NISTIR 7187

XML Schema Validation Process for CORE.GOV

KC Morris
Serm Kulvatunyou
Simon Frechette
Josh Lubell
Puja Goyal

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7187

XML Schema Validation Process for CORE.GOV

KC Morris
Serm Kulvatunyou
Simon Frechette
Josh Lubell
Puja Goyal
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

December 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Hratch G. Semerjian, Acting Director

White Paper
XML Schema Validation Process for CORE.GOV

KC Morris, Serm Kulvatunyou, Simon Frechette, Josh Lubell, Puja Goyal

Manufacturing Systems Integration Division,
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260, U.S.A

Abstract

Many integration projects today rely on shared semantic models based on standards represented using Extensible Mark up Language (XML) technologies. Shared semantic models typically evolve and require maintenance. In addition, to promote interoperability and reduce integration costs, the shared semantics should be reused as much as possible. The GSA Component Organization and Registration Environment (CORE.GOV) initiative is an effort to promote the sharing and reuse of components to reduce the acquisition costs of software needed by government. To be effective, CORE.GOV components must be consistent and valid in terms of agreed upon standards and guidelines. In this paper, we describe an activity model for validation of shared semantic models that is coherent and supports efficient enterprise integration. We then use this activity model to frame our research and the development of tools to support those activities. Overviews of these supporting tools are described primarily in the context of the W3C XML Schema. At the present, we focus our work on the W3C XML Schema as the representation of choice, due to its extensive adoption by industry. We believe this validation model and associated tools could serve as the basis for a CORE.GOV validation and acceptance process.

Table of Contents

1. Introduction

The Federal Enterprise Architecture Agency (FEA) Project’s Component Organization and Registration Environment, CORE.GOV, is a newly created resource intended to provide a collaborative environment for component development, registration, and reuse. CORE.GOV defines a “component” to be a “self-contained business process or service with predetermined functionality that may be exposed through a business or technology interface.” It provides a place to search for the components you need or to submit components for use by others. Reusability of components is the key to CORE.GOV and offers the potential to reduce software acquisition costs by leveraging work across multiple agencies. CORE.GOV is a private-public effort that grew out of the FEA Project Management Office. It was developed with the assistance of Collab.net and uses Collab.net's SourceCast tool, which provides a Sourceforge.net-like, open-source community for US government organizations starting with Federal agencies and including state and local entities. Although still in development, CORE.GOV could become a necessary infrastructural element for creating cost effective, interoperable, and reusable standards-based software solutions for Federal government agencies.

Reuse is one of the most compelling features of the World Wide Web Consortium’s (W3C) [W3C] XML (Extensible Mark up Language) [XML] technologies because it has the potential to save so much time. Developing new information elements in multiple contexts can consume countless hours. Component management solves that problem by allowing XML documents to reuse content across documents.[Nicholson] This is made possible by creating standardized and interchangeable parts with XML and employing a component management technique to provide intelligent access to components. In order to provide consistent, effective, reusable components, it will be necessary for CORE.GOV to provide some degree of component validation based on accepted standards, rules, and practices. This paper is an effort to develop a lifecycle model for XML schemas with emphasis on validation and approval activities; and tools to support those activities.

2. Model Development Life Cycle

In this section, we describe the highest-level activity model, called the *Model Development Life Cycle*, with particular attention to the inputs and outputs of this activity. They indicate the main objective of this activity and all subactivities (described in subsequent sections). The input is the *Data exchange requirements* and the output is the *Library of semantically coherent XML schemas* and *change requests.*

The *data exchange requirements* input includes all documentation that capture the detailed information requirements for integration. At this high-level, several kinds of models, such as use case models, integration activity models, object/information models, process models, etc., are considered part of the *data exchange requirements*.

The *library of semantically coherent XML schemas* output is a collection of data interchange terms and data structures represented as XML Schemas. These terms and data structures shall either have individually unique semantics or overlapping semantics and shall contain no duplicates. Those overlapping terms and structures should be related such as by extension, restriction, redefinition, or subsumption. The library may incorporate XML-based content standards and will include new XML content models. The resulting library also should contain supporting data to help maximize the reusability of these terms and data structures. These supporting data include but are not limited to classification schemes for categorization, the models provided in the information exchange requirements, sample instance data, more expressive semantic models, and documentation.

The *change requests* output is reflective of the cyclical nature of a life cycle. The other output, the XML schemas library, may incorporate XML content models, which are owned by external entities. In some circumstances one of the results of the model development life cycle will be requests to the owning entity to modify their model in order to fully cover requirements or maintain consistency. The result is the evolution of the library.

The figures in this paper are drawn using IDEF0 [IDEF0]. Included in the diagram from the top are constraints or control data used in the activity and from the bottom are tools and mechanisms supporting the activities. These control data and tools are briefly described below and will be expanded upon again in the subactivities.

