Benefits of Avoiding Runtime/Build-Time Distinctions for Metadata Vocabularies

Kenneth **Laskey**

Abstract

The paradigm of runtime vs. build-time is predicated on the assumption that we have two groups of people doing very different work and this work requires different metadata systems and tools. While in general we can say the skill sets of different user groups are different, is the build vs. run distinction necessary and useful in the context of Web services and service-oriented architecture? This paper will look at the activities of representative groups in building and using metadata vocabularies and argue that a rigid distinction may be neither real nor helpful.

Table of Contents

1. Introduction

The concepts of build time and run time have morphed from a fairly clear separation during traditional software development and operation cycles to fuzzier terms in the context of Web services and service-oriented architecture (SOA). Software developers traditionally wrote code during build time, possibly making use of known software libraries associated with their development language and environment. The software was tested and deployed. The end user bought and used this software and did not search for pieces of it from a registry. The end user may have searched for domain-specific information, but that was likely done through their organization's content management system.

The World Wide Web began to change that. Everyone with a question goes to their favorite search engine, poses the question, and then wades through the pages of responses hoping to find a relevant answer. More often than not, they do. Search engines preferred by developers are generally no different than those preferred by accountants. In fact, the common capabilities and common interfaces of the World Wide Web has provided a larger audience and incentive for technical innovation. Is there a message here for Web services and SOA?

As we look to expanding our search capabilities through the use of quality metadata, the build time vs. runtime perspective is often invoked in describing metadata systems and processes, but metadata is not software and may benefit from an approach more tailored to its requirements. To better understand metadata creation and use, an analysis was conducted on representative use cases and the results conclude

- a modular design of metadata is a natural approach to schema development and will facilitate interoperability;
- user roles related to metadata creation and use comprise common activities and can naturally be supported by common mechanisms and tools;
- rather than build vs. run, a more useful partitioning for metadata creation and use is query and populate;
- the conclusions are relevant beyond an isolated view of the metadata life cycle and are applicable to search activities for content, services, and any other resource.

2. Motivation

Build time vs. run time or common vs. differentiated search facilities were not the original questions when work began on a metadata concept of operations (CONOPS) to support the Global Information Grid (GIG) Core Enterprise Services (CES). The GIG CES Strategy [GIG-CES] calls for "robust [GIG] enterprise services (GES) [to] provide visibility and access to data, enabling the end user to execute an intelligent pull of mission-tailored information from anywhere within the network environment." Moreover, "the CESs provide the basic ability to search the DoD enterprise for desired information and services, and then establish a connection to the desired service/data."

This vision describes an environment where the interaction between the providers and consumers of resources must be flexible and readily configurable across entities (consumers, providers, and resources) that had no previous knowledge that the others existed. The DoD Net-Centric Data Strategy [DataStrategy] lays out a path for accomplishing this through the use of metadata. A primary user (both provider and consumer) of this metadata is Net-Centric Enterprise Services (NCES), a program created to provide the services and capabilities that are key to enabling the ubiquitous access to reliable decision-quality information that is envisioned by GES. The initial scope and requirements for GES were defined through the NCES Analysis of Alternatives (AoA) [AoA]. In support of the AoA activity, an initial set of core enterprise services was identified and further defined by inter-Service, inter-Agency teams, and then the AoA effort defined a set of use cases corresponding to these core services, with the use cases representing typical scenarios that an early NCES deployment might support.

In addition to the AoA effort to define services, it was widely recognized that there had been no detailed presentation of what metadata must be created and managed, how it would be managed, and by whom. Thus, a subsequent effort

was chartered to fill that gap by providing a concept of operations for metadata. In order to provide continuity with the work of the AoA, the metadata CONOPS effort [OCD] analyzed a subset of the AoA use cases to determine

- what types of metadata were implied by the use cases;
- what functions were implied if such metadata was to be created, maintained, and used;
- what was implied about a metadata infrastructure needed to support this metadata and the related functions.

The metadata analysis observed several patterns regarding the form metadata should take to effectively support goals such as those for GES, and assembled several generic scenarios identifying user roles and activities. The remainder of this paper will discuss those roles and activities and will present conclusions derived from further independent analysis.

