Battling with WS-Reliability, major companies on both sides
Testability	Negative Difficulties in testing failure scenarios	Immature Battling with WS-Reliability, major companies on both sides
Usability	Positive Overcomes problems with failures	Immature Battling with WS-Reliability, major companies on both sides

Impact Average: **0.23** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Resource (WS-Resource)**

Organization: **OASIS, Ver: v1.2d 10/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature Although not key QA, standard is in an active working group
Auditability	Minimal Not key QA	Immature Although not key QA, standard is in an active working group
Availability	Minimal Does not guarantee availability of resource	Adolescent Not key QA and is not likely to change
Extensibility	Positive Extensions can be made to the existing resource handling	Immature Key QA and still in active working group
Interoperability	Positive Provides a standard mechanism for describing resources across organizations	Immature Key QA and still in active working group
Modifiability	Minimal Not key QA	Immature Although not key QA, standard is in an active working group
Operability and Deployability	Positive Allows for aggregation of resource and service information into dictionaries which can be published	Immature Key QA and still in active working group
Performance	Minimal Not key QA	Adolescent Not key QA and is not likely to change
Reliability	Minimal Not key QA	Adolescent Not key QA and is not likely to change
Scalability	Positive New resources can be added	Immature Key QA and still in active working group
Security	Minimal Not key QA	Immature Although not key QA, standard is in an active working group
Testability	Minimal Not key QA	Adolescent Not key QA and is not likely to change
Usability	Positive Provides for standardized forms of messages for interacting with a resource	Immature Key QA and still in active working group

Impact Average: **0.38** ***Maturity Average:*** **-0.69**

WS Standard: **Web Services Secure Conversation Language (WS-SecureConversation)**

Organization: **Other, Ver: v0.0 2/05**

	Impact	*Maturity*
Adaptability	Minimal	Adolescent
	Not key QA	Not submitted yet but unlikely to be modified relative to this QA
Auditability	Minimal	Immature
	Not key QA	4 years and not submitted yet
Availability	Minimal	Immature
	Not key QA	4 years and not submitted yet
Extensibility	Minimal	Adolescent
	Not key QA	Not submitted yet but unlikely to be modified relative to this QA
Interoperability	Positive	Immature
	Defines standard for handling security across systems	4 years and not submitted yet
Modifiability	Minimal	Immature
	Not key QA	4 years and not submitted yet
Operability and Deployability	Minimal	Immature
	Not key QA	4 years and not submitted yet
Performance	Minimal	Immature
	Not key QA	4 years and not submitted yet
Reliability	Minimal	Adolescent
	Not key QA	Not submitted yet but unlikely to be modified relative to this QA
Scalability	Minimal	Adolescent
	Not key QA	Not submitted yet but unlikely to be modified relative to this QA
Security	Positive	Immature
	Establishes context, sharing and session keys	4 years and not submitted yet
Testability	Negative	Immature
	More scenarios for testing	4 years and not submitted yet
Usability	Minimal	Immature
	Not key QA	4 years and not submitted yet

Impact Average: **0.08** ***Maturity Average:*** **-0.69**

WS Standard: **Web Services Security (WS-Security)**

Organization: **OASIS, Ver: 1.0 3/04**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Mature Widely implemented
Auditability	Negative More information needs to be audited	Adolescent As auditing is addressed better, changes might happen
Availability	Minimal Establish secure communication but no guarantee of service failure	Mature Widely implemented
Extensibility	Positive Security messages are extensible and additional fields can be added	Mature Widely implemented
Interoperability	Positive Allows for loose or tightly coupled systems, requires policies to be well defined	Mature Widely implemented
Modifiability	Positive Underlying service can change without change in message	Mature Widely implemented
Operability and Deployability	Minimal Not key QA	Mature Widely implemented
Performance	Negative Additional message and increased size	Adolescent Always looking for ways to improve performance
Reliability	Positive Establish secure communication	Mature Widely implemented
Scalability	Minimal Not key QA	Mature Widely implemented
Security	Positive Built for confidential message transmission	Adolescent Although widely implemented, this key QA may be affected
Testability	Negative More messages and scenarios to be tested	Adolescent As testing is addressed better, changes might happen
Usability	Minimal Not key QA	Mature Widely implemented

