CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Best Practices for the Development and Use of XML Data Interchange Standards

By

William Behrman

CIFE Technical Report #131
July, 2002

STANFORD UNIVERSITY

If you would like to contact the authors, please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

Best Practices for the Development and Use of XML Data Interchange Standards

William Behrman
Department of Civil and Environmental Engineering
Stanford University

Last modified: 2002 July 26

Contents

1. Overview

When you consider what computers, software, and networks are made of, you may think of chips, circuit boards, lines of code, and cables. What you cannot see, but what is absolutely essential to the functioning of information technology, is a myriad of standards. Chips have standardized pin layouts so that they fit into different manufacturers' circuit boards. The lines of code are written in standardized programming languages to allow software to run on different manufacturers' computers. The cables that make up networks transfer bits using a range of standardized protocols that allow, for example, an email sent from one computer to be delivered to computers all over the globe.

The World Wide Web is based upon a very successful set of standards. The Hypertext Markup Language (HTML) provides a standard medium for exchanging documents, and its use to transmit information from computer to human has characterized the first phase of the Web. The Extensible Markup Language (XML) provides a standard medium for data interchange, and its use to transmit information from computer to computer is the basis for what many feel will be the next phase of the Web. XML is only the basis, however. It is a language in which data interchange standards may be written, and the development of such XML standards has only recently begun.

Though the development of XML data interchange standards is relatively new, the development of information technology standards has a long history, and in Section 2 we examine best practices for the development of such standards, which have been applied across a wide range of technologies. In Section 3, we examine how these best practices have been applied in one of the most successful XML standardization efforts thus far, RosettaNet. In Section 4, we examine the mixed results of standardization efforts in the architecture, engineering, and construction (AEC) industry. Finally, Section 5 contains conclusions and recommendations that apply to the AEC industry and more generally.

2. Best Practices for the Development of Information Technology Standards

The development of information technology standards has a long history. Scholars have studied past standardization efforts, the different approaches taken, and the relative success of these different approaches. In this section we will examine the lessons of the past, the different approaches, and especially the strategies that have been most successful.

2.1 Desirable Results in the Development of Information Technology Standards

Before we compare different standardization efforts, let us consider how to measure success. A successful standard has several characteristics. First, it solves the problem it was meant to solve, for instance to enable *interoperability,* allowing two systems to work together, or to enable *data exchange,* allowing information generated by one system to be read by another. Second, it is developed in a timely manner. Third, it achieves widespread adoption and use. Finally, it anticipates and allows for future technological change, constraining future development as little as possible.

2.2 Different Approaches to the Development of Information Technology Standards

Standardization efforts are characterized by very different approaches.

The *minimalist* approach values simple standards and rapid adoption by the user community. It is a bottom-up approach in which standards start small. The development process places heavy emphasis on experimentation, testing, and iterative improvement of proposed standards in applications before adoption. Once such standards are adopted and gain acceptance, they are further developed as needed. The protocols that underlie the Internet, such as TCP/IP, were developed with this approach (see Section 2.3.2). HTML, the C programming language, and the SQL database language were also developed in this manner.

The *structuralist* approach values comprehensive and complete standards. It is a top-down approach. The development process starts with a high-level model and then proceeds with the elaboration of more and more detail. The process is often daunting and time-consuming. The Open Systems Interconnection (OSI) has taken this approach (see Section 2.3.1). The Ada programming language, Integrated Services Digital Network (ISDN), and the Standard for the Exchange of Product model data (STEP) have also taken this approach.

2.3 Examples

It is instructive to examine two standardization efforts that were directed at the same goal: providing data communications over a network. The Open Systems Interconnection (OSI), an initiative of the International Standards Organization (ISO), took a structuralist approach that consumed decades of work with only limited success. The Internet, on the other hand, was developed using a minimalist approach that has been very successful. In this section, we will look at these efforts and the approach that each took.

2.3.1 Open Systems Interconnection (OSI)

In 1977 the International Standards Organization (ISO) began work on a comprehensive reference model for data communications, the Open Systems Interconnection (OSI). In 1980 the reference model was finished, and in 1983 it was formalized. The model specified seven layers:

- Application
- Session
- Presentation

- Transport
- Network
- Data-Link
- Physical

Individual standards would be developed for each layer. A standard for one layer would have to interact only with the standards on the layer above or below. It was envisioned that different standards for the same layer could be used interchangeably.

The Institute of Electrical and Electronic Engineers (IEEE) supplied local area network standards to OSI. The International Telecommunication Union (ITU) supplied the X.25 packet switching standard and the X.400 electronic mail standard. Other standards to fill out the model were developed over many years by the ISO itself. Throughout the development of the OSI standards, the effort was endorsed by governments and leading computer vendors. Over the years, billions of dollars were spent on the effort. In the end, however, despite the high-level endorsements and the enormous investment of time and money, OSI has largely lost out to the Internet set of standards. What happened?

The OSI standards took far longer to develop than the Internet standards. Vendors who needed protocols upon which to build products observed that the delivery date for the OSI standards always seemed to be two years in the future. In the meantime, the TCP/IP Internet protocols were available in software that was tested and worked. So vendors built products upon TCP/IP. Many described their choice of TCP/IP as a temporary measure, that they would switch to the OSI standards when they were ready. Time went on, and the products were further and further developed. When the OSI standards finally became available, most vendors did not switch. Even though the OSI Transport and Network protocols drew heavily upon TCP and IP, changing the standards upon which software is built can be very difficult -- it is analogous to changing the foundation upon which a building is built. Other reasons were perhaps more important, however, in explaining why vendors did not switch to the OSI standards once they became available.

The OSI standards are complex, difficult to implement, and laden with features that were included without market feedback. The structuralist approach is to develop comprehensive and complete standards. When members of a standardization committee try to anticipate all the features that they think users might want, a standard can grow large and complex. Features can be added without an appreciation of the cost and difficulty of implementing them, and without market feedback regarding their desirability. The resulting OSI standards were very difficult to implement, placed a heavy burden on computing resources, and were often out of step with market needs. To produce code fully-compliant with all the features and options of a thousand-page standard could take a vendor years. Vendors who attempted to implement the OSI standards often chose to implement subsets of the standards, but different vendors chose different subsets, resulting in incompatibilities that thwarted the purpose of the standards. Implementation difficulties were an obstacle to the adoption of the OSI standards, but there were more fundamental issues.

The OSI standards, even when completely implemented, did not work as intended, and often were not solutions to the most pressing problems that vendors faced. Once the standards were implemented, some key deficiencies became apparent. The implementations required considerable computing resources, making them unsuitable for use with PCs and PC networks. The implementations would be of a "stack" of standards from the OSI layers. But some layers had more than one OSI standard, and these standards were not interchangeable in practice. This led to a number of OSI profiles in which a particular standard was specified for each layer. When different organizations chose different profiles, incompatibilities resulted that again undermined the purpose of the standards. Even when the implementations worked, they often were not solutions to the urgent problems that vendors faced. In the structuralist approach, there is a long lag between the beginning of development and the ultimate implementation of standards. With rapidly changing computer technology and market needs, it is easy to understand how the needs anticipated by standards committee members could differ from the actual needs many years later.

Further information on the development of the OSI standards may be found in (Libicki, 1995a, pp. 75-129).