XML Schema specification controls the syntactical and grammatical representation of terms and data structures for the data exchange specification. It also limits the expressiveness in which the relationships between overlapping data structures can be modeled.

XML Schema design guidelines enforce the resulting XML Schemas compliance to a selected set of design principles. These design principles can be ways of utilizing the XML Schema specification when alternatives exist, common data structure patterns, or required meta-data. While some of the guidelines appear to be mere stylistic options, their consistent use is critical to supporting schema reuse. These design guidelines bring bottom level consistency to the resulting schema and support ease of analysis, usability, extensibility, maintainability, automatability, and model expressiveness.

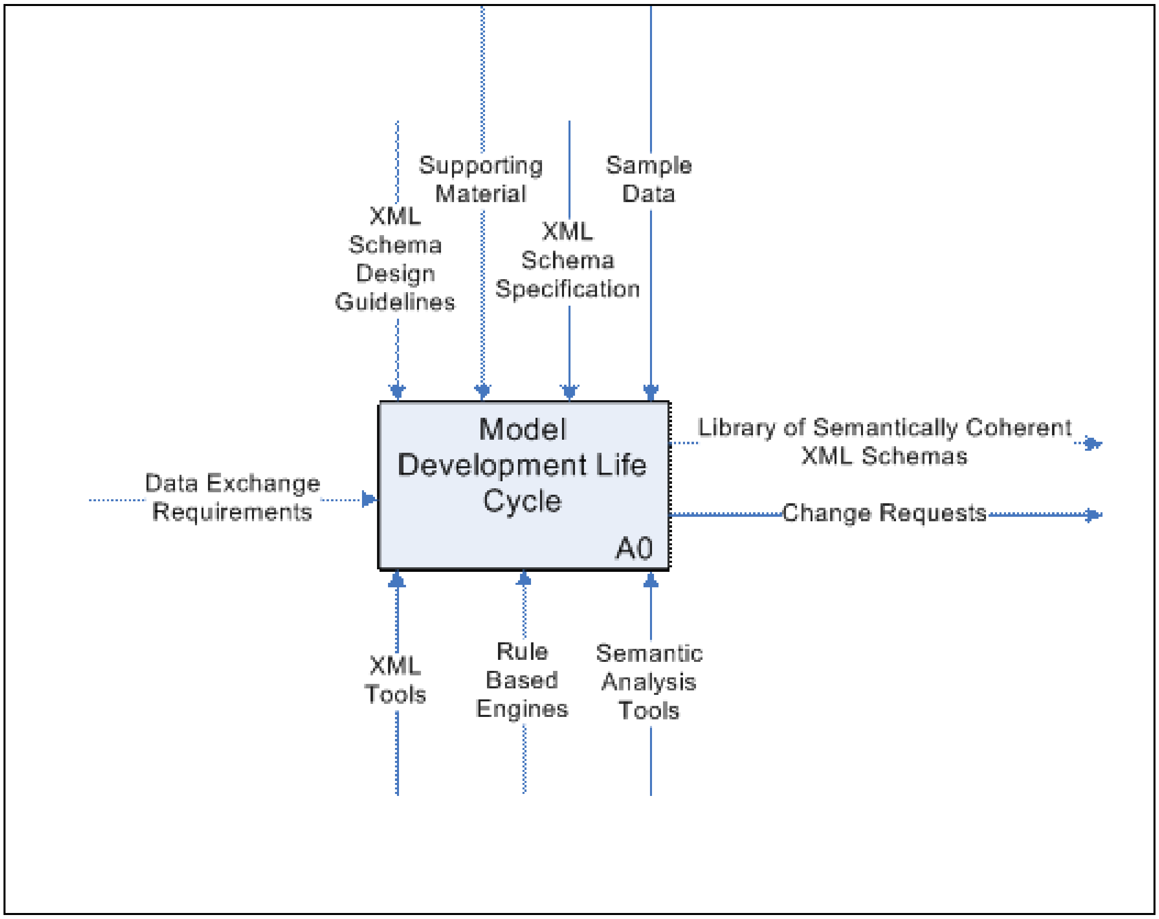
Supporting Material
Sample Data
XML Schema Design Guidelines
XML Schema Specification
Model Development Life Cycle
A0
Library of Semantically Coherent XML Schemas
Change Requests
Data Exchange Requirements
XML Tools
Rule Based Engines
Semantic Analysis Tools

Figure 1: Activity A0 – Model Development Life Cycle

Supporting material is the collection of source material for understanding the systems and data involved in the integration. It may include implementation documentation that clarifies the intent of the data, business rules for use of the data, classification schemas again clarifying the intent of the data, and external ontologies.

Although *sample data* may be viewed as part of the *data exchange requirements* input, the purpose here is as reference data to support requirement satisfaction and compatibility analyses.

XML tools encompass tools that implement the XML Schema specification. These include XML schema validators, XML parsers and validators, XML editors, and other tools that implement utility standards related to XML such as the XML Path language [XPATH] and the Extensible Stylesheet Transformation Language [XSLT].

