MIT-AUTOID-WH-003

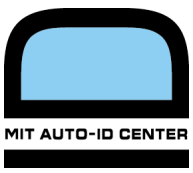
ePC: 21.0000001.06003.XXXXXXXXXX

The Physical Markup Language

David L. Brock

February 2001

Auto-ID Center
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
http://auto-id.mit.edu/



The Physical Markup Language

A Universal Language for Physical Objects

David L. Brock
Auto-ID Center
Massachusetts Institute of Technology
Cambridge, MA USA 02139

Abstract – **The Physical Markup Language (PML) is intended to be a common "language" for describing physical objects, processes and environments. Much as the Hypertext Markup Language (HTML) has standardized the way in which information is presented on the Internet, PML is designed to standardize descriptions of physical objects for use by both humans and machines. The primary objective of PML is to serve as a common base for software applications, data storage and analytic tools for industry and commerce. This paper presents some fundamental issues in the design of the language, as well as assumptions underlying its development and implementation.**

I. INTRODUCTION

The Physical Markup Language (PML) is intended to be a general, standard means for describing the physical world. When we consider that human language performs a similar function, it is clear we must carefully consider the goals and objectives of PML for any hope of successful adoption.

The objective of PML is a simple, general language for describing physical objects for use in remote monitoring and control of the physical environment. Applications include inventory management, automatic transaction, supply chain tracking, machine control and inter-object communication.

Given the objectives of PML, the language should encourage the rapid development of software tools and applications. Therefore, PML should be as simple as possible (at least in its initial implementation), yet as comprehensive as needed to provide general utility.

As opposed to the many standards and languages, which have been successfully developed in specific application domains, the intention of PML is to provide broad definitions, describing those characteristics common to all physical objects. The assumption is that applications built on this language can be applied across a broad range of industries and businesses.

In the following sections, we will present fundamental assumptions in the development of the Physical Markup Language and discuss design approaches intended to achieve the objectives for the language.

II. BACKGROUND

The intelligent infrastructure, which we envision, automatically and seamlessly links physical objects to each other, people and information through the global Internet [1]. This intelligent infrastructure has four major components: electronic tags, Electronic Product Code (ePC), Physical Markup Language (PML) and Object Naming Service (ONS).

Electronic tags refer to a family of technologies that transfer data wirelessly between tagged objects and electronic readers. Radio Frequency Identification (RFID) tags, often used in "smart cards," have small radio antennas, which transmit data over a short range [2]. The Motorola BiStatix™ tags, an Electromagnetic Identification (EMID) technology, uses capacitive coupling to transmit information [3]. Electronic tags, when coupled to a reader network, allow continuous tracking and identification of physical resources. In order to access and identify tagged objects, a unique naming system was developed.

The Electronic Product Code (ePC) was conceived as a means to identify physical objects [4]. The ePC code was created to enumerate all objects and to accommodate current and future naming methods. The ePC code was intended to be universally and globally accepted as a means to link physical objects to the computer network, and to serve as an efficient information reference.

The Object Naming Service (ONS) is the "glue," which links the Electronic Product Code (ePC) with its associated data file [5]. More specifically, the ONS is an automated networking service, which, when given an ePC number, returns a host addresses on which the corresponding data file is located. The ONS, currently under development, is based on the standard Domain Naming Service (DNS). When complete, the ONS will be efficient and scaleable, designed to handle the trillions of transactions that are expected.

Finally, the Physical Markup Language (PML) is intended to be the standard in which networked information about physical objects is written. In one sense, all the complexity of describing and classifying objects has moved away from the object label and into the PML file. The formation of this language — together

with the associated software tools and applications — is one of the most difficult aspects of this *"Internet of Things."*

III. Approach

The effective design of the Physical Markup Language must balance a myriad of competing design issues and constraints. Since we have essentially eliminated most of the information and structure on the tagged object, all the complexity of description has moved to the networked database. Issues such as syntax, data types, complexity, extensibility, security, application domains, units of measure and more, must be weighted to effective achieve the objectives set out for the language.