2.3.2 The Internet

The standards that underlie the Internet were developed in a very different way from that of the OSI. Those who developed the Internet standards took a minimalist approach. They developed the suite of standards incrementally and experimentally, from the bottom up instead of from the top down. They focused on solving specific engineering problems, and required multiple implementations and substantial operational experience before they allowed any specification to become a standard. They produced standards in a timely manner. The Internet suite of standards that they developed has clearly been very successful. Below we will examine the two central factors for their success: the process they followed and who participated in this process.

The Internet Architecture Board (IAB) of the Internet Society (http://www.isoc.org) manages the development of Internet standards, using a process described in (Bradner, 1996). The process aims to produce standards, each of which "is stable and well-understood, is technically competent, has multiple, independent, and interoperable implementations with substantial operational experience, enjoys significant public support, and is recognizably useful in some or all parts of the Internet" (Bradner, 1996, p. 3). The goals of the process are:

- technical excellence
- prior implementation and testing
- clear, concise, and easily understood documentation
- openness and fairness
- timeliness (Bradner, 1996, p. 4)

Here openness and fairness refer to allowing all interested parties to comment on specifications in the various stages of their becoming standards. Timeliness can conflict with the other goals, but the process was designed "to be as short and simple as possible without sacrificing technical excellence, thorough testing before adoption of a standard, or openness and fairness" (Bradner, 1996, p. 4).

The Internet Engineering Task Force (IETF) (http://www.ietf.org) of the IAB is where the process primarily takes place, organized into nine broad areas. Working groups within each area manage the development of individual standards. Specifications intended to become Internet standards evolve through a set of three maturity levels: *Proposed Standard, Draft Standard,* and *Standard.* The Internet Engineering Steering Group (IESG), consisting of the area directors and the IETF chair, determines whether specifications advance from one level to the next, based upon the results of implementation, testing, and public comment.

The first maturity level is the Proposed Standard. It is intended for a specification that "is generally stable, has resolved known design choices, is believed to be well-understood, has received significant community review, and appears to enjoy enough community interest to be considered valuable" (Bradner, 1996, p. 12). Implementation and operational experience is not required but is "highly desirable," and it may be required for very important specifications. A specification remains at this level for at least six months to allow for public comment, which is accepted from anyone. Significantly, a specification at this level can not advance to the next without implementation and operational experience.

The next maturity level is the Draft Standard. It is intended for a specification "from which at least two independent and interoperable implementations from different code bases have been developed, and for which sufficient successful operational experience has been obtained" (Bradner, 1996, p. 13). Every feature and option of the specification must be implemented in at least two interoperable implementations. A specification remains at this level for at least four months to allow for public comment, which is again accepted from anyone. A specification at this level gains broader field experience, especially to "demonstrate unforeseen behavior when subjected to large-scale use in production environments" (Bradner, 1996, p. 13).

The benefits of developing a specification concurrently with its implementation and testing are illustrated by an incident in the development of the TCP/IP protocols. In 1980, when these protocols were still at the draft stage, ten vendors developed implementations to test interoperability. It turned out that only two of the systems could communicate with each other. The problem was that different vendors interpreted a part of the specification differently. So the specification was changed to follow the interpretation of the two working systems and to remove the ambiguity. The eight other vendors then simply changed their code to follow the clarified interpretation, and all the systems were interoperable (Libicki, 1995a, p. 90).

The final maturity level is that of Internet Standard. It is intended for a specification "for which significant implementation and successful operational experience has been obtained" and "is characterized by a high degree of technical maturity and by a generally held belief that the specified protocol or service provides significant benefit to the Internet community" (Bradner, 1996, p. 14). The Internet standards process was designed to produce proven standards that solve the problems they are intended to solve. The process uses the minimalist approach in which standards are developed in a decentralized manner to address specific pressing problems, instead of being part of a comprehensive grand scheme. In this way, the need for a solution to a pressing problem can enter the process and emerge with a standard, without being slowed down by all the other efforts.

Just as there is a minimum time a specification has to remain at a certain maturity level, there is also a maximum time it can remain at a level without review. When a specification has been at a level for 24 months, and every 12 months thereafter until its status changes, the IESG will "review the viability of the standardization effort responsible for the specification and the usefulness of the technology" (Bradner, 1996, p. 21). After the review, the IESG will determine whether to continue or terminate the development effort. In the past, the Internet standards process has been able to achieve its goal of producing standards in a timely manner, but that may be changing. The reason is that while the process itself has remained largely unchanged over the years, those participating in the process have changed considerably.

As the importance of the Internet became more and more apparent, the nature of those participating in the standardization process changed dramatically. In the early years, the participants were a small community composed largely of academics and computer scientists who knew each other. Though there was disagreement and debate, they maintained a collaborative spirit and provided rapid testing and feedback regarding each others' ideas. In the early years, the IETF semiannual meetings had on the order of 100 attendees; now they have 2000 (Libicki, 2000, pp. 21-22). Vendor representatives are now much more common, and the collaborative spirit has given way to much more competitiveness. As the number of participants in the process has grown larger, the average length of time it takes for a Proposed Standard to become a Standard has grown longer, averaging 5 years in 1993-1999 (Libicki, 2000, p. 22). And though the number of Proposed Standards has grown dramatically, the production of Standards has actually decreased, as the following table illustrates.

Internet Proposed, Draft, and Final Standards (by year)

	Proposed	**Draft**	**Final**
-1980	8		1
1981-1983	2		12
1984-1986	2		11
1987-1989	10	4	16
1990-1992	34	10	7
1993-1995	107	40	9
1996-1998	263	31	6

(Libicki, 2000, p. 22)

2.4 Achieving Desirable Results in the Development of Information Technology Standards

The minimalist approach, with relatively few people, produced the successful Internet standards, whereas the structuralist approach, with far more people, produced the unsuccessful OSI standards. These are not isolated examples. The other standardization efforts for each approach mentioned in Section 2.2 had similar results. In this section, we will examine the reasons why minimalist is successful in achieving the desirable results mentioned in Section 2.1 for a standardization effort:

- solves the problem it was meant to solve
- is developed in a timely manner
- achieves widespread adoption and use
- anticipates and allows for future technological change, constraining future development as little as possible

The minimalist approach is minimal in several different senses. In one sense, it divides the task of standardization into relatively small independent subtasks to solve independent subproblems. In another sense, its seeks for each subproblem a standard that is as small and simple as possible. With small, independent standardization efforts, each can proceed without being tied to and slowed down by other efforts. Each effort focuses on the solution of a specific subproblem, and each new standard builds incrementally upon other working standards. Standards developed to be independent solutions of discrete subproblems can also evolve independently. As technology or needs change, each can be changed or replaced as necessary, again without being tied to other efforts. Small, simple standards tend to be faster and easier to develop and implement than large, complex standards, and they tend to constrain future development less.