Rule based engines are mechanisms to support the analysis of schemas conformance to design guidelines and other conformance testing requirements. Schematron is a specific example of a rule-based engine that is widely used with XML Schema.

Semantic analysis tools are quantitative and qualitative measures to enhance reuse of the semantic model or XML Schemas. They may support discovery, harmonization, and library management and maintenance.

One important note throughout this paper is that the activity names in the activity model are generic to semantic model representation. However, to keep our work focused all discussions are based on XML Schema as the semantic model representation mechanism and is indicated in the input, output, control, and mechanism labels. This does not preclude incorporating other semantic model representations into our research to assist in other activities.

3. Activities of the Model Development Life Cycle

The Model Development Life Cycle Activity *A0* is broken down into the six sub activities shown in Figure 2. These activities, *A1 – Model Requirements, A2 – Model Discovery*, *A3 – Model Validation*, *A4 - Model Piloting*, *A5 - Model Registration*, and *A6 – Model Integration*, are described in this section.

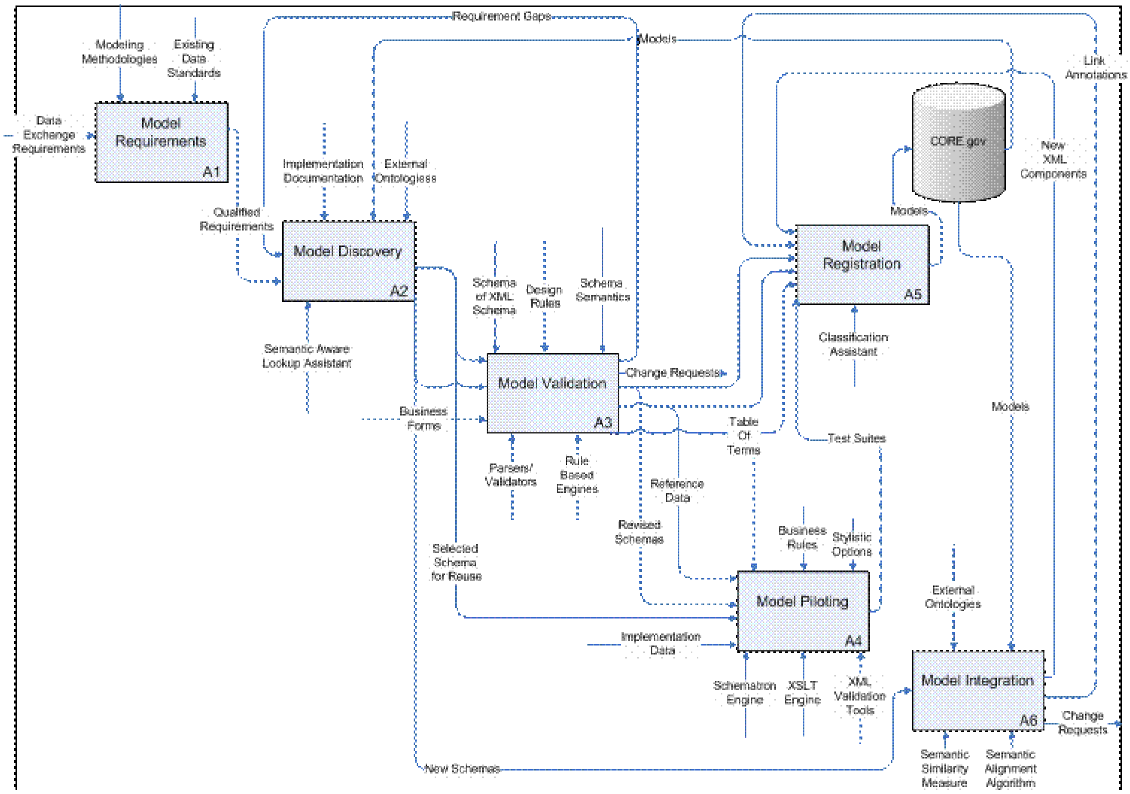

Figure 2: Decomposition of the Model Development Life Cycle

For the purpose of this paper we are considering the activities surrounding systems integration through data exchange using XML. We consider authoring of XML Schemas and requirements gathering as it relates to integration. We do not consider interactive systems integration, implementation of translators, model evolution, or retirement. The focus in this paper is on issues surrounding model reuse and validation in the context of a given integration project.

It is important to realize the following regarding model development:

- An XML Schema is mainly a syntactic device. It is not capable of representing the entire model's semantics. In order to represent semantics, the schema must be augmented with additional information. This information may take the form of rules, visual models, ontologies, supporting documentation, as well as the programming logic of an implementation of the model.
- XML validation and the processing of an XML instance document are distinct from one another. Furthermore, the processing of instance data can be independent of a given schema language, i.e., XML Schema. In other words, an XML document can be validated against multiple schemas, perhaps specified using languages other than XML Schema (e.g., Schematron, RELAX NG). In fact, validation can be thought of as a pipeline of various steps, where an application or user processes the XML document after it has completed all of its validation steps.

