Why Standards Are Not Enough To Guarantee End-to-End Interoperability

Grace A. Lewis, Edwin Morris, Soumya Simanta, Lutz Wrage
Software Engineering Institute, Pittsburgh, PA USA
{glewis, ejm, ssimanta, lwrage}@sei.cmu.edu

Abstract

Standards have been instrumental in achieving the significant level of systems interoperability we rely on in almost every domain. Many organizations are betting on the ability of standards to provide unprecedented end-to-end systems interoperability with partner organizations.

Our experience suggests that the expectation of what can be achieved with standards is too high. For example, many large health care providers assume that moving to a new version of a major health care standard (HL7 3.0 and the associated reference information models) will guarantee “seamless” interoperability across all health care systems inside the enterprise. However, this new standard does not take into account clinical workflows and operational contexts that differ across point-of-care systems and have a large effect on how accurately clinicians interpret data.

This paper offers a caution, not that standards are not useful but that organizations need to be aware of limitations of standards they are adopting to achieve systems interoperability. After discussing these limitations, we present some strategies to minimize their effect.

Copyright Information

This Paper is subject to the Copyright Terms contained in Section 10. Please read the Copyright Terms fully before making any use of this paper.

1. Introduction

“Over the years, our industry has tried many approaches to come to grips with the heterogeneity of software. But the solution that has proven consistently effective—and the one that yields the greatest success for developers today—is a strong commitment to interoperability. That means letting different kinds of applications and systems do what they do best, while agreeing on a common ‘contract’ for how disparate systems can communicate to exchange data with one another.”

—Bill Gates

By now, almost everyone agrees that the only realistic approach to getting diverse applications to share information involves reaching agreements on the structure and function of the information to be shared. These agreements are often reflected in standards that provide a common interface that multiple vendors and application builders support.

Standards have been instrumental in achieving a significant level of interoperability that we rely on in almost every domain. Many of us use far more of these standards on a daily basis than we are aware of. For example, the Wikipedia category "Internet Protocols" lists 171 pages for protocols used primarily on the Internet—some standardized and some not.

Currently, most standards primarily support machine-to-machine interoperability. For example, Web services standardization activity spearheaded by the World Wide Web Consortium (W3C) [http://www.w3c.org] and OASIS [http://www.oasis-open.org] has made service-oriented architecture a reality. The goal of the Web services activity is to provide standards that support interoperability such that systems can be combined in a loosely coupled way to achieve complex operations. In the last several years there has been an increased demand to support people-to-machine and people-to-people interactions using computers. For example, health care organizations expecting to leverage automated decision support systems have made considerable investments in creating a formal and standard vocabulary to enable this type of interactions.

The promise of standards is attractive. Many organizations are betting on the ability of the community to develop standards that guarantee seamless
interoperability. While standards are useful and in many ways indispensable, expectations of what can be achieved through standards are unrealistic.

Section 2 outlines different levels of interoperability that frame the discussions in the rest of the paper. The following sections are organized according to limitations or problems with standards that make it difficult to achieve the desired level of systems interoperation. Section 3 discusses what standards have achieved so far for systems interoperability. Section 4 outlines some of the reasons why standards are not enough. Section 5 presents our position on why standards may never be enough. Section 6 outlines why in some cases standards may be too much. Finally, Section 7 presents ways in which to address the limitations of standards presented in the paper.

2. At What Levels Do Systems Interoperate?

Our first task will be to distinguish different levels of interoperability. We’ll create a framework of interoperability that builds on previous work [1, 2, 3, 4, 5]. And then use this framework in analyzing current and potential standards through the rest of the paper.

Interoperability has been defined as: The ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to an agreed operational semantics [4]. Although models of interoperability exist, we provide a different model that focuses on the notion of "end-to-end" interoperability as discussed in this paper.