With standardization efforts focused on solving relatively small, independent subproblems, the minimalist approach allows and emphasizes implementation and testing during the development of standards. A specification is much more likely to solve the problem it was meant to solve if several different implementations are required actually to demonstrate that it is a solution before it becomes a standard. Implementation and testing during development can provide feedback on the utility of proposed features and on the difficulty of implementing them, while features can still be dropped, added, or changed. Implementation and testing can identify unforeseen problems while they can still be remedied. Once a final standard is adopted, its implementation time can be substantially reduced if it has been implemented during development. In the Internet standards process, for example, a draft standard is considered largely final. Very few changes will be required of an implementation of the draft standard for testing to create an implementation of the final standard. In contrast, the structuralist approach can take years to develop standards, followed by years to implement them, with how they will work remaining unknown all the while. The OSI took decades to produce implementations of standards that did not work as intended. Finally, having vendors involved in implementing, testing, and providing feedback during the development of standards aids in their ultimate adoption and use. By the time a final standard is ready, it has been proven, and vendor implementations are nearly ready.

A standardization effort is made up of both process and people. Teams that are relatively small, that have members representative of users and key vendors, that possess the necessary technical expertise, and that take a collaborative approach, have proven effective in the past at developing working standards in a timely manner. The right technical expertise is especially necessary in anticipating and allowing for future technological change. In contrast, large, international efforts populated with people of widely varying expertise can bog down, especially if the process become contentious.

Martin Libicki is a leading scholar in information technology standardization who has studied and written detailed analyses of scores of efforts, the approaches taken, what works, and what does not. He summarizes the lessons learned:

> Although any specific approach to standards must be sensitive to particulars of the relevant technology, applications, and markets, the one emerging from the standards community reflects these lessons: collect a small group of vendors, write a small, simple specification that covers the important functions, omit nonessentials, leave room for both new technologies and possible backtracking, identify real-world test-beds for the standard, and get it out the door as soon as possible (Libicki, 1995b, p. 75).

3. The Experience of RosettaNet

In February 1998, a significant new technology became available, the Extensible Markup Language (XML). XML solved a very important problem by providing a standard way to represent data for transfer over the World Wide Web. With XML as a foundation, other standards could be developed to allow software applications to interact over the Web. That very month, 40 information technology companies formed the RosettaNet consortium (http://www.rosettanet.org) to develop e-business standards in XML to improve supply chain management in their industry. The consortium has since expanded to include electronic component companies and semiconductor manufacturing companies; it now has over 400 members with combined revenues over $1 trillion. While numerous consortia are attempting to develop XML standards, RosettaNet is unique in the success that it has achieved in bringing XML supply chain standards into production use. Gartner forecasts that "through 2004, RosettaNet will be the only source of 'plug-and-play,' XML-based, consensus supply chain standards to have achieved more than token use in production (0.7 probability)" (Rishel, 2001). RosettaNet achieved its success by following a minimalist approach in developing its standards. Its experience and the reasons behind its success are instructive to other industries seeking to develop XML e-business standards. The standards that it developed are also worth examining; some are applicable across industries.

3.1 The RosettaNet Industry Context and Organization

RosettaNet began in 1998 as a consortium of information technology companies, expanded in 1999 to include electronic component companies, and expanded further in 2000 to include semiconductor manufacturing companies. A new industry segment is included only with the commitment and membership of companies that represent at least two-thirds of the overall trade for the segment and at least two-thirds of the trade for each type of trading partner within the segment. The nature of this industry is such that a relatively small number of large companies account for two-thirds of the trade. Members include such significant players as Intel, IBM, Sony, Microsoft, Federal Express, UPS, and the GSA. In this industry, supply chain management is very important, and all members stand to benefit from improvements that lower overhead costs and time delays. Member organizations typically possess a high degree of technological sophistication. They typically possess enterprise resource planning (ERP), supply chain management (SCM), and customer relationship management (CRM) systems, servers, networks, firewalls, Web servers, and e-business standards and guidelines; many have years of experience in electronic transactions using Electronic Data Interchange (EDI). In addition to industry participants, members also include solution providers who have committed themselves to extending the trading infrastructure by implementing the RosettaNet standards in products.

The RosettaNet members provide more than dues: they provide the substantial involvement of senior-level executives. From each member organization, RosettaNet typically seeks the involvement of the VP of information technology, the VP of operations, and the VP of e-business strategy. These executives serve on boards that set the priorities and direction of RosettaNet. Each of the three industry segments has a supply chain board that guides the development of standards for that segment. The solution providers also have a board to represent their interests. Finally, three members of each of these four boards together form an executive board for the overall management of the organization and for issues that apply to more than one supply chain. Through these boards, the executives align RosettaNet's efforts with industry needs. They put a heavy emphasis on milestones, and keep an internal "scorecard" of the implementation of RosettaNet standards by the members. Many industry executives have their bonuses tied to their company's success in implementing and using RosettaNet. The development of the standards is a large undertaking, requiring substantial resources. RosettaNet has a budget of about $10 million per year. It has 20-25 full-time employees and an additional 60-75 FTEs on loan from the member companies, working in a virtual organization.

3.2 The RosettaNet Standards Development Process

The RosettaNet standards development process puts to effective use the principles of the minimalist approach described in Section 2.4. One of the key success factors is the number and type of people involved. RosettaNet divides the process for each supply chain between senior-level executives, who set the direction, and technical experts, who follow this direction and actually develop the standards. Each supply chain has a board of executives representing its various trading partners. The board sets the priorities for the public business processes that need to be standardized, and provides milestones and guidelines for the deliverables. Each board is no larger than necessary to provide adequate representation of the supply chain trading partners, about 30-40 executives from the same number of companies. For the most recently formed board, RosettaNet contracted an outside consulting company to guide the selection of executives from representative companies. From the priorities and guidelines set by the supply chain boards, groups that will actually develop the standards are formed as independent projects. These groups have the technical expertise necessary for developing the standards, including intimate domain knowledge and proficiency in business process modeling. In contrast, there have been numerous unsuccessful XML e-business standardization efforts, which have possessed neither senior-level executive involvement nor people with the necessary technical expertise.

RosettaNet divides the work of standards development first among its three supply chain boards and then into a number of independent projects. This minimalist approach of dividing up the work allows for incremental and experimental standards development. A basic unit of development is the Partner Interface Process® (PIP®). The purpose of a PIP is to enable one discrete transaction between trading partners. Examples include: Request Price and Availability (PIP 3A2) and Request Purchase Order (PIP 3A4). A PIP includes both the necessary data and the choreography for the transaction between two systems. The data include items such as product number and price. The choreography describes the process of the transaction, such as the receiver acknowledging the request, the time allowed for a response, and the requester acknowledging the receiver's response. Once a PIP is developed, trading partners can implement it to allow their computer systems to automatically execute the transaction between themselves. The supply chain boards decide upon the transactions for which PIPs will be developed, and a PIP will be developed only if at least five board members and a subset of their trading partners agree to implement it. An advantage of the independent and incremental *development* of PIPs is that it allows for their independent and incremental *adoption* by trading partners, depending upon the unique needs and circumstances of each.