- XML validation refers to the validation of instance data represented in an XML document but model validation refers to validating an XML Schema against the requirements of the system or systems to be integrated.

These ideas form the basis of the Document Schema Definition Languages (DSDL) project (http://xml.coverpages.org/dsdl.html). DSDL is a project under ISO/IEC JTC1/SC34 Information Technology — Document Description and Processing Languages whose objective is to “create a framework within which multiple validation tasks of different types can be applied to an XML document in order to achieve more complete validation results than just the application of a single technology.” DSDL allows for a multi-step validation process that not only can involve multiple schema languages, but can also include transformations of the schemas as part of validation.

The idea of manipulating an XML Schema as part of validation is very powerful. This approach offloads the responsibility for ensuring interoperability from the schema developer onto the validation process itself. However, validation then becomes a more challenging task involving the pipelining and management of multiple steps. For an application with a large schema, validation resembles the building of software distributions from source code.

3.1. Model Requirements

Model Requirements marks the beginning of the Model Development Life Cycle. Identifying and documenting the business rules and data requirements are a necessary precursor to any piloting or implementation activities. The functionalities of the product or services to be integrated are outlined at this stage in order to capture the correct information. If this planning process is thorough in the beginning, it can save much time and energy when creating the actual schemas and instance data. Figure 3 illustrates the sub-activities of *Model Requirements*.

In the *Define Business Procedure A 1.1* sub-activity, business processes, systems, and transactions required of the model will be identified. *Identify and Gather Data A1.2* supplies data based on the business processes defined. In this activity the data elements, definitions, data types, data model, and other information are gathered for the data analysis matrix. The relevant data structures are also recognized.

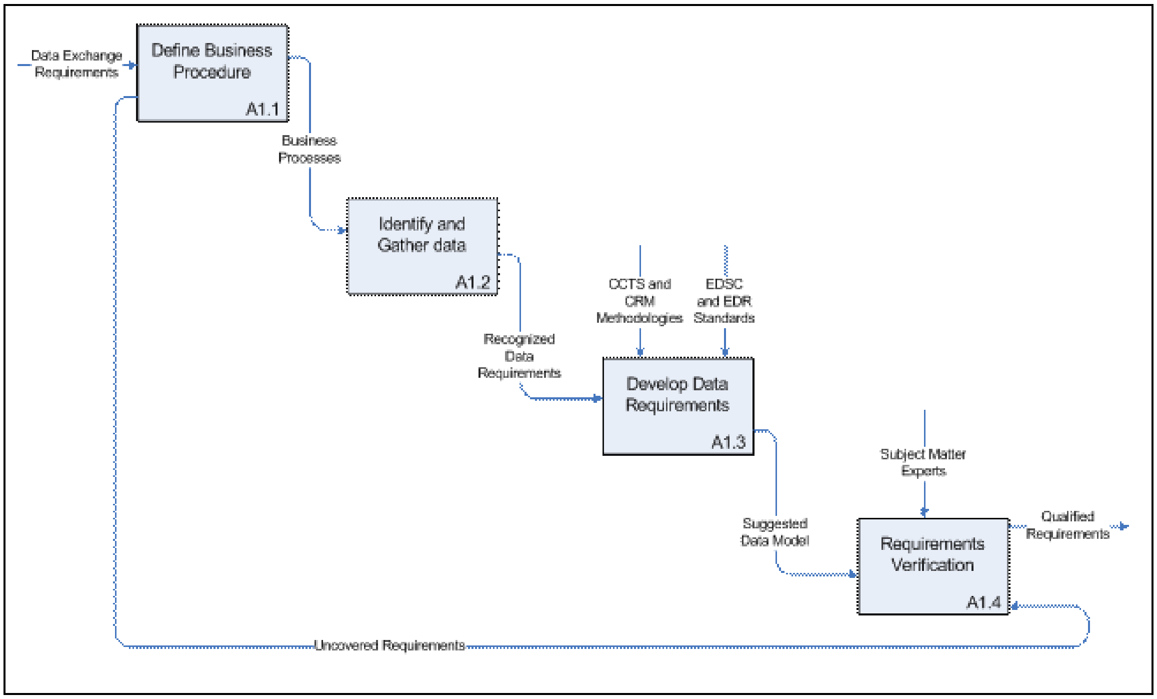

Figure 3: Activity A1 – Model Requirements

Once the data requirements have been identified and documented, data models representing this information are created as shown in sub-activity *Develop Data Requirements A1.3*. Here we also identify the practice of adopting The Environmental Data Standards Council (EDSC) and Environmental Data Registry (EDR) standards, as well as Core Components Technical Specification's (CCTS) and Core Reference Model's (CRM) methodologies. These various specifications encourage developers to use standard development practices and procedures which include setting data standards, assigning hierarchies in a matrix, and the naming of terms within an XML schema. The data models constructed are not necessarily in XML Schema format but contain the information needed to create the XML Schemas.