In the following sections, we will consider a variety of design issues, key assumptions and other considerations in the formation of the PML. This in not an exhaustive list, but a starting point in the language design.

We must remember numerous languages and standards have developed in the past, yet few see wide spread adoption. We wish to avoid the pitfalls of the past, and develop a standard, which is simple, convenient and effective.

A. Generality

The objective the Physical Markup Language is to be a universal standard for describing physical objects, processes and environments. Clearly given the broad scope of this objective, the language cannot be overly detailed or specific. In the classic choice between depth and breath, the proposed PML will lean toward a more general standard, rather than industry specific implementations.

There are number of reasons for this decision. First, a broad language will address the largest number of industries. Second, software developed for the language will have the greatest potential market. The quality and capability of the code will likely be superior to any specific implementation. This is analogous to Web browsers, such as AOL's Netscape™ or Microsoft's Internet Explorer™, both of which are generally superior to similar applications targeted for specific industries. The more generic software also tends to be more robust and less expensive than focused applications.

Third, physical objects and systems do indeed have common, underlying characteristics. Since most physical objects of interest to industry and commerce are those designed and built by humans, they tend to have shared features, such as shape, symmetry, materials and function, as well as business processes, ownership and transaction. Furthermore, many industries, such as healthcare, manufacturing, defense, logistics, transportation, disposal and many others, describe similar characteristics in different ways. By offering a unifying language, these characteristics can be shared and translated across industry groups, multiplying the amount of available information. Automated, industry specific translators may be written allowing the shared information to be presented in familiar ways.

Finally, a broad descriptive language will encourage a greater degree of industry cooperation and facilitate information sharing for mutual benefit. Often data, such as between a retailer and supplier, is not available simply because of lack of standards.

B. Simplicity

Many standards are not adopted because of their inherent complexity and steep learning curves involved in acquisition and implementation. Although the Standard General Markup Language (SGML) has existed for many years, it has not seen wide spread adoption in part because of its size and complexity [6, 7].

Its derivative, the Hypertext Markup Language (HTML), has, of course, seen phenomenal growth, in part because of its simplicity and because of the tools and viewers available for the standard [8]. The Extensible Markup Language (XML), also based on the Standard General Markup Language, has seen increasing growth as a tool for tagging data content [9]. The XML is a simple subset of the larger SGML and is readily accessible to the casual programmer.

Thus complex standards and languages – even though powerful and effective – have slow learning curves and limited audiences than smaller, simple languages. Therefore, even though the initial PML may be limited in scope, we propose a relatively simple language easily understood and adopted by a larger population.

C. Adoption pathway

Rather than a monolithic, immutable standard, we will assume the Physical Markup Language will proceed through a number of iterations. In fact, rather than a deficiency, this process can be advantageous. While a simple standard is being learned and adopted, modifications and extensions can be developed. In this way familiarity with the language can proceed along with its capability. In fact, this process may be necessary, since a complex language would not be learned and a simple language would not be sufficient.

Although the HyperText Markup Language (HTML) was a simple language and easily understood, it was, in its initial version, quite limited in scope and in power. Multiple versions and extensions followed once the significance and utility of the language were understood. Extensions, such as Cascading Style Sheets (CSS), Dynamic HTML, Flash Media and so on, were added to the basic capability.

In the same way, we intend the initial PML specification to be limited in depth and power. By design, we will incrementally introduce extensions to increase its scope and functionality.

D. Comprehensive data types

We may consider the Physical Markup Language to have different *'types'* of data – static, temporal, dynamic and algorithmic. These types will not be defined explicitly in the specification, but are useful distinctions when discussing the language.

Static data is information, which essentially remains constant through the life of the object, such as material composition, geometry and physical properties.

Temporal data is that information which changes discreetly and intermittently throughout an object's life. These may include configuration or location. For example, the location of an object on a shelf or whether a part is attached to an assembly, are examples of this type of data. These data must be associated with a time and duration to record the temporal configuration of the environment.