RosettaNet also applies the minimalist approach of implementing, testing, and getting feedback on its PIP standards during their development. It has a PIP Assembly Line Methodology to guide the PIP development process. With experience and the input of an outside consulting company, this methodology has evolved so that the development time for a new PIP has fallen from 9-12 months to 3 months -- very little time for the development of relatively complex standards in a process that involves consensus among trading partners, implementation, experimentation, testing, and public feedback. Those who participate in the development of the standards include operational managers, domain experts, technical specialists, and business process modelers who meet in workshops and collaborate over the Web. A considerable amount of the effort is in business process modeling. This modeling and the development of PIP choreographies was originally done by an outside contractor, but the expertise has since been brought in house. The difficulty in developing the PIP choreographies is that they are not simply electronic versions of current business processes; automation allows for improving and streamlining the business processes. For this reason, experimentation and testing are an important part of the PIP development process. Once a PIP is roughly 80% complete, it is implemented in a pilot environment for 30-60 days for testing, during which time anyone can provide feedback on it. In this experimental environment, the last -- most difficult -- 20% is completed. The solution providers, who participate in the process, facilitate it with tools that allow for rapid prototyping and testing of PIPs. This phase is comparable to the two cycles of implementation, testing, and feedback of the Internet standards process combined into one, with a carefully selected set of participants. With member companies and solution providers intimately involved in implementation, testing, and feedback, by the time a PIP standard is complete, its effectiveness will have been demonstrated in company implementations, and solution providers will be ready to serve as a channel for its spread to other companies.

3.3 The RosettaNet Standards

RosettaNet has employed an effective process to produce an effective set of standards, which have been implemented and are in use. The standards themselves, and how they are organized, are issues worth examining by those seeking to learn from RosettaNet's success. The standards are organized into an architecture that consists of nine layers:

- Universal Specification Schema and Architecture
- Supply Chain Business Processes
- Supply Chain Technical Dictionary Content
- Business Model Business Processes
- Universal Business Processes
- Universal Technical Dictionary Structure
- Universal Business Dictionary Structure and Content
- Universal Registry and Repository Structure
- Universal Messaging Service

Not all of the layers are filled in with standards; the industry executives guiding RosettaNet's development have taken a pragmatic approach of developing a minimal set of standards so they could be brought quickly into production use. The standards in some of the layers are evolving with experience. For some of the layers, other organizations are developing standards with the same function. RosettaNet has a policy of converging its standards with others that prove effective in production use and gain widespread support. XML is a relatively new technology, and XML e-business standards are rapidly evolving, but the current state of RosettaNet, which we will examine now, represents the current state of the art.

The Universal Messaging Service is the bottom layer of the RosettaNet standards architecture. E-business trading partners using RosettaNet interact via electronic messages, and this layer provides the means for such messages to be packaged, transferred, and routed over the Internet to their destinations. The **RosettaNet Implementation Framework (RNIF)** is the standard for this layer, and it provides an infrastructure for sending messages. Since security in e-business is of fundamental importance, RNIF provides for encryption and secure authentication, authorization, and non-repudiation through the use of digital signatures. Authentication establishes the identity of the sender of a message. Authorization determines whether the sender is permitted to send the particular type of message. Non-repudiation is a mechanism for sending messages in which the senders can not deny having sent them and the receivers can not deny having received them. Digital signatures are also used to detect tampering with messages or their corruption in transit. The sorts of issues that a messaging standard addresses are industry-independent, and it is therefore sensible to have only one cross-industry messaging standard instead of multiple standards. The ebXML Message Service Specification, released in May 2001, was designed to be such a standard (http://www.ebxml.org). It built upon RNIF, and, in an example of standards convergence, future versions of RNIF will include support for it.

The Universal Registry and Repository Structure is the next layer up in the RosettaNet architecture. The idea of a registry and repository is to create an online database in which companies and their services can be listed and discovered by other companies. Ideally, such a database would allow companies' e-business systems to query in search of potential trading partners, and it would have the necessary information to enable the systems of two trading partners to automatically interact. RosettaNet has not put much effort into developing a standard for this layer. Following the priorities of the supply chain boards, the first set of standards developed by RosettaNet have enabled fixed links between established trading partners, and the first links have been predominantly between large companies high in the supply chain. However, as the board executives look to extending the reach of RosettaNet down the supply chain and to smaller companies, these priorities are changing, and this layer is now receiving attention. The Universal Description, Discovery, and Integration (UDDI) effort (http://www.uddi.org) is a separate standardization effort for this layer that has gained considerable momentum, and RosettaNet is planning to build upon UDDI in developing a standard for this layer.

The Universal Business Dictionary Structure and Content layer and the Universal Technical Dictionary Structure layer are the next layers up. The **RosettaNet Business Dictionary** and the **RosettaNet Technical Dictionary** are the respective standards for these layers. The dictionaries contain specifications for data that appear in the higher layers. Each specifies the structure for how its data will be represented and organized. Each is then "populated" with content following this structure. The content for the Business Dictionary contains data entities for common business transactions such as BillingStatement, Confirmation, Invoice, OrderStatus, and PurchaseOrder. The Technical Dictionary contains a structure for technical data, but not content. Since there is a vast amount of technical content, and since it is rather specialized, RosettaNet takes the minimalist approach of leaving it up to each supply chain to develop this content as needed. For example, to buy and sell resistors, technical content would be developed to represent the properties needed by the trading partners. The technical content developed for the supply chains is contained in a higher layer. Outside of RosettaNet, other efforts have focused on developing XML data dictionaries for common business documents. Proprietary examples include Commerce XML (cXML) (http://www.cxml.org), developed by Ariba, and XML Common Business Library (xCBL) (http://www.xcbl.org), developed by Commerce One. Efforts are also under way to develop XML dictionaries from dictionaries for the older Electronic Data Interchange (EDI) technology. In September 2001, the OASIS industry consortium began an effort to synthesize these various business dictionaries, including RosettaNet's, into a Universal Business Language (UBL) (http://www.oasis-open.org/committees/ubl/). If successful, this will be another example of standards convergence.

The Universal Business Processes layer and the Business Model Business Processes layer are the next layers up. The first is for business processes between trading partners that are common across supply chains and business models. The second is for business processes between trading partners of a particular business model. A business process consists of both content, such as that from the dictionaries, and choreography. RosettaNet has spent less effort in developing standards for these layers than for supply chain specific standards, and, as a result, development is not as far along.

RosettaNet standards in the layers from the bottom Universal Messaging Service layer up to the Universal Business Processes layer are not industry specific and may be used in any industry. Those for a given business model in the Business Model Business Processes layer may be used by trading partners in any industry that follows that model. The standards for the next two layers up are specific to the RosettaNet supply chains.

The Supply Chain Technical Dictionary Content layer and the Supply Chain Business Processes layer are the next layers up. The first layer consists of supply chain specific technical content. The second layer consists of the **Partner Interface Processes (PIPs)**, whose creation has been the bulk of RosettaNet's development effort. As mentioned above, each PIP is a basic unit of development. Its purpose is to enable one discrete transaction between trading partners, such as Request Price and Availability (PIP 3A2). It contains both the necessary data, such as business and technical content, and choreography for the transaction between two systems. To organize and facilitate the independent development of the PIPs, RosettaNet has subdivided the task into categories. This is similar to the way in which the IETF has divided up the development of Internet standards. The categories are as follows.