To ensure the data is represented comprehensively and accurately, sub-activity *Requirements Verification A1.4* verifies this data analysis with subject-matter experts. The final output of this activity are what we call qualified requirements– requirements that showcase the agreed upon version of the desired business rules and data. If any changes or additions are to be made anywhere from the business processes definition down to the data model, they are identified in this final sub-activity and the process is reiterated.

3.2. Model Discovery

Typically integration projects first try to identify existing XML Schemas that support their scope. If none are found, they may make the decision to build their own XML Schemas. Figure 4 depicts the activities of *Model Discovery*. The initial activity is *Model Selection*. This is either followed by *Model Extension* when a suitable model has been found or *Model Creation* when it is determined that an appropriate model is not available.

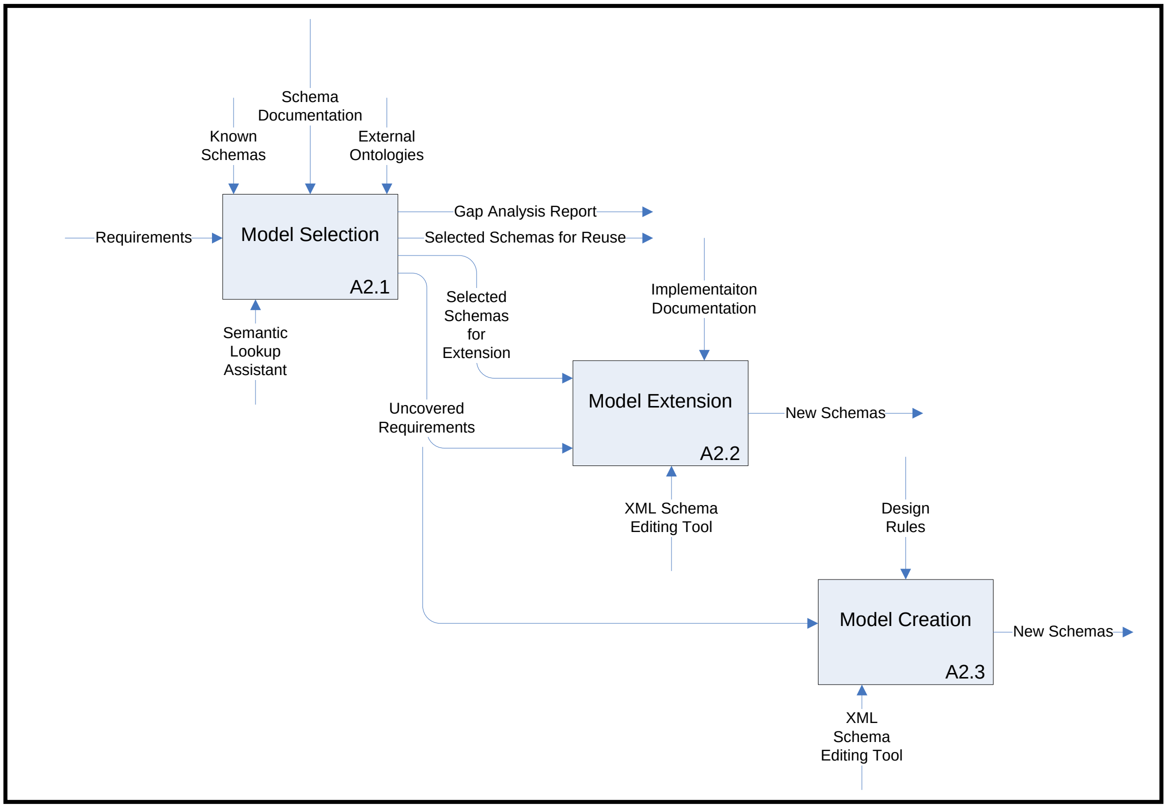

Figure 4: Activity A2 - Model Discovery

Model Selection involves finding a pre-existing model which meets the needs of the integration project. It can be a difficult process and integration projects may be tempted to skip it and create their own models; however, this conflicts with the goal of achieving interoperability with other systems. If a suitable model is available, it should be used to avoid integration problems with systems using it. The first activity under model discover should always be to find an integration model that fits the scope of the project and supplement or improve on it to meet the specific needs of the project as captured in activity *A1 Model Requirements*.

To make the discovery process less difficult we envision a tool called a *Semantic Lookup Assistant*. The semantic look up assistant would operate on schemas registered in a model registry using one or more classification schemes (see *Model Registration* below). A semantic look up assistant provides a search capability that goes beyond keyword search. For instance, it may provide a guided search based on question and answer interaction with the user. The questions asked would be based on the artifacts stored in the registry and the contexts used to drive the semantics associated with the schemas.