Dynamic data is information that varies continuously. The temperature of a shipment of fruit or the EKG from a heart monitor are examples of dynamic data. Unlike most database systems these data must be cached and transmitted intermittently to limit the network bandwidth and to provide only the most relevant and necessary information.

Finally, algorithmic data includes simulation models, system processes and software associate with a physical object. Not all physical properties can be described by a simple number. For example, the expiration data on a perishable item may be a complex calculation involving temperature history, humidity and ambient light. Cooking instructions could be another example. Heating profiles depend on personal preference, food type and quantity, atmospheric pressure, ambient temperature and oven type.

These designations – static, temporal, dynamic and algorithmic – are simple different views of the same data. A static description such as the shape of a glass would be temporal if it hit the floor. The variation of viewpoint just depends of time scale and complexity of description. Therefore, we will allow time variation on *all* data descriptions.

E. Abstract nomenclature

Clearly if we hope for a broad application of this language, we cannot expect familiar names for all physical properties. For example, "harvest time" for produce or "assembly time" for an automobile, may be replaced by a more generic "configuration" plus "timestamp." Generally, we will use abstract names to describe a wider range of physical systems and processes, rather than industry specific descriptions.

Why use abstract notation? The answer is – when we consider the primary objective of the language – to provide a convenient, high-level description for software and application development. More generic terms allow more powerful, general-purpose software to analysis similar configurations independent of industry specific nomenclature.

F. Robust operation

Unlike most Web pages, PML files will be much more dynamic and have a greater degree of connection to other network files and data streams. Object position, physical state and material descriptions will likely be in multiple data files scattered over the network. General physical properties, such as material and chemical information, will likely be stored in common repositories. Material Safety Data Sheets (MSDS) are good examples of this type of data.

The PML language, together with associated tools and applications, will have to operate robustly with incomplete and intermittent information. Its operation may be similar to streaming image systems do today.

G. Facilitate data archives

Although Web pages change frequently, PML data files will change even more rapidly. History files and efficient archiving will therefore be critical important. The temperature history of a perishable item, administration of drug or stress on structure must be carefully recorded and maintained.

The PML data format will have to provide simple and convenient methods for associating time with data and for denoting periodic and continuous data.

H. Standard units of measure

For much of recorded history, physical states of matter have been compared to known references. From cubits to nanometers from stones to dekagrams, multiples of common standards provide the means of communicating physical properties. A difficulty arises when different countries, groups, organizations and people use different and competing standards.

Our desire for the Physical Markup Language to be a global standard must be weighed against the utility and convenience for the user. In particular we must decide on a method for recording data and units, and converting it from one system to another as necessary.

Fundamental physical properties of matter – length, mass, time, force, velocity, density, magnetic field, luminosity and temperature – must be described precisely to be communicated effectively. Many physical properties are not independent. Speed, for example, is the ratio of length to time. Certain quantities must be selected as fundamental, while others derived.

Fortunately, these issues have been resolved by standards bodies, such as the International Bureau of Weights and Measures (*Le Système International d'Unités – SI*) in conjunction with others such as the Nation Institute of Standards and Technology in the United States. The seven quantities selected as the basis of the International System of Units, abbreviated SI, were selected, and are shown in Table 1. Furthermore, all other units can be described by multiples or ratios of these units. Pressure, for example, is given by $m^{-1} \cdot kg \cdot s^{-2}$. Finally, names for common combinations, such as Pascals for the pressure given above, are provided under the SI system.

Base	**Name**	**Symbol**
Length	meter	m
Mass	kilogram	kg
Time	second	s
Current	Ampere	A
Temperature	Kelvin	K
Amount	Mole	mol
Intensity	Candela	cd

Table 1. The seven SI base units assure mutual independent, unambiguous measurement [10].

Although the above discussion is fine for scientific precision of weights and measures, we have the practical problem describing physical properties in the multiple common systems people use today. Considering the options, we may allow PML to use any standard –International System, British or other. We may also allow any designation of unit, such as "kilograms," "kgs" or "Kg." This makes the creation of PML files easy, since any standard of measure written in any language and with any abbreviation may be used. The software tools that must process these data files, however, must be complex, since they must recognize and translate any arbitrary designations.