Partner Interface Process (PIP) Categories

1. RosettaNet Support
    A. Administrative
2. Partner, Product, and Service Review
    A. Partner Review
    B. Product and Service Review
3. Product Information
    A. Preparation for Distribution
    B. Product Change Notification
    C. Product Design Information
4. Order Management
    A. Quote and Order Entry
    B. Transportation and Distribution
    C. Returns and Finance
    D. Product Configuration
5. Inventory Management
    A. Collaborative Forecasting
    B. Inventory Allocation
    C. Inventory Reporting
    D. Inventory Replenishment
    E. Sales Reporting
    F. Price Protection
6. Marketing Information Management
    A. Lead Opportunity Management
    B. Marketing Campaign Management
    C. Design Win Management (Electronic Components Supply Chain)
    D. Ship from Stock and Debit (Electronic Components Supply Chain)
7. Service and Support
    A. Provide and Administer Warranties, Service Packages, and Contract Services
    B. Provide and Administer Asset Management
    C. Technical Support and Service Management
8. Manufacturing
    A. Design Transfer
    B. Manage Manufacturing Work Orders and Work-in-Process
    C. Distribute Manufacturing Information

As of December 2001, 71 PIPs had been completed, and 250 companies had implemented at least one PIP. In total there were an estimated 630 connections, where 1 PIP used by 1 company = 1 connection. A large percentage of these connections were concentrated in the 72 largest companies. An obstacle to the wider implementation of the standards is the expense: the current set-up cost to a company is about $500,000. To make it easier for the small- and medium-size companies who are mid-tier trading partners to use the standards, RosettaNet has begun a Basics program to reduce this cost to under $50,000.

The Universal Specification Schema and Architecture is the top layer. The purpose for this layer is to specify how all the various layers and their standards fit and work together. Using the minimalist approach, RosettaNet is first getting the experience of how its standards are working together before it specifies this layer. Just as individual standards evolve through testing during their development, the overall suite of standards is being tested through its use, and its organization and design are evolving. A pioneering effort based upon rapidly changing technology requires flexibility, and it is well suited to the incremental, experimental approach that RosettaNet has taken.

3.4 Summary of the Reasons for the Success of RosettaNet

The production use of the RosettaNet standards dates only from October 2000, and electronic trade using these standards represents only a tiny portion of the trade in its industry segments. Nevertheless, this modest success far outstrips that of XML e-business standardization efforts for other industries. This success draws from the nature of RosettaNet's industry and its leadership, and it flows from the approach that RosettaNet has taken to standardization.

In the information technology, electronic component, and semiconductor manufacturing industry, supply chain management is very important, and a relatively few large companies account for a significant fraction of the trade. These companies posses a high degree of technological sophistication, and have internal computer systems that can support e-business. Realizing that e-business had the potential for substantial benefits across their supply chain, these companies created RosettaNet, and they have given it considerable resources to develop the necessary standards. They provide an operating budget that is currently about $10 million per year, and they provide key personnel, including the close involvement of senior-level management.

RosettaNet has effectively employed the principles of the minimalist approach described in Section 2.4 to develop its standards. Development is organized by supply chain, with boards of carefully chosen senior-level executives setting standardization priorities. Within a supply chain, work is divided into the development of minimal independent standards, each having a project team consisting of operational managers, domain experts, technical specialists, and business process modelers. This organization allows for the standards to be developed and used in an incremental and experimental manner, an approach especially well suited to such a pioneering effort. Member companies and solution providers participate intimately in the development process by implementing, testing, and providing feedback on proposed standards. By the time a standard is complete, its effectiveness has been demonstrated in company implementations, and solution providers are ready to serve as a channel for its spread to other companies.

4. Standardization Efforts in the AEC Industry

The AEC industry is very different from the computer industry that RosettaNet serves. The computer industry has a large portion of its trade concentrated among relatively few, large companies, such as IBM, Intel, and Microsoft. The AEC industry is much more fragmented, with relatively many, smaller companies. The nature of manufacturing computer equipment is also very different from that of producing buildings or roads. In manufacturing, supply chains are established for the ongoing production of hundreds or thousands of units, but in AEC each project is unique. Perhaps most importantly, in AEC, much of the potential for savings and creating value is in the early design and engineering phase and not through better management of the supply chain in the construction phase (Paulson, 1976). For example, the choice of a heating and cooling system for a building is likely to have a greater impact on the life-cycle cost of a building than the choice of how the system is procured and delivered to the construction site. In a sense, before there is a physical supply chain, there is a supply chain of professional expertise that links the various members of the AEC project team. This supply chain of expertise could be improved by increasing the ability of the project team's software to interact and exchange data.

Much of the standardization effort in the AEC industry has focused on facilitating data interchange between the software applications of an AEC project team. Below we will examine some of the key efforts. We will look at the International Alliance for Interoperability (IAI), which has taken a structuralist approach, and the Building Lifecycle Interoperable Software (BLIS) Project, which has tried to graft some minimalist principles on top of the IAI's effort. Finally, we will look at some independent efforts to develop XML schemas.

4.1 The International Alliance for Interoperability (IAI)

The International Alliance for Interoperability (IAI) (http://www.iai-international.org) is a large standardization effort for the AEC industry. The IAI draws its origins from Autodesk and a precursor group that was formed in 1994 and incorporated in 1995. The IAI currently has nine regional chapters, with members from North America, Europe, Asia, and Australasia. Each of these chapters is a separate legal entity with its own membership and budget. The IAI international budget, to which the chapters contribute, is about $124,000, down substantially from past levels.

The IAI has taken a structuralist approach. Its goal is

> to develop a **comprehensive conceptual model** of commercial buildings, defining physical elements such as walls, windows and doors, etc.; delineating processes such as HVAC operations and design intent; describing operational activities such as facility management/asset management (IAI, 2001a, p. 2). (Emphasis in original.)

The IAI standardization effort differs fundamentally from XML data interchange standardization efforts such as RosettaNet. It is attempting to develop a comprehensive *product model* representation of the data of the AEC industry, with a scope that includes the physical elements as well as processes and activities. The structuralist vision is to develop a comprehensive model of the data of the AEC industry, with the hope that AEC software applications would be able to interchange data by writing or reading to this model. Of course, it is much more difficult to develop a comprehensive standardization of the data of an industry than it is to standardize the minimal data necessary to achieve desired interchanges. The IAI started before XML was available, and its product modeling methodology and technology are from ISO/STEP. It takes an object-oriented approach, in which data are organized into a hierarchy of classes, with classes at higher levels inheriting properties from classes at lower levels. The IAI standards are called the **Industry Foundation Classes (IFCs)**. The IFCs are developed and represented in a product modeling language called EXPRESS.

Instead of a bottom-up approach of independent standardization efforts, the IAI has a top-down, centralized approach. The IFCs are developed by the Model Support Group (MSG), six people trained in the ISO/STEP product modeling methodology who collectively contribute approximately 3 FTEs of effort. Starting from an abstract, high-level model, the MSG works to develop and extend the IFC product model through elaboration. It periodically releases updates of the current state of its model. The object-oriented modeling methodology used by the MSG does not lend itself well to independent decoupled standardization efforts, the approach used by the developers of the Internet standards and RosettaNet. A product-model class hierarchy is tightly coupled though inheritance. If one class in the hierarchy is changed, all the classes that inherit properties from that class are changed. Changes to the foundational classes are especially disruptive. To reduce this disruption and to allow for more decoupled development, the IAI in its most recent release of the IFCs, version 2x in October 2000, agreed to freeze changes to the foundational classes for a period of time. By contrast, RosettaNet PIPs can be developed, tested, and implemented completely independent of each other.