When models have been identified for use in the integration project, some of them may be selected for reuse “as is” but often they will need to be extended to support the full scope of the integration as seen in activity A2.2. The need for extension can be determined by analyzing the extent to which the selected model covers the data exchange requirements for the project. During this activity implementation documentation will also guide the processes of extending the schemas.

Activity A2.3 *Model Creation* is relatively straightforward and can be done using several publicly available tools. Some of these tools may be customized to tightly integrate with the schema design guidelines to assist the schema developer. Both the *Model Creation* and *Model Extension* activities result in new XML Schema files, which should then be validated as described below.

3.3. *Model Validation*

The *Model Validation* activity takes as input an initial information specification, e.g., the XML schema, produced by the *Model Discovery* activity. Just as with other types of software, before the schema is deployed it should be tested. Releasing a schema that is not of a high enough quality will result in frustration for both the users and the software developers and could result in failure of the entire project. However, unlike other types of software an XML Schema at this stage has no execution requirements; therefore, the *Model Validation* activity includes tests for quality of design. Figure 5 illustrates the sub-activities of *Model Validation*.

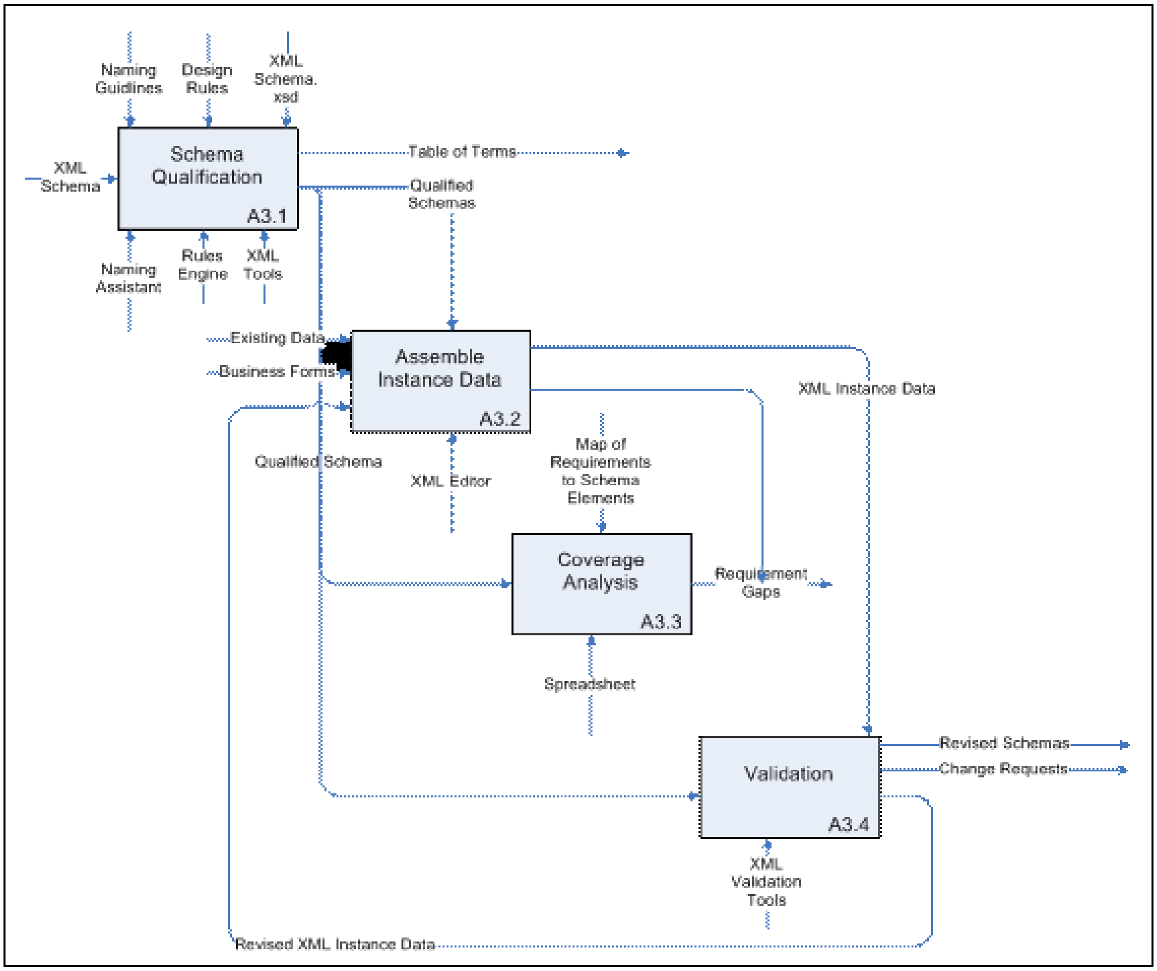

Figure 5: Activity A3 - Model Validation

Model Validation involves two types of quality validations. The first validation, represented in activity A3.1, is schema qualification. In this activity an XML Schema is tested against the standard specification for XML Schemas, xml-schema.xsd [XSD]. The XML schema is also checked for compliance with the project's design rules and naming conventions. This step ensures that modeling practices are used consistently which enhances the specification's intelligibility tremendously, thereby avoiding confusion during the piloting and implementation phase of the integration project. Naming conventions may be viewed as a form of design guidelines. However, their importance should not be underestimated and they, therefore, are called out. Modeling guidelines (including naming guidelines) should be established, documented, and enforced as early as possible in model development in order to avoid rework.