On the other hand, if we rigidly dictate a particular standard in a single language, we have difficulty in readability and usage. Each PML application must translate units into their common, local standards. In the whole, translating from a known standard to another is easier then converting from an unknown, arbitrary language.

From this reason, it seems likely PML will adopt a single system for weights and measures, with particular designations, and rely on the software tools to provide common translations. Furthermore, common translation software can be accessed and shared from the network. This creates smaller, more easily understood data files, which are precise and accessible. Further, we will rely on the years of effort by the many standards bodies to prescribe these systems.

I. Fundamental and derived data

Many schemes used to store information include redundant and derived data. As much as possible, the PML language should not provide any data that can be calculated or inferred from other data. Unit conversion for example may be computed by a client application, remote server, or perhaps by a dedicated conversion/computation system.

J. Standard Syntax

Rather than reinvent a new syntax for the Physical Markup Language, we propose to use the extensible Markup Language (XML). Although different syntactic representations could be used, XML has been well defined and in general use as a simple method for embedded meta-data in flexible database structures.

Furthermore, the extensions, such as the XML Query specification, provide a uniform and simple method for accessing data through Simple Query Language (SQL) notation [11]. In addition, general utilities, tools and validation software exist to parse, modify and access XML files.

The Physical Markup Language (PML) will therefore be – at least initially – an XML scheme, described in any of the common schema languages, such as the Document

Type Declaration (DTD), Resource Description Framework (RDF) and others [9, 12].

K. Global language

As with current trends in standards development and network languages, we will attempt to craft PML as a global standard and avoid national terms and descriptions. We will rely on existing standards bodies, such as the Uniform Code Council (UCC), the European Article Number (EAN) Association, the American National Standards Institute (ANSI) and the International Standards Organization (ISO), as well as commercial consortium and industry groups, to aid in the definition of the language.

L. Facilitate application development

One of the primary purposes of the Physical Markup Language is to facilitate the development of software applications. Therefore, we must design PML with consideration for the needs and requirements of application programmer.

Almost all the issues discussed so far relate to this objective. Widely adopted, simple languages encourage application development and ease the programming task. Extensions and enhancements to an established language will be paralleled by modifications to existing code. Simple, unambiguous nomenclature reduces the complexity of the PML parser and uniform units for weights and measures ease the burden of software translators. Finally, common, globally accepted syntax, such as XML, together with software libraries, such as the JAVA DOM and SAX packages, provide useful tools for the software developer [13].

The design of the Physical Markup Language will accommodate the application developer and provide the systems and tools to facilitate their efforts. As future versions of the PML become available, we will streamline the semantics to speed software upgrades and new applications.

IV. DESIGN

A. Overview

In the following subsections, we will consider characteristics common to physical objects, for the purposes of forming a basis for the Physical Markup Language. The physical features considered are by no means exhaustive, but serves as simply starting point for the general design of the language.

B. Hierarchy

Physical objects often display some degree of regularity and organization. A fundamental type of organizational structure is *hierarchy* – the composition of parts and subparts. We think of machines having assemblies, systems, subsystems and parts, as illustrated schematically in Figure 1. These hierarchical descriptions apply not only to assemblies, but also to aggregates and collections. A tea set, for example, may be comprised of cups, saucers and spoons, yet have no physical connection.

Even natural objects have hierarchical structure. The tree being the classic example – having a root, trunk, branches and leaves. This characteristic of natural and man-made objects to exhibit a hierarchical structure should be included in any language of the physical world

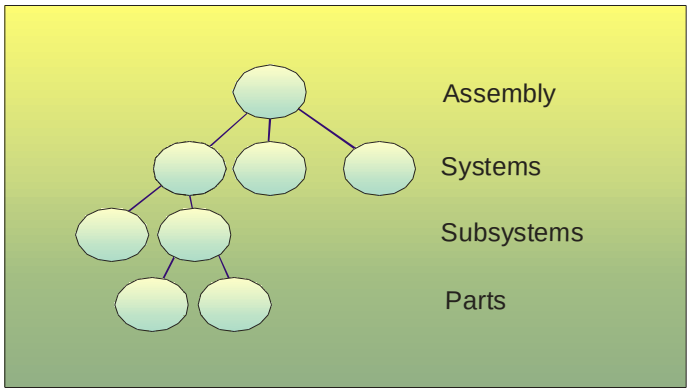


Figure 1. Physical objects – both natural and man-made – often display a hierarchical structure.