Instead of involving software vendors in implementing, testing, and improving proposed standards before they are released, the MSG follows the structuralist approach of releasing standards to then be implemented. The elaborate process by which the IFCs are developed is described in (IAI, 1999). A new IFC release is developed in three stages: *Propose Project, Specify Requirements*, and *Integrate Model.* In the Propose Project stage, committees drawn from the IAI chapters set the scope for a new release, considering available resources. In the Specify Requirements stage, the MSG actually develops the new IFCs. In the Integrate Model stage, separate IFC projects are synthesized into a single model, and documentation is prepared. This new model is then open for review, and after any final changes, the new IFCs are released. Once released, the IFCs may then be implemented. As we saw in Section 2.4, implementation and testing of a standard before it is released has many benefits. For standards focused upon solving specific problems or enabling specific interactions, implementation and testing during the development process provides feedback on how well a proposed standard would meet its objectives. The process also sheds light on implementation difficulties and identifies problems in a proposed standard while they can still be changed. Without testing, problems that are not found before a standard is released will remain to be found afterwards. One reason that the structuralist approach commonly lacks implementation within the development process is that it is very difficult to implement a large, complex standard, such as the IFCs.

Instead of a process that produces useful standards as quickly as possible, such as the three-month PIP Assembly Line, the IAI process has taken over seven years to produce just a small part of the comprehensive model it hopes to produce, and implementations of small subsets of the model take years more. In this regard, the experience of the IAI has been very similar to that of the OSI described in Section 2.3.1. The chronology is as follows. In January 1997, IFC version 1.0 was released, covering a very small model. In December 1997, IFC version 1.5 was released with a reorganization of the model architecture and foundational classes. Trials with IFC version 1.5 uncovered implementation problems, necessitating IFC version 1.5.1, released in August 1998, again with changes to the foundational classes. This version was the first to have commercial implementations. As with the OSI, vendors chose to implement only small subsets of the standard. In mid-2000, three CAD vendors became the first to have certified implementations of subsets of version 1.5.1. In April 1999, IFC version 2.0 was released. This time, as with the OSI, vendors were reluctant to undertake the difficult and time-consuming task of implementing an unproven standard. To facilitate vendor implementation, the Building Lifecycle Interoperable Software (BLIS) Project, discussed below, was formed in mid-1999. In mid-2001, vendor participants in the BLIS Project became the first to have certified implementations of subsets of version 2.0. With both version 1.5.1 and 2.0, there was a two-year time lag between the release of the IFCs and their first certified implementations of *subsets*. In October 2000, IFC version 2x was released, and there are no certified implementations of this version as of February 2002. As we saw with the OSI, standardization and implementation using the structuralist approach requires many years. After seven years of effort, partial implementations of the IFCs are now available; it is still too soon, however, to tell whether these implementations will be successful at solving the problems they were meant to solve.

EXPRESS and XML are two different languages for representing data. The IAI began its work several years before the appearance of XML, and the IFCs are written in EXPRESS. Since its arrival, XML has quickly become the standard method for exchanging data over the Web, with extensive supporting standards and software infrastructure (see, for example, http://xml.coverpages.org). The ability to transfer data and transact over the Web is clearly a desirable goal for a data interchange standard. In the next two sections, we will examine how the IAI has tried to reconcile this goal with the fact that its model is represented in EXPRESS.

4.1.1 ifcXML

ISO/STEP has a mechanism for transferring EXPRESS files, and such files can be transferred over the Web. The problem with such files is that they can be interpreted only by software designed for the EXPRESS format. To software designed for XML, these files would be unintelligible. The ability to make use of all the software based upon XML depends upon the ability to translate between EXPRESS and XML. The problem with such translation is that the two languages are very different -- each has concepts that are not present in the other -- so simple mappings are precluded.

ISO/STEP Part 28 is a mechanism for representing EXPRESS in XML without loss of information. However, the XML representation of data by this method is very different from the way data is typically represented in XML. Part 28 files can be manipulated using XML technology, but they can still be interpreted only by software expressly designed for them. Part 28 files also suffer from being many times greater in size than would be the case for the same data as typically represented in XML.

The BLIS Project and the IAI have developed XML representations of the IFCs. The BLIS Project first developed BLIS-XML (http://www.blis-project.org/BLIS_XML/). The IAI then developed the very similar ifcXML (IAI, 2001b). ifcXML illustrates the difficulty of representing EXPRESS in XML. ifcXML has two representations. The first representation is created in the same way as Part 28, through automatic translation, and it is similarly large and different from typical XML usage. As the ifcXML documentation states, ISO/STEP modeling "leads to a rather complex data structure within the IFC model, which, if directly translated, does not reflect best usage of XML" (IAI, 2001b, p. 10). The second representation condenses the first and tries to make the result closer to typical XML usage. However, due to the difference between EXPRESS and XML, there is an inherent trade-off between accurately representing the original EXPRESS model in XML and producing a result that is manageable in size and close to typical XML usage. Neither ifcXML representation maintains all the information of the original model, so the conversion of an IFC model into ifcXML and then back into EXPRESS will more than likely result in a model different from the original.

In summary, the IAI has a fundamental problem in attempting to make use of the considerable and ever-growing infrastructure that exists for exchanging data in XML, since the IFCs are written in EXPRESS. On the one hand, to represent an EXPRESS model in XML without any loss of information, as with Part 28, requires large, cumbersome files that are not in typical XML usage -- this undermines the goal of using the XML infrastructure. On the other hand, to represent an EXPRESS model in XML with smaller files closer to typical XML usage, as with ifcXML, requires the loss of information -- this undermines the goal of creating a standard data model for the exchange of information in the AEC industry. The approach taken by numerous other standardization efforts, such as RosettaNet, has been to develop their standards directly in XML.

4.1.2 aecXML

aecXML (http://www.iai-na.org/domains/aecxml.html) is a domain of the North American Chapter of the IAI. It started at Bentley in 1999 with the objective of developing within one year XML schemas for the AEC industry. (A schema is a representation of data.) Its initial meetings drew curious industry participants, but attendance greatly diminished thereafter. For an undertaking whose scope is analogous to that of RosettaNet, aecXML has had very limited resources -- it has no paid staff or budget -- and it has made little progress. Given its constraints, aecXML early on changed its objective from developing schemas to accepting schemas developed and submitted by others. XML schemas are very easy to develop, and there has been a great proliferation of schemas of varying quality and degrees of use. The schemas that have been submitted to aecXML are listed at http://www.iai-na.org/schemas/. As of February 2002, five schemas have been submitted: LandXML and gbXML, both of which will be discussed in Section 4.3, and three very small and specialized subschemas. In addition to accepting schemas, aecXML participants envision an approval process for the schemas, though currently such a process is more hypothetical than actual.