2.1. Level 1: Machine Level Interoperability

Each interoperating system must support mechanisms that allow exchange of data at the lowest level—the machine or hardware level. For example, a big endian hardware platform interoperating with a little endian machine requires understanding about the notation used by the other entity. Today, such agreements at the machine level are standardized and implemented in compilers, virtual machines, and operating systems that hide the issues, therefore facilitating interoperability. Another example is the popular TCP/IP protocol set that hides details of actual interactions between communicating machines from users.

2.2. Level 2: Syntactic Interoperability

Software systems implemented using various high-level programming languages on different platforms are the norm today. For these heterogeneous systems to “talk” to each other, they must understand the format and structure of the data exchanged. For example, a C++ software component running on a Unix machine must know what parameters to pass and in what order to successfully invoke a remote Java component running on a Windows machine. Another example is middleware technologies and XML-based Web services that provide syntactic interoperability allowing systems implemented in different programming languages to exchange data with each other.

2.3. Level 3: Semantic Interoperability

This level of interoperability concerns the actual meaning of the data being exchanged. Although the previous two levels allow successful sharing of data, they do not ensure that all interacting systems have the same understanding of the shared data. This is not possible unless the interacting systems agree and conform to a common semantics for all shared data. For example, it is necessary to know if a quote from an online vendor includes sales tax or not when exchanging price data. Semantic interoperability is most commonly achieved by informal agreements [11]. Domain-specific ontologies provide a promising mechanism for formalizing this understanding among interacting systems.

2.4. Level 4: Organizational Interoperability

The most complex level of interoperability recognizes that the context of the information being shared is important when business processes, workflows, and data transcend organizational boundaries. A fundamental goal of interoperating systems is not only to share and understand data but also to perform “actions” on it. “Actionable” data allows systems to make changes to data that are consistent and understood by all other interoperating systems. Additionally, these changes should not produce any undesired consequences on the other systems.

If systems are to successfully interoperate they must have consensus and reach agreements at these levels. Agreements on more levels mean higher interoperability. However, current technologies, stan-
Achieving Level 3 (Semantic Interoperability) often requires human interpretation of data, while Level 4 (Organizational Interoperability) requires human-to-human interaction to ensure that actionable information is treated in a manner that is consistent with organizational processes of each interacting entity [15]. As discussed later in the paper, achieving semantic and organization level interoperability is significantly more complex and sometimes not even possible today. In the model described above, interoperability at any level implies interoperability at all the lower levels.

3. What Have Standards Done For Us?

Standards have enabled significant progress in achieving interoperability between large numbers of interacting entities [4]. The popularity and wide reach of the Internet, based on standards such as TCP/IP, SMTP, UTF-8, XML, and HTML, demonstrates the importance of standards. It is hard to imagine an Internet where different Web browsers would be required to browse different Web sites.

Standards, when mature and widely adopted, reduce the implementation burden by insulating the developers and administrators from the complexities of the underlying protocols, thus allowing them to focus on core functionality of the product or service they are building. For example, Web services are fundamentally based on a small number of widely accepted standards created mainly by OASIS and W3C. Although these standards are still evolving, they have played a significant role in adoption of service-oriented and grid architectures. By adhering to these Web services standards, service providers as well as vendors of web service-related tools and infrastructure products can create platform-independent services and products that are interoperable at Levels 1 and 2 (at least). Also, the more standardized the products from the vendors, the more choices a consumer has when it comes to selecting among various products based on the same standard.

We take for granted standards that enable the first two levels of interoperability. Current standards and technology allow the interactions at the first two levels to be completely machine-to-machine (i.e., fully automated). For example, a computer system—regardless of the operating system it is running on—can be plugged into the Internet and can communicate with millions of other computers transparently. Having standards that facilitate interoperability at the first two levels has motivated the software research community to move toward finding solutions and standards that address the next two levels.

4. Why Current Standards Are Not Enough

Despite the achievements of current standards, they are not sufficient for achieving end-to-end systems interoperability. We focus on two specific areas: their inability to address semantic and organizational levels of interoperability as outlined in the levels of interoperability framework, as well as the need for addressing quality of service needs.