Impact Average: **0.15** ***Maturity Average:*** **0.69**

WS Standard: **Web Services Security Policy Language (WS-SecurityPolicy)**

Organization: **Other, Ver: v1.1 7/05**

	Impact	*Maturity*
Adaptability	Negative	Immature
	Need to rewrite engine to support additional specification mechanisms	Recently released, relies on other immature standards
Auditability	Negative	Immature
	Difficulty in auditing multiple policies and underlying security	Recently released, relies on other immature standards
Availability	Minimal	Adolescent
	Not key QA	Although recently released, unlikely to change relative to this QA
Extensibility	Positive	Immature
	Can be extended to handle additional security specifications	Recently released, relies on other immature standards
Interoperability	Positive	Immature
	Generic to a security specification and not confined to use WS-Security	Recently released, relies on other immature standards
Modifiability	Negative	Immature
	Have to be re-implemented for each security spec to verify policy	Recently released, relies on other immature standards
Operability and Deployability	Minimal	Adolescent
	Not key QA	Although recently released, unlikely to change relative to this QA
Performance	Negative	Immature
	More messages and increase in message size	Although recently released, performance improvements are unlikely
Reliability	Minimal	Adolescent
	Not key QA	Although recently released, unlikely to change relative to this QA
Scalability	Positive	Immature
	Can handle multiple specification mechanisms	Recently released, relies on other immature standards
Security	Positive	Immature
	Build specifically for managing security	Recently released, relies on other immature standards
Testability	Negative	Immature
	Difficult to test underlying security specifications and policies	Recently released, relies on other immature standards
Usability	Minimal	Adolescent
	Not key QA	Although recently released, unlikely to change relative to this QA

Impact Average: **-0.08** ***Maturity Average:*** **-0.69**

WS Standard: **Web Services Transaction Management (WS-TXM)**

Organization: **OASIS, Ver: v1.0 7/03**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Auditability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Availability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Extensibility	Positive Allows for different transaction models	Immature Although released in 2003, it has not been incorporated into products yet.
Interoperability	Positive Defines mechanisms for structuring long running transactions across applications and services	Immature Although released in 2003, it has not been incorporated into products yet.
Modifiability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Operability and Deployability	Positive Allows for long-running transactions to be handled	Immature Although released in 2003, it has not been incorporated into products yet.
Performance	Negative More messages and coordination needed	Immature Although released in 2003, it has not been incorporated into products yet.
Reliability	Positive Mechanisms for handling the reliable execution of transactions	Immature Although released in 2003, it has not been incorporated into products yet.
Scalability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Security	Negative More places where security could be impacted	Immature Although released in 2003, it has not been incorporated into products yet.
Testability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.
Usability	Minimal Not key QA	Immature Although released in 2003, it has not been incorporated into products yet.

Impact Average: **0.15** ***Maturity Average:*** **-1.00**

WS Standard: **Web Services Trust Language (WS-Trust)**

Organization: **Other, Ver: v0.0 2/05**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA
Auditability	Negative More specifications and scenarios to be audited	Immature Key security standard, recently updated (2005)
Availability	Positive Ability to establish more trustworthy services	Immature Key security standard, recently updated (2005)
Extensibility	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA
Interoperability	Positive Defines standards for handling secure communications	Immature Key security standard, recently updated (2005)
Modifiability	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA
Operability and Deployability	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA
Performance	Negative More messages may need to be transferred	Immature Performance may need to be improved
Reliability	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA
Scalability	Minimal Not key QA	Immature Scalability may need to be improved
Security	Positive Extends WS-Security for secure communication	Immature Key security standard, recently updated (2005)
Testability	Negative More specifications and scenarios to be tested	Immature Key security standard, recently updated (2005)
Usability	Minimal Not key QA	Adolescent Not key QA, unlikely to change relative to this QA

Impact Average: **0.00** ***Maturity Average:*** **-0.54**

WS Standard: **WS-Addressing or WS-MessageDelivery**

Organization: **W3C, Ver: v0.0d 8/04**

	Impact	*Maturity*
Adaptability	Positive Addressing and Message delivery options can be changed	Immature Battle between these 2 standards
Auditability	Minimal Not key QA	Adolescent Not key so neither standard will change for this QA
Availability	Positive Improves message transmission	Immature Battle between these 2 standards
Extensibility	Positive Easily to add fields and formatting to underlying SOAP message	Immature Battle between these 2 standards
Interoperability	Positive A standard way of identifying endpoints	Immature Battle between these 2 standards
Modifiability	Minimal Not key QA	Adolescent Not key so neither standard will change for this QA
Operability and Deployability	Positive Improves reliability of message transmissions	Immature Battle between these 2 standards
Performance	Negative Adds additional information in messages making them larger	Immature Battle between these 2 standards
Reliability	Positive Improves reliability of message transmission	Immature Battle between these 2 standards
Scalability	Positive Improves message transmission	Immature Battle between these 2 standards
Security	Positive Secures end-to-end endpoints in messages	Immature Battle between these 2 standards
Testability	Minimal Not key QA: but endpoint addressing improved	Immature Battle between these 2 standards
Usability	Minimal Not key QA	Adolescent Not key so neither standard will change for this QA