Soon after it was formed, aecXML became a part of the IAI. This move entailed something of a clash of cultures, since the typical practice for developing XML schemas is very different from that of ISO/STEP object-oriented product modeling. The aecXML Framework (aecXML Technical Committee, 2001) shows how this clash was resolved. Blending language from XML and object-oriented modeling, the Framework envisions a Common Object Repository (COR) and a Common Object Schema (COS) with the following process. The schemas submitted to aecXML would enter the COR. "Common objects" from schemas in the COR would be "consolidated" and placed in the COS. The COS would serve a role analogous to the foundation classes of the IFCs. "Objects" from the COS would replace "objects" in schemas in the COR. If this were possible, it would, of course, be as disruptive as changes to the foundation classes of the IFCs are currently. ifcXML is seen as the preferred schema in the COR. One serious problem with this approach is that XML in not an object-oriented language that readily enables the sort of manipulations envisioned. As mentioned above, only a few schemas have been submitted to aecXML, and the rest of this process is currently more hypothetical than actual.

An actual repository for XML schemas, including a section for the Construction Industry, is maintained by OASIS (http://www.oasis-open.org), a leading industry consortium in XML, at http://www.xml.org.

4.2 Building Lifecycle Interoperable Software (BLIS) Project

In April 1999, the IAI released IFC version 2.0. As we saw above, the structuralist process that the IAI follows in developing the IFCs does not include implementation and testing by software vendors during development. After this IFC release, vendors were reluctant to undertake the difficult and time-consuming process of implementing an unproven standard. In mid-1999, the Building Lifecycle Interoperable Software (BLIS) Project (http://www.blis-project.org) was formed to facilitate vendor implementation. BLIS is an organization separate from the IAI, with its own membership. As of February 2002, it had 60 members, including 33 software organizations (http://www.blis-project.org/participants.htm). Many of the software organizations are relatively small; roughly half are from Europe. They include government research laboratories, architectural CAD vendors, and vendors of more specialized applications such as energy simulation, HVAC design, and cost estimating.

BLIS has brought some minimalist principles lacking from the IAI process to the task of selectively implementing the IFCs. It first carefully selected specific data interchanges, or "use cases," of an AEC project to focus on. It then organized its efforts into the independent implementation of subsets of the IFCs, or "views," corresponding to these use cases. BLIS use cases include:

- Architectural design -> Quantities take off / cost estimating
- Architectural design -> Thermal load calculations / HVAC system design

Today, data input to quantity take off / cost estimating software is largely a manual process. Thermal load calculations are largely made using simplifying approximations rather than by manually entering the necessary data into software that could make more accurate calculations. The goal of these two use cases is to enable the automatic transfer of needed data from CAD software via the IFCs. As we saw above, the IFCs are developed by product modelers who are not involved with implementation. With BLIS, the IFCs serve only as a starting point in the task of getting software applications on each side of a use case to actually exchange and understand data. BLIS has found that implementation requires considerable additional information and agreement beyond the IFCs, and it has assumed a standardization role by developing implementation agreements to fill the gaps. This further standardization is closely tied with implementation and testing. In May 2001, after two years of effort, BLIS began certifying applications, and by February 2002, 14 certified applications had been released.

Though BLIS has increased vendor participation and brought a much-needed implementation component to the IAI process, fundamental and inescapable problems remain. There is still a structuralist separation between the standardization of the IFCs and their implementation. The standardization of the IFCs comes first and cannot benefit from the experience of implementation and testing. As we saw above, the development of the IFCs is a very slow process, and BLIS is constrained by the IFCs that are available. For example, Timberline is a leading vendor of cost estimation software and a BLIS member. It has long been interested in automatically importing data for its software from CAD software, the first use case above. The BLIS view for this use case, drawn from IFC version 2.0, represents only a small fraction of the data that Timberline would like to import, and the development of IFCs for the remaining data could take many years. Since the IFCs are written in EXPRESS instead of XML, BLIS has the same problems that we saw above in using the growing XML-based infrastructure for exchanging data over the Web. Finally, as we saw with RosettaNet, the participation of the key players in an industry is important to the successful adoption of a new standard. In BLIS, the large CAD vendors, such as Autodesk (from which the IAI originated) and Bentley, are notably absent. The absence of such vendors is particularly problematic, since CAD software plays a central role in the data of an AEC project.

4.3 Independent Efforts

Much of the development of XML schemas has taken place outside of large, international standardization organizations such as the OSI or the IAI. The development principles of Section 2.4 also apply to smaller, independent efforts, and they can explain the varying degrees of success of these efforts. In this section, we will examine two such efforts from the AEC industry.

4.3.1 LandXML

LandXML (http://www.landxml.org) is an effort to standardize civil engineering and survey data for land development. Such development is typically designed with CAD software, and a trend in this software has been to extend the data representation from simply a set of lines to include the meaning of the lines. In land development, the lines could mean a parcel, the centerline geometry of a road, or a pipe network. The owner of a land development project, such as a state department of transportation, typically requires the submission of this data from the AEC project team for future use. The goal of LandXML is to provide a medium for the exchange of land development data among the various software applications of the AEC project team and for official submission to owners.

LandXML has put minimalist principles to good use, from its focused scope to its gathering of participants and the process it has followed. The project began in December 1999 at the CAD vendor Autodesk, and one Autodesk engineer is responsible for most of the schema development and for coordinating feedback from participants. CAD software vendors such as Autodesk have a long history of developing incompatible proprietary data formats to lock in customers to their products, but LandXML has taken a different approach. It has built an online community of over 160 people from 80 organizations -- representing owners, software vendors, and end users -- to provide feedback on the evolving standard. The organizations include state and federal government agencies, the American Association of State Highway and Transportation Officials (AASHTO), engineering firms, software vendors, and the manufacturers of surveying equipment.

LandXML started small, incorporating the existing Engineering and Survey - Exchange (EAS-E) standard, and it has grown incrementally and experimentally, with implementation and testing, and with comments from the online community. After nine preliminary versions, the first working draft of LandXML version 1.0 was released in July 2001, and it has been revised roughly every two months after that. It is too soon to tell whether LandXML will succeed as a standard, but with the support of surveying equipment manufacturers who generate the initial data, the support of AASHTO and owners who receive the final data, and with the participation of key land development software vendors, its outlook is promising. With Bentley, an archrival of Autodesk, planning to use LandXML in its products, there are the beginnings of a healthy shift from competition based upon data formats to competition based upon software functionality.

4.3.2 Green Building XML (gbXML)

Green Building XML (gbXML) (http://www.gbxml.org) is a schema developed by the small engineering consulting firm GeoPraxis (http://www.geopraxis.com) for data used in energy analysis software. Sophisticated software packages, such as DOE-2 and EnergyPlus developed by the US Department of Energy, can accurately predict the energy characteristics of a building from its design, but the difficulty of entering the necessary data into such packages has been a significant obstacle to their use. The desire to get such data automatically from CAD software led to one of the BLIS use cases that we saw above and is also the motivation for gbXML.

In contrast to LandXML, gbXML was developed by GeoPraxis without a wide representation and involvement of participants who would use the standard. GeoPraxis consulted with potential users, but instead of a process of incremental development tied with user implementation and testing, gbXML was released for comment in May 2000 and published one month later. Most notably absent from the development process were the federal energy labs and the large CAD vendors. Whether on a small scale, such as gbXML, or a large scale, such as RosettaNet, the commitment of key players and their participation in the standardization process is crucial to the successful development and ultimate adoption of a standard. Without the commitment or participation of the key players in its domain, gbXML has languished. It has gathered only a few users, including Artifice, a small CAD vendor, and Trane, the HVAC vendor.