Beyond simple containment, the relationship between a parent and child object is often critical in describing the physical system. These relationships exist not only up and down the hierarchical tree, but also across sibling elements. A mechanical joint is a good example. Kinematic pairs, including revolute, prismatic and ball-in-socket joints, are often used to describe the coupling between elements in a mechanical system. A bolt, illustrated abstractly in Figure 2, shows how this may be done.

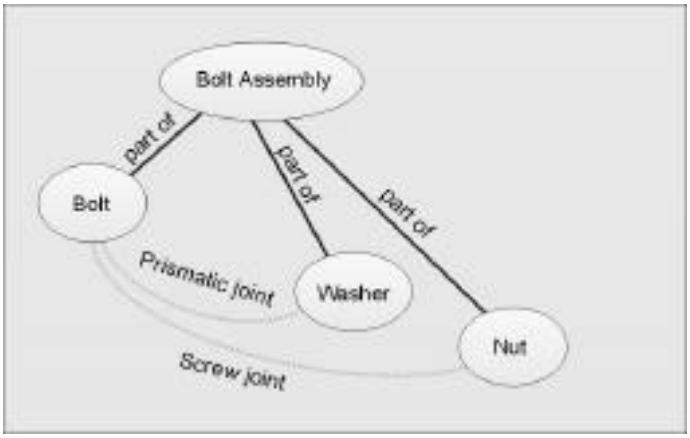


Figure 2. A bolt assembly, which consists of a bolt, washer and nut, may be thought of as a hierarchy with well defined relations between elements.

Elements in the supply chain can also be thought of as an assembly. The transport vehicle, pallet, container and item form elements in a linked hierarchy, as shown in Figure 3. In this case, we explicitly define the pallet and pallet assembly as separate elements. A pallet, for example, would be considered a discrete item for a pallet logistics company, but a shipping unit for a transport company. It is important when developing the Physical Markup Language to provide unambiguous descriptions for all possible users.

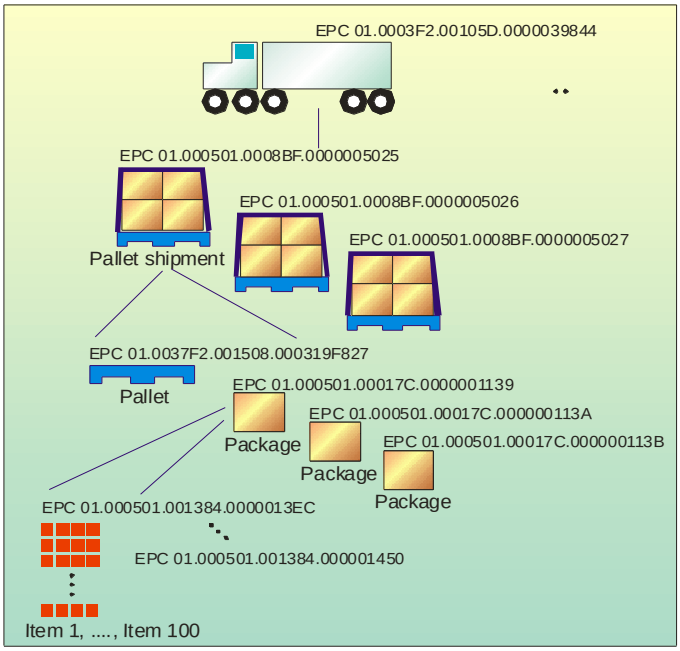


Figure 3. Elements in a shipment form a hierarchy composed of a transport vehicle, pallets, containers and items

It is important to note, these hierarchies change over time. Links are continually forming and breaking. Consider a shipment loaded from a truck into a warehouse. The virtual link describing this assembly disassociate from the truck and reform with the warehouse. The transition in structure may trigger events, such as a change of ownership or responsibility, or perhaps a financial transaction such as a payment or refund.