3. Initial Findings – The Case for Modular Metadata

The methodology for analyzing the AoA use cases was to consider each step of each selected use case, identify the likely metadata and metadata support processes needed, and then to look for commonality across the identified metadata elements. This proved essential to the final findings because it highlighted common activities and the degree to which these common activities can be supported by reusable vocabulary structures.

The analysis [metadata][oasis_symp] found numerous metadata structures that provided similar if not identical functionality in all uses. Such commonality is not unexpected because the underlying assumption for a schema registry is that interoperability will be facilitated by reuse of common schema elements. However, the analysis highlighted the granularity at which reuse is most likely to occur and the extent to which commonalities can be leveraged to further the goals and ultimate value of metadata creation, maintenance, and use.

For example, date was a common item. While how the date applies to a context will certainly change (e.g., date can be the first date I can register for the conference or the date early registration expires), the information contained in the date string fits a finite number of patterns with well-known mappings between the patterns. (See, for example, the ISO date standard [DateTime].) The same can be said for a metadata item for a person's name. If one focuses (or better, finds someone else whose interest is to provide the focus) on creating a structure that captures the nuances of the name concept (see for example the HR-XML [HR-XML] work on a standard format for person names), there is an immediate degree of interoperability between any users of the common representation or representations related through readily-accessible mappings. Indeed, more complex structures can be easily assembled by combining the basic ones. While this is certainly consistent with the usual plea for reuse, it is important to note that it is much easier to reuse a representation of a concise concept than it is to isolate portions of a schema that is several (or dozens of) pages long. Identifying a known reference-able pattern also avoids the question of whether copying and pasting part of someone else's schema is really reuse if there is no lasting link between the original and the copy.

The modular approach of defining complex metadata structures in terms of more elementary metadata building blocks is a fundamental change from the typical pattern of developing distinct, complex metadata structures for every different use. This modular approach to metadata is consistent with the current paradigm for building software, but up until now metadata has more often been compartmentalized, hindering its reuse in the same way that compartmentalization had hindered reuse in early software development. In addition, the information in metadata instances created against these differing schemas has also been compartmentalized because reuse required the information to be reentered into each structure. If common modules can be reused, it is likely common data can also. Thus, a more effective approach to metadata would be to define generic modules and support the user in assembling these modules for custom purposes.[1]

[1]The discussion of modular metadata is easily a paper in itself. For example, note that metadata structures have additional flexibility over software modules in that mappings among similar metadata structures can provide acceptable interoperability even if users find module variations to be useful. The n^2 problem can be lessened if mappings are transitive, i.e. a mapping of A to B can be combined with a mapping of B to C to give mapping information about A to C. Schema to schema mappings can also capture details that mapping each to a common schema may lose. In general, if concepts have a complexity for which the n^2 problem cannot be avoided, then the common schema approach will be unsatisfactory.

4. The Metadata Users

While the initial findings indicated the benefits for modularization of metadata, the question remained on how this affected the tasks performed by typical users. The next step in the analysis was to identify user roles and characteristics of those roles. The roles fell into four groups:

- schema and tool developers,
- metadata instance producers,
- metadata instance consumers, and
- metadata managers.

While the roles are often considered discrete, the individuals in each of these roles also assume the other roles in the process of completing their primary tasks. The primary description of each role follows below, but each description also highlights where aspects of other roles apply.

Schema and tool developers. Developers are generally human agents who create the classifications, schemas, and ontologies that provide a representation of the metadata [2]. Discovery requires metadata instances to describe resources in terms of properties that enable a user to discriminate between one resource and another. The resource is the item of interest for the current task, whether that item be a car, a document, or a schema. Before metadata instances can be created, schemas must be defined that express the structure of the metadata [3] which will convey each resource's metadata values [4]. The developers are expected to be knowledgeable in the construction of schemas using XML syntax, have expertise in use of schema development tools, and be familiar with the cataloguing of schemas in schema registries. Developers must know how to (1) search for existence of schemas from which new schemas can be constructed and (2) register the new schemas. **In the former, the developers assume the role of metadata consumers and in the latter, the role of metadata producers.** The developers define the structures that provide the metadata instance syntax and may develop metadata generation tools to facilitate creating instances consistent with the metadata structure and constraints. The developers will also likely have some familiarity with a domain in which the schema is to be used in order to effectively understand and incorporate the needs of domain experts.