4.1. Semantic and Organizational Level Interoperability

Current standards fail to fully address semantic and organizational level interoperability, and it is not certain that standards can ever address these levels completely. At these levels, there are basically two problems in reaching agreement on standards:

1. A technical problem involving the development of a language sufficiently rich to capture and model a particular domain
2. A human problem involving agreement on what is contained within the domain, and why it is important

In terms of semantic interoperability, the first problem is usually addressed through the creation of domain-specific ontologies as one way to standardize “meaning” and “understanding” of shared data. There has been significant progress in efforts to define languages that describe domain-specific ontologies in a machine-processable form, such as OWL-S and WSMO, and languages such as BPEL and BPML, which can be used to model business processes [12, 13].

However, the creation of an ontology, even within a single domain is not an easy task. Using the medical domain as an example, it would require the standardization of terminology and procedures across the entire medical domain. This not only requires considerable effort for defining, socializing and evolving the ontology. Standard ontologies often require tailoring because of differences between organizations in how they may require specific features. This tailoring can defeat the whole purpose of standardization because often the tailoring is organization-specific, thus making the ontology non-standard.
Where we have not made significant progress is in addressing the second problem of reaching human agreement [10]. For the most part, the community is unaware of this issue because attention has been focused on the lower levels of interoperability (e.g., machine and syntactic). Moreover, we have only recently developed the computer and networking capabilities that have created the impetus for organizations to develop interoperable business processes. Practitioners (primarily researchers) are only now trying to build domain ontologies and have met with very limited success precisely because of the difficulties involved in people reaching agreements regarding concepts in the domain of interest. For example, consider an interaction between two medical care providers. The admission process for both providers involves creating or retrieving a patient medical record and adding information about the current admission to the record. However, in the admission process one provider (Provider A) performs blood screening prior to actual admission of the patient, while in its admission process the other (Provider B) performs the screening after the patient has been physically admitted to the facility. Even though both have the same understanding of the patient medical record (semantic interoperability), the difference in workflow could create inconsistencies if the patient goes through the admission process with Provider B (which does not perform the test) and then is transferred to Provider A. If the workflow is completely automated but not organizationally interoperable, Provider A will assume that the blood test has been performed. This inconsistency in workflow order could have dire consequences if the patient is a carrier of a highly infectious blood-borne disease.

Expectations for interoperability will continue to increase and exceed the capabilities of standards. Any attempt to standardize workflows and business processes across multiple organizations will be short-lived because the organizations, and their workflows and business processes, are constantly changing.

4.2. Quality of Service

The term quality of service (QoS) describes characteristics of a system that are observable at runtime. Examples of QoS attributes include performance, availability, and security [14].

Interoperability has a syntax and semantics component, and a QoS component, and all are important. For example, if one system needs a response from another system within 1 second, but the second system can only guarantee a response time of 5 seconds, then these systems are not interoperable with respect to this QoS requirement.

Currently, most QoS characteristics are managed in an ad hoc manner that is not based on standards because there are few standards that address QoS. One important area where standards and standards-conformant solutions are becoming available is security in Web services. There are standards available that define how to attach security information to SOAP messages (WS-Security), how to establish and maintain trust between participating service providers (WS-Trust), and how to establish secure sessions (WS-SecureConversation). Another standard (WS-SecurityPolicy) that is currently under development will allow service providers and consumers to specify the level of service required and provided.

In other areas, there are no standards available, and there have been only few attempts to formalize QoS [7]. As a result, it is necessary to introduce individual agreements between providers and consumers that govern these issues. This is usually done in form of service level agreements. A service-level agreement (SLA) is a contractual agreement that covers certain services and includes agreed-upon QoS characteristics, how to measure them, and the level of service expected and provided. For example, an SLA may specify that for a system the mean time between failures is more than 60 days, and that the mean time to repair is less than 4 hours, resulting in an availability of 99.7%.