To support quality validation NIST has prototyped the three tools described below. Each of these tools represents a proof-of-concept prototype. Some work has been completed in designing enhancements to the tools based on our experiences.

Naming Assister. One result of the *schema qualification* activity is a table of terms to be used for naming in the XML schema. An initial table may have been provided by the *Model Discovery* process. NIST has prototyped a tool, known as the *Naming Assister*, to help with naming. The Naming Assister specifically aids in creating consistent compound names by verifying the construction of these names against a table of allowable terms. The table is based on extensions to the International Standardization Organization (ISO) -11179's recommended naming convention developed for the Automated Equipment Exchange (AEX) [AEX] Testbed. The tool was originally created to identify naming inconsistencies within the AEX Testbed's XML schemas and to assist in establishing a table of standard terms.

Schema Quality Assessment Tool. The XML *Schema Quality Assessment Tool* provides a repository of rules and a framework to publish and execute design rules. The repository has been loaded with an initial set of rules based on published "Best Practice" [Best Practices] guidelines for XML authoring resulting in a diagnostic tool for checking an XML Schema for compliance with the encoded guidelines. This experience has shown the possibility of extending the tool to support a larger set of rules, more complex rules, and the capability of creating an extensible rule set which can be tailored to the requirements for specific projects.

XML Validation Page. NIST prototyped an XML Validation page [Goyal] which would allow a user to upload XML instance files and have them validated against the content of a particular set of XML Schemas files using a selection of XML tools. This tool is similar to web pages made available by others with the important distinction being that it operates over a repository of XML Schema files for a specific project, NIST's AEX Testbed.

Activities A3.2-A3.4 represent the second type of validation that ensures that the model meets the original information requirements. The most direct way of doing this is to analyze the relationship between an XML schema and the application data. Activity A3.2 gathers instance data. Activity A3.3 maps that data into the XML Schema checking for complete coverage of both the data by the schema and the schema by the data. This is a manual process usually accomplished with the use of a spreadsheet to map from data fields in the systems to be integrated into the XML schema, and vice versa. The output from this activity is a requirement gap analysis that is fed back into *Model Discovery* and the process is reiterated. Activity A3.4 validates the data with the XML schema, and thereby validates that the XML schema meets the requirements represented by the data. In this phase of the model development life cycle when problems are uncovered in validating the instance data with the XML schema, the problems are often indicative of the problems in XML schema or its supporting material and not just in the instance data. Resolution of the problems should result in improvements to either the integration schema or the supporting documentation to clarify the intention.

Model Validation is an iterative activity the end result of which is a valid schema meeting a given set of quality criteria along with documentation describing the schema and how it is to be used including reference data. Reference data and naming conventions are extremely important to the success of a project. Therefore, we've made them required accompaniments to the XML schema at the end of the *Model Validation* activity, as is illustrated by the three input arrows to Activity A5 *Model Registration*. (Model Registration will be discussed further below.)

3.4. Model Piloting

Model Piloting focuses on how an integration model will be used in a given context. It involves supplementing an XML Schema with additional usage criteria specific to the processes to be integrated. It may also involve a simplification of the XML schema to make it more usable in the implementation context. This activity is especially important when the source of the implementation schema is external to the project (i.e., a standard schema used across an industry.)

Often when the time comes to use the integration models for integration, the implementers do not have freedom to modify the models directly for a variety of reasons. In this situation they often devise workarounds for addressing implementation issues. In this case, while the integration schema presumably covers most of the needs for the project, there may be either extensions that are necessary, conventions that need to be followed in the instance data, or the project may choose to modify the schema in a systematic way.