Metadata instance producers. These are the agents that create and maintain metadata instances. They have expertise in use of metadata creation tools and understand the basics of schema structure but are not necessarily expert at creating new schemas. In addition, they are familiar with cataloguing of schemas in schema registries in order to (1) find and access schemas to be used as the template for metadata instances and (2) register new instances. To create quality metadata, those filling this role must be familiar with data sources from which metadata instance values will be extracted or copied. **Note, a schema developer will be the metadata producer for new or modified schemas; a metadata producer is a metadata consumer when searching for the metadata template.** The metadata production process can be manual (requiring humans) or may be automated (with some combination of human and software interaction) using metadata generation tools. Using such tools may require developing and maintaining mappings between schemas and metadata value sources, where the mappings are also resources which should be described by metadata.

Metadata instance consumers. Consumers are those who access and process metadata in order to achieve domain-specific goals for which the use of metadata is a facilitating, as opposed to the primary, function of the consumer. These consumers may know little about schema development or schema cataloguing, but they are familiar with the cataloguing of metadata instances to the extent necessary to (1) discover a resource that satisfies their search criteria,

[2] In the following, the term "schema" can equally refer to any useful representations, whether schemas, ontologies, or others.

[3] This implies that to find schemas there is a schema that defines the properties for describing schemas. While this logic appears circular, it is consistent with descriptions in the XML Schema [XMLSchema] specification. The power of this construction is that the metadata describing schemas is no different from the metadata describing any other class of entities, and thus the metadata can be created, organized, and searched by common mechanisms.

[4] Schemas may alternately be created as data structures to hold information about instances of a class of entities rather than explicitly serve as a structure for metadata. The line (if there is one) between data and metadata is the topic of endless discussion, but the activities described here for a schema developer apply equally to either intent.

(2) retrieve related information about the resource (such as critical constraints and pedigree), and (3) invoke the mechanism for accessing the discovered resource. Consumers must identify the class of resource for which they have an interest because a schema that defines the metadata for that class is also the source of the available search criteria, i.e. one cannot search over arbitrary criteria for which no descriptive values have been provided. **Note, both schema developers and metadata producers will be consumers for discovering and accessing catalogued schemas, but in this role, they do not make use of schema development or metadata production knowledge – they are simply users looking for "something".**

Metadata managers. Numerous manager activities are required to exercise responsibility for the organization, storage, and appropriate access of products created by and used by the developers, producers, and consumers. These include configuration management of schemas, instances, tools and the mappings between schemas and between schemas and data value resources. It also includes maintaining the metadata storage and the infrastructure needed for metadata use. A domain manager must understand schemas from the context of their use and not necessarily the details of underlying methodology or specifics of development. They should be familiar with the use of directives and business rules in system governance. It is noted that managers will have many opportunities to be metadata producers and consumers, but a detailed discussion of manager activities is beyond the scope of this paper.

5. User Scenarios for Query and Populate

The user roles show a typical pattern of searching for resources, accessing and using the discovered resources, and cataloguing the products of use. The query and populate phases can be summarized as

- Query
  - find appropriate template
  - select properties of interest
  - supply target values
  - identify and access appropriate resources from list of query results (or access immediately if saved from past query)
- Populate
  - find appropriate template
  - assign descriptive values (including error checking)
  - save result and have this entered into metadata registry

The use phase combines the query and populate phases as

- Use
  - query to find resources
  - perform primary tasks using resources identified and accessed during query
  - populate metadata to catalogue results of task

The identification of such generic activities implies opportunities for developing common mechanisms and tools that would be useful across the traditional partitions of responsibility. This is the perceived benefit of SOA, and here we present notional scenarios to begin to see where and how this can be applied.

5.1. The Query Phase

A user in the query phase may range from someone looking for a resource for the first time and needing the full power of a distributed query over a space of unknown resources to a user who may only need to verify that a previously discovered resource is still the appropriate one for the current task. Assuming a full search is needed, the user must first identify the class of resource wanted. This may be done by browsing a domain ontology/taxonomy for the closest concept/class or initiating a keyword search.[5] Once a resource class is identified, the user should be shown the properties from the associated schema (or ontology, etc.) used for generating metadata instances for this type of resource. The user can then supply target values for the properties of interest and be returned a list of metadata instances corresponding to the search criteria.[6] The user can retrieve additional metadata associated with any of the listed results, including constraints and policies associated with the resource and the means to access the resource which the metadata describes. Information related to constraints and policies may have already been used as part of the search criteria or may be evaluated separately to ensure appropriateness (e.g., whether terms of use are consistent between the resource provider and consumer, or whether bandwidth constraints make use of a video resource impractical).