5. Conclusions and Recommendations

5.1 Conclusions

1) Best practices exist for successfully developing data interchange standards

The development of information technology standards has a long history. From the experience of past standardization efforts, the relative success of the two broad approaches to standardization is well understood (Section 2). The *minimalist* approach aims to produce incremental standards relatively quickly, with a heavy emphasis on implementation, testing, and iterative improvement of standards before they are released. The minimalist approach has proven to be the best practice for developing information technology standards, especially in environments with rapidly changing technology and market needs (Section 2.4). Successful standardization efforts using this approach include the Internet standards, HTML, the C programming language, and the SQL database language. In contrast, the *structuralist* approach aims to produce comprehensive standards and is characterized by a relatively slow process that is not very amenable to implementation and testing of standards before they are released. Structuralist standardization efforts in information technology have been much less successful than minimalist efforts. Unsuccessful structuralist efforts include the Open Systems Interconnection (OSI), the Ada programming language, Integrated Services Digital Network (ISDN), and the Standard for the Exchange of Product model data (STEP).

The success of the minimalist approach to standardization has been across underlying technologies, including the relatively new technology XML. One of the most successful industry standardization efforts in XML, RosettaNet, is based upon this approach (Section 3).

2) The data interchange standardization needs of the AEC industry are not well served by the IAI

The IAI is a structuralist effort, and it is characterized by the problems and lack of success typical of such an effort (Section 4.1). Instead of having independent teams focused on producing independent, incremental, minimal standards to quickly enable specific data interchanges, the IAI has centralized its standardization into one small team attempting to develop a comprehensive product model of the data of the AEC industry. Where RosettaNet can produce in three months a standard with implementations to enable a specific data interchange, the IAI has taken over seven years to produce only a small part of the comprehensive model it hopes to produce, and the implementations of small subsets of the model take additional years. Instead of involving software vendors in implementing, testing, and improving its standards before they are released, the IAI releases its standards for vendors to then work out the implementation problems, and vendors have been reluctant to undertake the difficult and time-consuming task of implementing an unproven standard. The BLIS Project (Section 4.2) has tried to facilitate implementation, but it can only build upon the partial and slowly evolving IAI standards. The large CAD vendors, such as Autodesk and Bentley, are absent from BLIS, and this absence is particularly problematic, since CAD software plays a central role in the data of an AEC project. Finally, the IAI standards are not in XML but in EXPRESS. XML-based software has become the foundational infrastructure for exchanging data over the Web, but efforts to translate between EXPRESS and XML all involve undesirable trade-offs (Section 4.1.1).

In addition to taking the right approach, the success of industry standardization efforts, such as RosettaNet, requires the high-level commitment of the industry's key players, along with substantial resources. Recommendation 3) below reflects the reality that these prerequisites are absent in the AEC industry.

5.2 Recommendations

1) Data interchange standards for use over the Web should be developed in XML

The Web is clearly an extremely important medium for exchanging data, and not just from computer to human, but also from computer to computer. It is governed by a set of standards, including HTTP and HTML, developed by the World Wide Web Consortium (W3C) (http://www.w3.org). The W3C standard for exchanging data over the Web is XML, which has been widely adopted and has an extensive and rapidly growing base of supporting standards and software infrastructure (see, for example, http://xml.coverpages.org). To take advantage of this base of supporting standards and software infrastructure, data interchange standards need to be in XML.

2) Data interchange standardization efforts should make use of existing, widely adopted, cross-industry XML standards wherever possible

After an early explosive proliferation of XML schemas of questionable quality, cross-industry standards of high quality are beginning to emerge. An example is the ebXML Message Service Specification (http://www.ebxml.org). As such standards become widely adopted and proven, they are replacing a variety of earlier specifications. It is counterproductive for industry-specific standardization efforts to attempt to duplicate the work of such cross-industry efforts and rather they should take advantage of them wherever possible.

3) The data interchange standardization needs of the AEC industry would be better served by independent minimalist standardization efforts

The fragmented AEC industry lacks the high-level commitment of a critical mass of key players along with the substantial resources necessary for a large-scale standardization effort such as RosettaNet, but data interchange standardization in this industry can be advanced through independent minimalist efforts such as LandXML. Such smaller efforts can better garner the necessary commitment and resources, and when they follow the best practices outlined in Section 2.4 they have proven to be very effective in developing information technology standards. Even successful large-scale efforts like RosettaNet and the IETF are themselves organized into smaller independent minimalist efforts.

The BLIS Project has organized its work around "use cases" of important data transactions in an AEC project, and such use cases could be the basis for independent minimalist XML efforts. The launch of a successful effort requires the political acumen to gain the necessary participation of a critical mass of key players, as seen in the case of LandXML. Once the right participants are assembled and the right processes set in motion, the standard is then developed through experimentation, testing, and incremental improvement.

RosettaNet and the IETF each provide an architecture for their multiple standardization projects. An architecture describes how the various independent standards fit together. When the AEC industry reaches the point of having multiple XML standardization projects, such an architecture would be helpful. Since the AEC industry is lagging other industries in its development of XML data interchange standards, by the time it reaches this point, it should be able to draw upon the experience of analogous architectures from other industries.

6. Acknowledgments

I wish to thank those who I interviewed for this paper, who are too numerous to cite individually. I also wish to thank those who reviewed the paper. Finally, I wish to thank Boyd Paulson of the Department of Civil and Environmental Engineering at Stanford for his input throughout the project.

This research was supported by a grant from the Center for Integrated Facility Engineering at Stanford.

7. References

aecXML Technical Committee. 2001. *aecXML Framework.* Version 1.8, June 25.

Bock, Geoffrey E. 2000. *The Three Faces of RosettaNet: Choreographing E-Process Interactions for the Internet Age.* Patricia Seybold Group, November 9.

Bradner, Scott O. 1996. *The Internet Standards Process -- Revision 3.* http://www.ietf.org/rfc/rfc2026.txt

International Alliance for Interoperability. 1999. *IFC Specification Development Guide.* March 15.

International Alliance for Interoperability. 2001a. *The Federal Initiative: Commitment to Information Interoperability in the AEC + FM + BPF + O&M Industries.* Draft Discussion Paper, August 1.

International Alliance for Interoperability. 2001b. *ifcXML Language Binding of EXPRESS.* Version 1.01, September 15.

Libicki, Martin C. 1995a. *Information Technology Standards : Quest for the Common Byte.* Boston: Digital Press.

Libicki, Martin C. 1995b. Standards: The Rough Road to the Common Byte. In Kahin, Brian, and Janet Abbate, eds. *Standards Policy for Information Infrastructure.* MIT Press, pp. 35-78.

Libicki, Martin C., James Schneider, David R. Frelinger, and Anna Slomovic. 2000. *Scaffolding the New Web : Standards and Standards Policy for the Digital Economy.* Science and Technology Policy Institute, RAND.

Paulson, Boyd C., Jr. 1976. Designing to Reduce Construction Costs. *Journal of the Construction Division, ASCE.* Vol. 102, No. CO4, December, pp. 587-592.

Rishel, Wes. 2001. *RosettaNet: A Refreshing Approach to Standards.* Gartner, Research Note, TU-13-6990, June 25.