The transport vehicle, pallets, packages and items, in the previous example, form a hierarchy of four levels. Suppose, however, we had considered the entire shipping fleet, or, conversely, included the contents of every shipped item.

Clearly different levels of detail are needed for different users and applications. The level of detail *depends* on the observer of the data. The concept of viewer dependent description will underlie the presentation of PML information.

C. Classification and Categorization

Perhaps one of the greatest challenges in describing physical objects is classification and categorization. Within the Physical Markup Language, we must include data structures and formats that provide efficient methods for classifying objects. There are, of course, many previously developed standards and languages that provide this capability.

In object oriented programming languages, such as C++, JAVA and ADA, as well as modeling tools, such as the Uniform Modeling Language (UML), there is an emphasis on building efficient class hierarchies [17].

A classic example is “the apples and oranges.” In this example, an ‘apple’ and ‘orange’ are a type-of ‘fruit’, and a ‘fruit’ is a type-of ‘food,’ as illustrated in Figure 4.

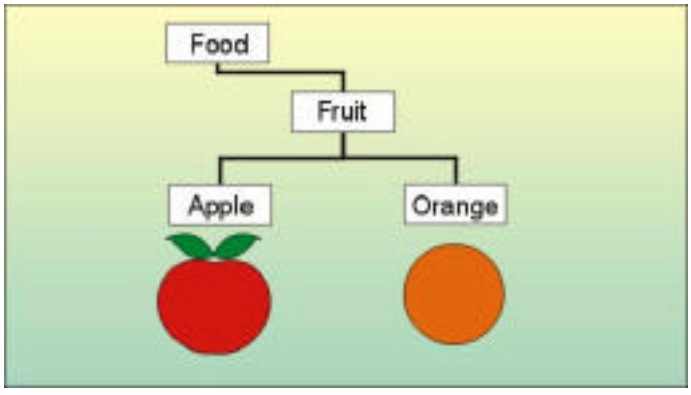


Figure 4. Classification and generalization are important functions for any language describing the physical world.

There are, however, many different ways to classify objects – and this is the real problem. We might say an 'apple' and a 'stop sign' are a type-of 'round red shape', which are a type-of 'red object'. This would be critical, for example, if you suffered from red-green color deficiency. This is not a singular example. There are, perhaps, as many ways to classify the physical world as there are people to observe it.

The important point here is that classification *depends* on a particular viewpoint. The Physical Markup Language will have to accommodate multiple classification schemes for identical physical attributes.

This classification list will help the application software organize, filter and present particular characteristics of the physical world. Different views of the same data can be presented to different individuals at different times. A distributor may view the data in terms of shipment size, a retailer in terms of product movement, a consumer by price and a recycler by toxicity.

In addition to classification within a particular PML file, object descriptions may subscribe to shared categorization schemes. In other words, common themes, such as material type, product class, storage system and recycling method, may be shared globally by the object description files.

D. Component description

At some point the description of the physical world must include the idea of a "part." In other words, an irreducible element composed of essentially homogeneous material. We may consider parts to be the "nuts-and-bolts" of an system, which may be literally nuts and bolts, the liquid in a container or the gas in a cylinder.

Descriptions of solid objects are well represented in computer languages, for example the Virtual Reality Modeling Language (VRML), the ParaSolids™ modeling core and many others [14-16]. Generally, these include Boolean solid geometry, polyhedral models and smooth surfaces.

Perhaps less well represented in current languages are descriptive tools for flexible planar objects, such as paper, film and clothing. Although the exact geometry of the material is often unimportant, the planar pattern and overall configuration are useful to describe. The planar shape may be described by two-dimensional geometry and thickness; however, the overall configuration of the sheet is more difficult to describe. Folding patterns, wrinkle and knotting, for example, may be useful for a laundry. The language for these objects must evolve needs of the application.