A user may perform multiple searches to refine their idea of what is needed and to address different aspects of the current problem. The user may also perform a string search of the catalogued metadata (or of the resource itself for, say, a schema or a document), assuming a string index has been previously generated. It is likely that a metadata search would be used in tandem with a string search to either prioritize the results of a string search or to thin the corpus subject for the string search before the string search was initiated.

For the scenario as described, useful search capabilities would include:

- selecting any combination of defined properties as the search criteria;
- supporting logical combinations (e.g. and, or, not) of the selected properties;
- specifying the relative importance of properties selected for the criteria of any search;
- identifying a range of target values as well as single-value targets for each property;
- using a previous search result as the corpus for further search;
- supporting mediation among nonnumeric target values if the metadata was generated from diverse sources using diverse vocabularies;
- optionally saving the search so it could be re-executed in the future;
- optionally saving the results so newer results could be compared with previous ones.

5.2. The Populate Phase

The populate scenario assumes that appropriate metadata schemas have been created and catalogued by a schema developer. As with the query phase, the user must find the schema that provides a template for the metadata instance being populated or may access a schema that has previously been discovered. The user accesses the appropriate schema and assigns values to pertinent schema elements. It is assumed that standard fields in any metadata schema would be a textual description and relevant keywords to enable a keyword search. The query capability can be used to find one or more vocabularies/classifications that provide documented definitions for the keyword terms, and the metadata would capture not only the keyword but a link to the vocabulary documentation. In this way, the keywords add to an

[5] Note that a precursor search may have facilitated finding the ontology/taxonomy or the vocabulary where the keywords were defined.

[6] Note here we assume metadata has been created against a single schema but in a distributed environment, it is possible (and maybe likely) that metadata may have been created using several schemas. For such cases, a mediation “service” would be needed to facilitate consistent search across such metadata.

unambiguous description of the entity being described. When the metadata instance is complete, it would be incorporated into a metadata registry.

For the scenario as described, useful populate capabilities would include:

- selecting properties for which metadata instance values will be provided;
- mapping to data sources for bulk generation of metadata, and creating metadata to describe and register the mapping;
- auto-filling any values that are already known by the system, such as current user or mapping used for bulk generation of the metadata instance;
- informing the user of the range of values already registered for a given property and thereby alerting the user of possible inconsistencies;
- error-checking of supplied values, such as range and type of values or dependencies.

The description of the populate scenario implies using one schema as the source of metadata properties, but the metadata instance may be created using elements of more than one relevant schema. This would be consistent with, for example, the Extensibility Layer for DDMS [DDMS]. Given the modular approach described above for the assembling of metadata structures, the use of documented modules would maintain interoperability that might otherwise be lost by the introduction of isolated customizations. Note that the source schema should be clearly indicated for all properties used in a metadata instance.

5.3. Comments on Use

The query phase supports a user finding resources for their tasks and the populate phase enables a user to catalog the task products. The use phase is specific to the task and will probably depend on additional services, some of which are specialized to given domains or disciplines. The scenarios have indicated numerous opportunities for invoking query and populate, and it may be useful to have these accessible from within task-specific tools and services. In particular, it is more accurate and less of a burden on the user to collect metadata values while the user is performing their primary tasks than to require metadata capture as a separate process after the task is complete.

Note the scenarios refer to a “user” but do not specify whether the user is human or machine. If the business rules can be encoded for specifying search criteria, responding to the search results, and identifying and populating the relevant metadata, the process and capabilities as described could be equally applicable to human or machine users. If SOA implementations are eventually capable of dynamic composability, the machine user must be able to perform such tasks.

6. The Benefits and Feasibility of Query and Populate Focused Systems

This analysis began by looking at a diverse set of use cases and concluded with a factoring of metadata into concise reusable modules that could be combined to create more complex, specialized metadata structures. As a parallel, we also see that the tasks of users for whom metadata is an enabler but not a primary concern can similarly be factored to yield the basic functions of query and populate in support of domain-specific tasks. While it may seem contradictory to have populate as a basic function for tasks that are not primarily concerned with metadata, we can point to ways in which the query and populate functions can be more flexibly integrated with these tasks to increase automation and decrease additional work required of the human user.

When looking at currently proposed, single-purpose search systems, we see that these systems support associating values with search criteria but the criteria is an integral part of the system and the means to access the search functions are restricted to that system’s specific interface or APIs. Thus, a search for services is distinct from a search for content

which is again distinct from a search for people, and modifications to any of the embedded schemas require changes to the systems or using extensibility mechanisms that may negatively impact interoperability. However, the analysis has shown that from the perspective of the user, the basic query function is the same for any class of objects, differing only in the details of the schema(s) used to delineate the properties that describe the resource. Thus, a system that can flexibly incorporate schemas of interest (and document these choices) can provide generic search capability across any class of objects. A major benefit is that query and associated populate capabilities can be expanded to the mutual benefit of all users without a redundant modification for each separate query system.

As an example of the feasibility of this approach, Figure 1 shows a collection of archival figures from a system developed under the DARPA Affordable Multi-Missile Manufacturing (AM3) program during the 1990s and from which results were presented [AM3] at the XML ’98 conference in Seattle. Figure 1 shows (1) the presentation for a chosen class of its associated properties and the selection of a subset of these properties to comprise the search criteria, (2) the assigning of target values to each of the selected properties, including relative importance (Medium or High) and required closeness of match to the target values (Close, Exact), and (3) a ranked list of results including actual metadata values and links to the full metadata set available for the instance. The system was able to be repurposed for different domains by reading new domain ontologies with no change to the underlying search system. While the system was not fielded by DoD, it did demonstrate the concepts that today are described in terms of service-oriented architecture.

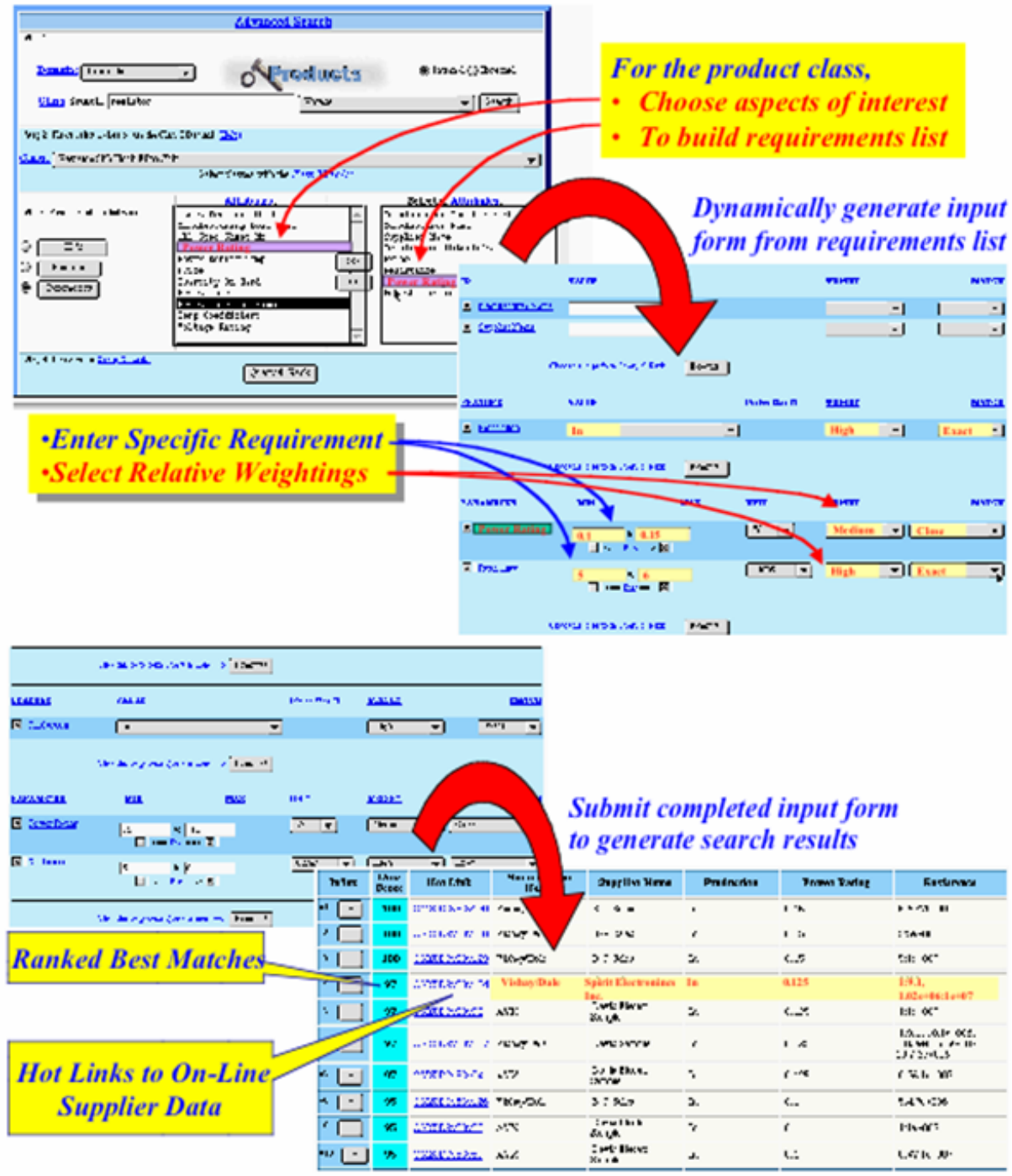

Figure 1. Search interfaces developed under DARPA AM3 program

Bibliography

[GIG-CES] ASD(NII) CIO, *Global Information Grid Core Enterprise Services Strategy,* Draft version 1.1a, 9 July 2003, http://www.defenselink.mil/nii/org/cio/doc/GIG_ES_Core_Enterprise_Services_Strategy_V1-1a.pdf

[DataStrategy] DoD CIO Memorandum, *DoD Net-Centric Data Strategy,* Version 1.0, 9 May 2003, http://www.defenselink.mil/nii/org/cio/doc/Net-Centric-Data-Strategy-2003-05-092.pdf

[AoA] *Net-Centric Enterprise Services (NCES) Analysis of Alternatives (AoA) Report,* 4 May 2004.

[OCD] *DoD Enterprise Metadata Operational Concept Description,* January 2005.

[DateTime] *Data elements and interchange formats - Information interchange - Representation of dates and times,* ISO 8601 : 2000.

[HR-XML] HR-XML Consortium, *Person Name 1.2 Recommendation,* 26 February 2003.

[XMLSchema] World Wide Web Consortium (W3C), *XML Schema Recommendations,* latest versions (28 October 2004). Linked at http://www.w3.org/XML/Schema#dev .

[DDMS] DASD (Deputy CIO), *DoD Discovery Metadata Specification (DDMS),* Version 1.3, 25 July 2005, http://www.defenselink.mil/nii/org/cio/doc/GIG_ES_Core_Enterprise_Services_Strategy_V1-1a.pdf

[AM3] Kenneth Laskey and Karl Seiler *Using XML for Open-Participation Supply Chains: The MISTI Solution,* presented at XML '98 Conference, Seattle WA, March 1998.

[metadata] Kenneth Laskey "Metadata Concepts to Support a Net-Centric Data Environment", **Net-Centric Approaches to Intelligence and National Security (Kluwer International Series in Engineering and Computer Science),** Chapter 3, Springer, 2005.

[oasis_symp] Ken Laskey *Modular Vocabulary Design in Support of Semantic Interoperability,* presented at OASIS Symposium, New Orleans, April 2005.

Biography

Kenneth **Laskey**
MITRE Corporation [http://www.mitre.org]
McLean, Virginia
United States of America

Ken Laskey began working on Web-based metadata cataloguing systems through a DARPA research project during the late 1990s and began examining the use of XML while the specification was still in draft form. He continued to investigate the use of the Web as the backbone of a distributed catalog and the particular challenge of converging and coordinating the diverse vocabularies of distributed contributors of information. This led to an interest in service-oriented architecture as a means to access distributed resources and effectively mediate between local vocabularies. The use of semantics technology to facilitate workable solutions is of particular interest because the local vocabularies often express specialized information and the challenge is to both preserve and make effective use of the subtleties and differences. Dr. Laskey is currently on the W3C Advisory Board and an editor for the SOA Reference Model TC. He is technical lead for the Information Semantics group at MITRE and a contributor to the description and analysis of mediation services for DoD's Net-Centric Core Enterprise Services (NCES).

The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.