Individual service level agreements help, but they do not address global QoS of a system as a whole. Reasoning about system-wide QoS is particularly difficult because QoS characteristics often do not add up in a simple way when systems are combined. The functionality of a system can be viewed as the sum of the function of all interacting systems. However, in addressing QoS needs, a different type of analysis is required. For example, availability needs to take into account how functionality is replicated in the system; security needs to account for analyzing all possible data flows to evaluate privacy of this data. It is also not always obvious how to improve QoS. For example, decreasing the response time of one system may or may not decrease response time of systems using the improved system.

Recently, QoS issues have received renewed attention in the area of service-oriented architecture, because service-based systems are generally distributed systems. Approaches are emerging to handle the combination of QoS in a service-based environment,
but there is no standardized way of handling these issues [8].

In addition, QoS depends on many other factors, such as topology of the connections between systems and resource sharing among systems. The topology may change dynamically, which further increases the difficulty of determining QoS characteristics. In some situations, it will be impossible to even know exactly which nodes participate in a network at any given time because nodes come and go too frequently, such as in a wireless network that connects mobile devices.

5. Why Standards Will Never Be Enough

The previous section focused on how current standards do not fully address needs of interoperability. This section suggests reasons why standards will never be enough by themselves to address end-to-end systems interoperability.

5.1. Extensions and Customizations

Ideally, every implementation of a standard should be identical and thus completely interoperable with any other implementation. However, this is far from reality. Standards, when incorporated into products, tools, and services undergo customizations and extensions because every vendor wants to create a unique selling point as a competitive advantage. For example, there are multiple variations in the implementation of the structured query language (SQL) standard in different database products. This means that a set of SQL statements written for one database is not completely compatible with other databases. In fact, they may not even be compatible with older versions of the same database.

At other times, standards are deliberately open-ended and provide extension points [6]. The actual implementation of these extension points is left to the discretion of implementers, leading to proprietary implementations. Finally, organizations sometimes deliberately refrain from following standards and implement their own alternatives. Tibco’s EMS and IBM’s MQ are examples of such proprietary implementations of messaging solutions for distributed systems.

5.2. Life Cycles of Standards

Standards, like any technology, have a life cycle of their own. Numerous popular standards are born from good practices that become dominant and are eventually accepted as standards by the community. Such standards are called de facto which is a Latin expression that means “in fact” or “in practice” [6]. TCP/IP, PostScript page description language for laser printers, and Microsoft Word document format are some examples of some popular de facto standards. Some de facto standards are good and others become standards only because they were first and not necessarily the best. Unfortunately, organizations cannot economically justify migrating to better standards and are often locked into such standards.

Standards created by standards organizations or committees tend to be more formalized, controlled, and documented. These standards are known as de jure or “by law” standards [6]. The Health Level Seven (HL7) [http://www.hl7.org] standard that is used for sharing and exchange of electronic health information is an example of such a standard. The problem with some de jure standards is that they are not based on real-life experience and requirements, which significantly reduces their ability to provide a solution when actually implemented.

Standards evolve and sometimes are not even backward-compatible with previous versions. Common examples are SOAP 1.1, SOAP 1.2, HL7 2.x, and HL7 3.0. Even though SOAP 1.2 is the latest version, most products still implement SOAP 1.1 because it would require many changes to upgrade the products to the next version. Version incompatibility between a group of complementary standards and products is yet another issue that vendors and organizations have to face. For example, a vendor decides to upgrade its product to a newer version of a standard. This may cause incompatibilities with products from other vendors used inside an organization.

As a result, a critical issue for many organizations concerns deciding when to adopt a new or revised standard [10]. Committing to a new standard that is not ready or eventually not adopted by the community is a big risk for organizations. On the other hand, waiting too long may also become a problem due to unsupported products, incompatibilities, and workarounds because everyone else is using it.

6. Can Standards be Too Much?

Because of the demonstrated value of standards, there is often a push to create or adopt standards under imperfect circumstances. However, in some cases, creating or adopting specific standards is unwise. We
argue that a standard may be too much in the following cases:

- when the available standards are bad standards
- when available standards conflict
- when the available standards limit flexibility

### 6.1. Bad Standards

Within the software community, there are as many bad standards as there are engineers with opinions and probably a greater number of reasons. Here we will try to categorize reasons why specific standards are sometimes considered bad. Categories of “bad” standards include:

- **under specification**: The standard addresses only a small part of the problem or allows so much variance in implementation that it fails to achieve the desired result. This is common among standards that purport to establish interoperability among technologies, particularly when the various standards proposed have powerful, but different, advocates. The RS232 standard for serial communications was underspecified which resulted in a wide variation in the serial cables.

- **over specification**: The standard leaves inadequate flexibility for different implementations or is tailored to the implementation of a subset of vendors. As a result, some vendors may chose to ignore the standard, thus defeating the intent.

- **inconsistently specified**: The standard is specified such that there are internal inconsistencies within the standard, sometimes due to variation in the level of detail in parts of the standard, and at other times due to internal conflicts that render the standard unusable. For example, UML 1.1 diagrams produced using different implementations were not always consistent.

- **unstable**: The standard is undergoing significant modification that will interfere with an organization’s attempt to establish a stable baseline of code based on the standard. Even well-respected standards are often unstable early in the specification process. Moreover, stable standards can become relatively unstable due to major modifications to expectations with new versions. For example, consider the set of HL7 standards that support communication among medical systems. Previous versions of HL7 (e.g., 2.3, 2.4) were focused on providing a standard message format to pass information between systems, which maintained their own unique information models. The problem, of course, was that it was not easy to map between information models of the various systems. This problem is addressed in the most recent release from HL7, 3.0, which adopts a reference information model (RIM). However, while critical for achieving interoperability among medical systems, adopting 3.0 and the RIM will require vendors of medical systems to extensively modify their applications. Needless to say, this is causing push-back from vendors, who are not only worried about the changes they have to make but also about whether the standard, which must be adopted by the wider community, is stable enough to adopt.

- **irrelevant**: The standard is not needed, does not solve a legitimate need, or does not have the necessary support to be viable. For example, the Portable Common Tool Environment (PCTE), a standard for establishing collaboration among tools used to construct software developed in the early 1990s, was implemented by very few software tool vendors and quickly died.

### 6.2. Conflicting Standards

It is quite common for standards to be pushed by competing organizations. In other cases, standards are developed by different organizations that do not view themselves as competing, but still produce standards that conflict in some way. Ways in which standards can conflict include:

- **overlap**: Multiple standards, all of which are viable, have a subset of features that are addressed in other standards. This requires users of standards to determine which features of which standards will be adopted, and to ensure that the conflicts do not lead to problems with development of the system. This mapping of features by standards users is considered part of a standards profile that reflects how an organization is applying various standards.

- **mutually exclusive standards**: Some standards only work with other standards, in that the selection of one standard precludes the use of another standard. This does not imply that a standard that leads to another being precluded is bad or that the precluded standard is bad—just that they are mutually exclusive. In this case, the standard’s user clearly has to make a decision as to what set of compatible standards provides the best overall solution.

- **competing standards**: Because they are so common, competing standards—one of the most common cases of mutually exclusive standards—
are treated separately here. Competing standards commonly arise from differing standards groups that are in direct competition. A well-known example is that of VHS versus BetaMax for taped video recording, a competition that is now being replayed between HD-DVD and Blu-Ray for next-generation DVD technology. Within the software community, OpenDocument format (ODF), developed by the OASIS consortium, has, at least until recently, been considered in competition with Microsoft’s Office Open XML formats. Both formats are intended to support the saving and exchange of office documents, such as reports, spreadsheets, and charts. Recently, Microsoft stated that it would finance construction of plug-ins to allow Office tools to save to ODF formats.

### 6.3. Inflexible Standards

For new and rapidly emerging domains, the argument is often made that standardization will actually be destructive because it will hinder flexibility. This argument commonly takes two forms: (1) in some cases, it may be too early to standardize, and (2) wide-scale standardization may be inappropriate for large-scale systems and networks of systems.

Those that believe that it is too early to standardize take the position that we do not know enough about a given domain or technology to standardize at the current point in time. It follows that premature standardization will force the use of an inadequate approach and lead to abandoning other presumably better approaches.

The more extreme position that wide-scale standardization is almost always inappropriate is taken by some proponents of highly dynamic capabilities that must respond to constantly changing environments. For proponents of this argument, almost any detailed specification of network-wide behavior is bound to be too constrictive. This argument claims that smaller, locally controlled standards are needed, with only minimal rules and standards widely adopted to maintain connectivity. This view sees most standards as a barrier to the extreme flexibility that will be required for the highly dynamic systems of the future, such as the DoD’s Global Information Grid.

These views express a caution relative to the dynamic environments of the future. The key for achieving the required interoperability while maintaining flexibility will be to allow domain groups to develop standards as they believe appropriate, while maintaining only a critical core of cross-domain standards that achieve connectivity and communication of semantic information. This contrasts with the approach taken by some large organizations that try to enforce enterprise-wide standards. Within such organizations, there is an unfortunate tendency to apply standards beyond their initial range of intent.

### 7. How Can We Address The Problems?

Despite the limitations of standards, organizations can take concrete actions to address some of the risks outlined above. These actions include identifying required levels of interoperability, understanding relevant existing standards, analyzing the gaps in standards, and taking measures to fill in the gaps. We provide recommendations for taking these actions in the subsections below.

#### 7.1. Identify the Level of Interoperability Required Between Systems

Not all interoperability problems will require addressing all levels of interoperability. For example, what follows are three situations in which an organization needs to obtain weather data from another organization. The examples show that even though the data is the same, the level of interoperability required depends on how the information is acquired and used.

1. An organization uses weather data from a weather Web service for its enterprise portal. This case requires Level 1 and Level 2 interoperability provided by the Web service standards and minimal Level 3 interoperability provided by the Web service description to understand the information received from the Web service so that weather data is appropriately presented to the portal user. This understanding can happen at design time, and if data is not appropriately represented there is no major impact.

2. An organization needs constant access to weather data. It does this by searching a service registry for weather services at runtime and linking to the first one available from the returned list. If the service fails, it tries with the second on the list and so on. This case requires Level 1, Level 2, and Level 3 interoperability because a common understanding of weather data has to be shared between service providers and consumers. Because this understanding happens at runtime, it is crucial to at least agree on the semantics of the service input and outputs.

3. An organization collects weather data from sensors and uses it for disaster prediction and re-
response. This case requires all levels of interoperability because data on the weather as well as its collection and use need to be standardized. If weather data is not accurate there may be a risk to the organization’s mission.

In order to determine the level of interoperability required for an organization, we recommend using end-to-end scenarios that illustrate where interoperability is required between systems. End-to-end guarantees the inclusion of elements that are outside of the simple interactions between systems and would include any requirements for Level 3 and Level 4 interoperability.

7.2. Understand Existing Standards

Areas in which standards are required need to be identified in the end-to-end interoperability scenarios. For each of these areas, existing standards that can address interoperability requirements need to be identified and studied. Because of the importance for maturity and stability of standards, it is important to answer the following questions for each standard.

1. How widely adopted is the standard in industry?
2. Are there domain-specific extensions or specialization of the standard?
3. What vendors and products support the standard?
4. How stable is the standards governing body?
5. Does the standard have the backing of the major industry players?

7.3. Analyze the Gaps in the Standards

It is now necessary to understand what the standards can do and what they cannot do. Contextual experimentation that exercises the identified end-to-end scenarios will help to understand the gaps in the standards and can start characterizing any solutions to fill in the gaps [16].