Impact Average: **0.54** ***Maturity Average:*** **-0.77**

WS Standard: **XML-Encryption**

Organization: **W3C, Ver: rec 3/02**

	Impact	*Maturity*
Adaptability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Auditability	Negative	Immature
	More information needs auditing but information is encrypted	May be impacted by future protocols for auditing
Availability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Extensibility	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Interoperability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Modifiability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Operability and Deployability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Performance	Negative	Immature
	Encryption and Decryption needed which requires extra time to process messages	Always looking for improvements in performance
Reliability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Scalability	Minimal	Adolescent
	Not key QA	Older standard, not supported widely in commercial products
Security	Positive	Immature
	Encryption of messages	May be impacted as new security features appear
Testability	Negative	Immature
	More scenarios to test	May be impacted as new features need to be tested
Usability	Negative	Adolescent
	Encryption may cause delays in user responses	Older standard, not supported widely in commercial products

Impact Average: **-0.23** ***Maturity Average:*** **-0.31**

WS Standard: **XML-Signature**

Organization: **W3C, Ver: rec 2/02**

	Impact	*Maturity*
Adaptability	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Auditability	Negative More information and scenarios need to be audited	Adolescent Older standard, not supported widely in commercial products
Availability	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Extensibility	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Interoperability	Positive Once keys are established XML documents can be exchanged between systems	Adolescent Older standard, not supported widely in commercial products
Modifiability	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Operability and Deployability	Minimal Not key QA: requires keys to be allocated and managed	Adolescent Older standard, not supported widely in commercial products
Performance	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Reliability	Positive Guarantee only user with key can access message content	Adolescent Older standard, not supported widely in commercial products
Scalability	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products
Security	Positive Associates a key with data passed in a message, needs additional standards	Immature May change since it is security related
Testability	Negative Difficulty testing without the keys sorted out	Adolescent Older standard, not supported widely in commercial products
Usability	Minimal Not key QA	Adolescent Older standard, not supported widely in commercial products

Impact Average: **0.08** ***Maturity Average:*** **-0.08**

References

URLs are valid as of the publication date of this document.

[Albert 02] Albert, C. & Brownsword, L. *Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview* (CMU/SEI-2002-TR-009, ADA405844). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports/02tr009.html

[Bass 03] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, Second Edition*. Boston, MA: Addison-Wesley, 2003 (ISBN 0-321-15495-9).

[DAU 04] Defense Acquisition University (DAU). *Defense Acquisition Guidebook.* http://akss.dau.mil/dag/DoD5000.asp (2004).

[DoD 03a] Department of Defense. *DoD Directive The Defense Acquisition System (DoD 5000.1)*. May 2003.

[DoD 03b] Department of Defense. *DoD Instruction Operation of the Defense Acquisition System (DoDI 5000.2)*. May 2003.

[DoD 05] Department of Defense. *Technology Readiness Assessment (TRA) Deskbook.* http://www.defenselink.mil/ddre/doc/tra_deskbook_2005.pdf (2005).

[O'Brien 05] O'Brien, L.; Bass, L; & Merson, P. *Quality Attributes and Service-Oriented Architectures* (CMU/SEI-2005-TN-014). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tn014.html

[SEI 05] Software Engineering Institute. *Guide to Interoperability: Procuring Interoperable Components*. http://www.sei.cmu.edu/isis/guide/engineering/procurement.htm (2005).

[Smith 04] Smith, J. *An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software* (CMU/SEI 2004-TR-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr013.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2006	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Acquiring Evolving Technologies: Web Services Standards	FA8721-05-C-0003

6. AUTHOR(S)

Harry L. Levinson, Liam O'Brien

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2006-TN-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Software development projects rarely are started or proceed without risks involving the technologies used. Typically, many facets of a project such as system functionality and tool support depend on the availability of a specific technology. This dependency poses risks: the required technology can disappear within the project's life cycle or a promised technology may not be available when it's required.

A popular software technology today, Web services standards, is a widely supported approach to implementing a service-oriented architecture. Because Web services standards promise system interoperability and flexibility to large projects, commercial and government organizations are including it as the cornerstone of future computer-based systems. In fact, many systems currently being architected and designed assume the availability of products built upon a stable and effective set of Web services standards. This assumption presents project stakeholders with a large technology availability risk.

This technical note discusses some of the challenges of using Web services standards and presents the results generated by an assessment tool used to track the appropriateness of using this technology. The appendix includes an example built using the authors' opinions about the current level of appropriateness of using Web services standards in a typical, large software-intensive project.

14. SUBJECT TERMS	15. NUMBER OF PAGES
acquisition, COTS, interoperability, security, software-intensive system, assessment, maturity, life cycle, reuse, service-oriented architecture, SOA, web services, standard, software development, risk	64

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL