Open Standards - Open Source

The Business, Legal, and Technical Challenges Ahead

June 24-25, 2003, Minneapolis, MN, USA

www.opengroup.org

THE Open GROUP

Open Source Standards

John Schmidt
Methodology Committee Chairman, EAI Industry Consortium

The great thing about software and standards is that there are so many to choose from. So goes the familiar adage. This statement regularly elicits snickers from IT professionals. But why? Aren't standards *good*? The issue is that we've got multiple proposed standards for the same problem, so how can it be a *standard* if there is more than one?

Part of the problem may be definitional. A generic dictionary definition is "something established by authority, custom, or general consent as a model or example". I propose the following definition which is more explicit for information technology:

> ***Standard**: A specific category of information technology that is a) defined by an open (public) specification, b) governed by an egalitarian (democratic) organization, and c) in use for at least one year by >50% of the applications in it's class.*

Since few technologies meet this test, we might want to add a few qualifiers. If a given technology satisfies (c), but not (a) or (b), then it is a *de facto* standard. If it meets criteria (b), but not (a) or (c), then it is a *proposed* standard. If it satisfies (a), but not (b) or (c), then it is an *emerging* standard. For a given technology to be considered a standard without qualifications, then it must meet all three criteria. With this stricter definition, most of what many refer to as standards would demand some form of qualifier. De facto standards include Windows, COM, and certain Java protocols. Proposed or emerging standards include MDA, XMI, XLANG, WSFL, BPML, SAML, XKMS, UDDI, ebXML, WSDL, and SOAP (you get the idea).

Examples of unqualified standards include HTTP, SSL, SMTP, FTP, Apache, and TCP/IP. Admittedly, even with a more strict definition, there is still lots of room for subjectivity. Achieving >50% usage is subject to the segmentation and classification scheme used and the accuracy of the market study. For example, is JMS in the category of Enterprise Messaging Middleware or in the category of Java Component Communications? The level of granularity and how you slice the pie can drive different results. Or look at Apache; it commands well over 50% of the web-server market, has a public specification (source code available to all), and is governed by an open foundation (a meritocracy actually). Although the traditional definition of standard would not have included Apache, it meets all criteria of my proposed definition.

There are many hurdles to achieving a standard. The "not invented here" syndrome, proprietary intellectual capital, and competitive pressures all conspire to create a never-ending stream of new standards. And it doesn't help that end-user organizations have, in general, abdicated responsibility for establishing standards. Take a look at the board, architectural committee, or sub-committee of any major standards organization and you will see that virtually 100% of *active, involved* participants are vendor organizations. Oh sure, end users are members of the standards bodies, but they typically don't take an active role on the committees.

One of the opportunities for open source standards driven by end-users may be in the development of interface standards. While many end-user IT departments have drastically reduced their level of in-house application development over the past 10 years as a result of increasing use of commercial-off-the-shelf software, the task of integrating the systems remains. Many IT departments are finding that much of their in-house development effort is around building and sustaining interfaces, and they are starting to create highly functional and reusable frameworks to solve the problem.

Many people believe that formal specifications should precede implementation. But since a true standard is only formed after broad acceptance, I would argue that an empirical process, rather than an analytical process, is a perfectly valid - and often better - way to achieve the goal.

Attribute	Traditional Approach	Open Source Approach
Leadership	Vendors	Users and developers
Validation	Analysis	Trial and error
Acceptance	Agreement	Empirical use
Definition	Specification (document)	Software (code)
Motivation	Sales and marketing	Production operations

As the chart highlights, there are two basic approaches for achieving standards. The traditional approach is driven by vendors through standards bodies, validated through analysis, accepted by agreement (a political process), delivered as a specification document, and ultimately motivated by market share objectives. The open source approach on the other hand is driven by developers and users, validated by an evolutionary process, accepted through successful production use, delivered as software, and ultimately motivated by addressing operational needs.

The marketing mechanism of the global open source community is described by Bruce Perens, a consultant and open source evangelist for many years, as *a massively parallel drunkards walk, filtered by a Darwinistic process*. The open source community may be sobering up. By adopting software such as Eclipse, JUnit/NUnit, Ant/Nant, and Linux, user organizations are creating the standards of tomorrow by supporting the open source projects of today.