T-CheckSM is a technique developed by the SEI for contextual technology experimentation that is based on very simple, focused experiments to validate a technology’s claims [9]. The T-Check approach can be easily applied to the evaluation of standards. The context for the experiments is provided by an interoperability scenario. Hypotheses are developed based on claims made by standards bodies or vendors or perceived gaps in the standards. Simple examples of hypotheses are that images can be transmitted in a SOAP message or that WS-Security can be used in combination with Security Assertion Markup Language (SAML) to provide single sign-on. These hypotheses are then examined against very specific criteria using the simplest experiment possible to validate or refute the claims or gaps. The result is the list of proven identified gaps in the standards that invalidate the end-to-end scenarios.

7.4. Fill in the Gaps in the Standards

Once gaps are identified, these gaps need to be filled. Based on the results of the evaluation there should be enough information to determine if

- The standard(s) is (are) a good fit for the scenario requirements.
- The standard(s) is (are) not a good fit with scenario requirements.
- The standard(s) has (have) some gaps that could potentially be solved.

If the solution provides information that sustains the set of hypotheses, assuming sufficient confidence in the results, the standards can be declared a good fit with the scenario requirements. Given that the solution is a simplification of the real solution, it should be determined whether the results can scale so that they are valid in the larger context as well.

If the solution provides information that refutes the set of hypotheses, and there is no room for negotiation or very strict constraints exist that are not met, it can be stated that the standard is not a good fit with the scenario requirements. In this case, it may be possible to find a different standard that will satisfy the requirements. The process should then be repeated, reusing the previously produced information and results as much as possible.

Most often, the solution will provide information that sustains some of the hypotheses and refutes others. In case of gaps, it is useful to determine if there are ways to fill in these gaps. Some potential ways to fill the gaps include

- reset technology expectations.
- negotiate scenario requirements with stakeholders.
- investigate complementary standards that in combination would satisfy the hypotheses.
- negotiate the possibility of performing modifications to the standard with the standards organization. The problem with this approach is that there is usually a formal process to put forth modifications that can involve a lengthy approval time.
- localize standards dependencies in the system architecture. This is especially important if the iden-
tified standards are emerging or unstable. The localization of standards dependencies in specialized architectural components will isolate the system from changes or replacements of standards.

- create a stack. This is a term that is commonly used to refer to a set of products and standards that work well together. An additional advantage is the unification of computing platforms across the organization, which in turn leads to potential reduced license costs.
- if possible, participate in standards organizations. This will provide more input into the development and approval of standards that will better satisfy the organization’s interoperability requirements.

8. Conclusions

There is general agreement on the need for standards to achieve interoperability. A good example of this is the Internet, where due to standards we are able, in most cases, to use any browser on any platform to navigate the World Wide Web. However, this is a case where machine-level and syntactic level interoperability (Levels 1 and 2) is enough for what users expect of the Web. There will be situations that require higher levels of interoperability, especially when multiple organizations are involved and there is a desired degree of automation in the interactions between organizations. We call these higher levels of interoperability semantic and organizational (Levels 3 and 4).

Even though an ideal situation is to have standards at every level, there will be situations where it is not necessary and others where it is simply not possible due to the lack of appropriate technology or the impossibility to reach an agreement between too many stakeholders. What is important is to understand the advantages, limitations, and risks associated with standards and to carefully plan a strategy that will help mitigate the risks. Standards need to be analyzed within the organizational context and the level of interoperability required, using contextual experimentation in cases where dependencies on a standard are crucial to success. Finally, organizations must understand that standards are necessary but not sufficient to guarantee end-to-end interoperability between systems. There will always be a part of the problem that is outside of the scope of what standards can do. Finding the best way to address this gap is the real challenge.

9. References


10. Copyright Terms

NO WARRANTY
THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This Paper was first published in the Seventh International Conference on Composition-Based Software Systems Proceedings published by IEEE. To obtain a copy of the full proceedings, please visit http://ieeexplore.ieee.org/Xplore/dynhome.jsp.

This paper may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.