Flexible linear objects are common, yet are not well represented in formal languages. These include thread, cord, rope, wire, conduit and cable. As with planar objects, the cross-sectional geometry of linear objects is relatively easy to describe. The linear geometry, however, is more difficult to represent. Exact geometry may be needed for some objects, such as piping, but unnecessary for others, such as cables and rope.

For geometric representation, PML will use establish, well-described languages, and extend these as needed for particular applications.

E. Ascribed Information

In addition to intrinsic information about an object, the Physical Markup Language must accommodate data ascribed to an object. This type of information includes names, titles, ownership, responsibility, cost and value. To a large degree, PML will use the extensive work already developed in this domain, particularly from electronic commerce initiatives, such as ebXML and UDDI [17, 18]. As much as possible, we will cooperate with these organizations to provide consistent and seamless integration with existing standards and languages.

F. Process and models

The physical world is characterized not only by the static arrangement of objects, but also by changes in these objects over time. The concept of process, that is the continuous change in the environment over time, is central to the concept of work. In addition, the anticipation changes or the projection of possible outcomes is planning. Although not in the initial implementations, the Physical Markup Language must eventually include descriptions process plans, schedules and timelines.

V. CONCLUSION

This paper proposes the concept of a united language for describing physical objects. We have presented some general guidelines, key assumptions and fundamental components of the language. From the initial specification through subsequent versions, we must evaluate breath and complexity relative to user benefit and industrial application. Clearly the successful standard is one that is used widely and applied effectively.

VI. REFERENCES

1. "The Networked Physical World - *Proposal for Engineering the Next Generation of Computing, Commerce and Automatic-Identification*," *http://auto-id.mit.edu/pdf/MIT-.AUTOID-WH-001.pdf*

2. Radio Frequency Identification (RFID) summary from the AIM Global Network (*http://www.aimglobal.org*). *http://www.aimglobal.org/technologies/rfid/*

3. Motorola BiStatix Technology *http://www.motorola.com/GSS/SSTG/smartcard/3_0_bst_home.htm* *http://www.motorola.com/GSS/SSTG/smartcard/white_papers/BiStatix_Whitepaper.pdf*

4. Brock, D. L, "The Electronic Product Code – *A Naming Scheme for Physical Objects*," Auto-ID White Paper, WH-002 *http://auto-id.mit.edu/pdf/MIT-AUTOID-WH-002.pdf*.

5. The Object Naming Service (ONS) summary from the MIT Auto-ID Laboratory *http://auto-id.mit.edu/research/naming.html*.

6. SGML Overview and references. *http://www.oasis-open.org/cover*.

1. St. Laurent, Simon, "XML™: A Primer, 2nd Edition," MIS Press, New York, 1999.

8. The HyperText Markup Language (HTML) Specification World Wide Web Consortium *http://www.w3.org/Markup*.

9. The Extensible Markup Language (XML) Specification World Wide Web Consortium *http://www.w3.org/XML*.

10. The International System of Units (SI) from the National Institute of Standards and Technology (NIST) *http://www.nist.gov* *http://www.nist.gov/cuu/Units/units.html*.

11. The Extensible Markup Language (XML) Query Specification World Wide Web Consortium *http://www.w3.org/XML/Query/*

12. The Resource Description Framework (RDF) Specification World Wide Web Consortium *http://www.w3.org/RDF*.

13. The Document Object Model (DOM) and the Simple API for XML (SAX) World Wide Web Consortium *http://www.w3.org/XML/DOM*. *http://www.w3.org/XML/SAX/*

14. The Virtual Reality Modeling Language (VRML) *http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm*

    from the Web3D Consortium *http://www.web3d.org/*.

15. UGS Corporation, Parasolid™ modeler *http://www.ugs.com/products/parasolid/*.

16. Foley, van Dam, Feiner, and Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, Reading, MA, 1990.

17. Unified Modeling Language Resource Center *http://www.rational.com/uml/*.

18. ebXML electronic commerce language *http://www.ebxml.org*.

19. Universal Description, Discovery, and Integration (UDDI) *http://www.uddi.org*.