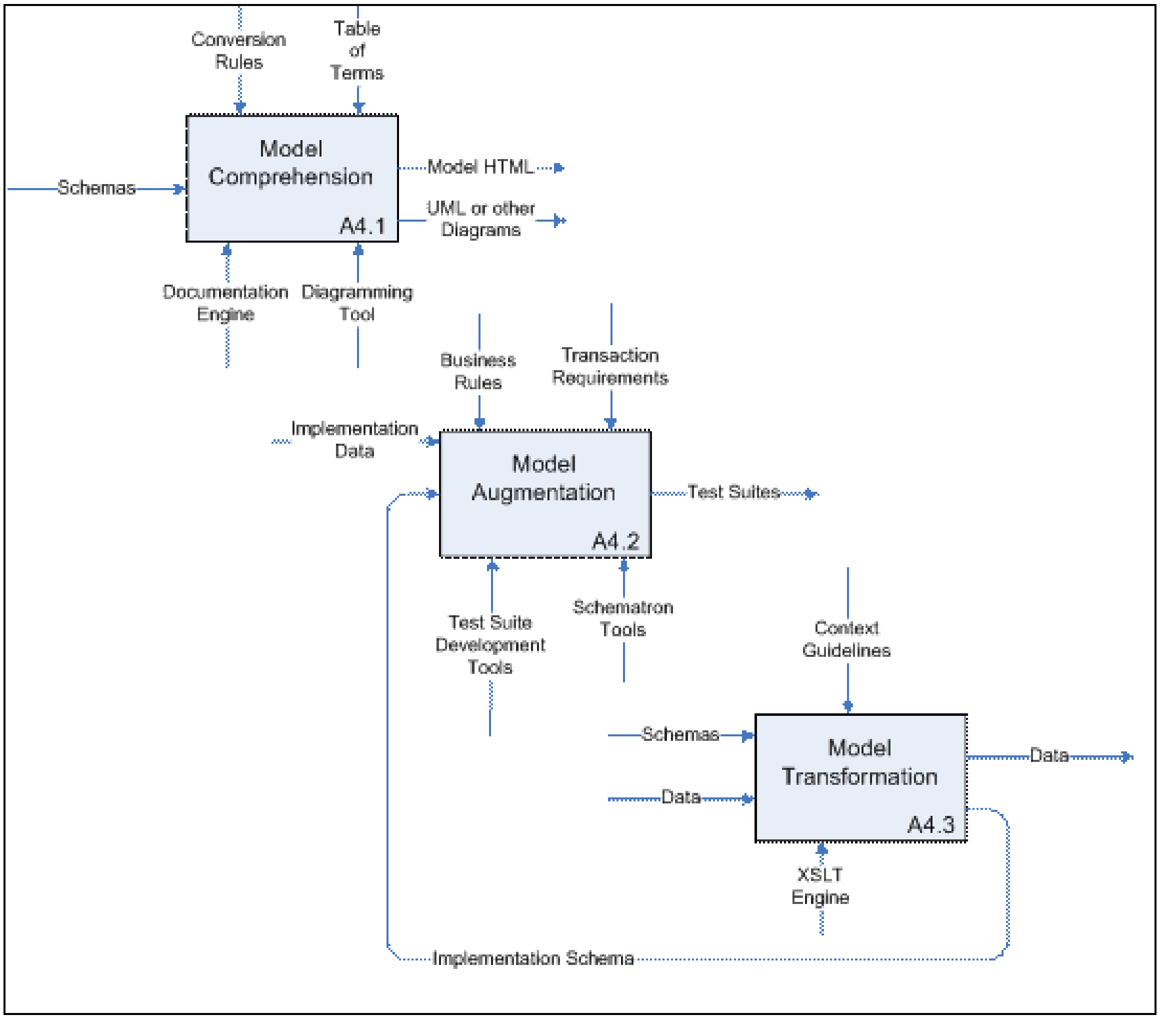

Figure 6: Activity A4 – Model Piloting

Figure 6 illustrates the three subactivities of *Model Piloting*. The first subactivity A4.1 *Model Comprehension* involves developing an understanding of the integration schema. Several types of tools, which generate various views of an XML schema, can assist a user to better understand an XML Schema. For example, one such tool can be used to create HTML pages that connect the various definitions in the schema through hyperlinked text [XSDDOC]. Another tool can be used to produce class diagrams of the structures defined in the schema [hyperModel].

Activity A4.2 addresses how to augment a model to specify business rules to be enforced during an exchange. These types of rules may not be generally applicable either across the industry nor during different types of transactions, yet there may be a requirement to enforce them at various times and for various purposes. For example, while a request for quote and a quote document share many of the same components, the former would not contain pricing information whereas the latter must. The *Model Augmentation* activity captures and codifies these rules and how they are to be applied. NIST has prototyped a tool, known as the *Content Checking tool,* to assist in this process in our B2B Testbed [B2BTtestbed]. The result of this activity is a test suite including the implementation schema, instance data, additional rules for validating the data based on the context, and guidance on how to use the schema in a given context.

Finally, activity A4.3 addresses *Model Transformation*. During *Model Transformation* an XML Schema can be transformed in a systematic way to support the needs for a particular implementation environment. Examples of when this may be desirable include the following scenarios:

- A project replaces the names used in a standard by terms more common to the businesses involved in the integration.
- An implementation group decides to use a single namespace or a namespace other than the one defined in the standard; this can also be accomplished through a transformation.
- An implementation group may prefer to work with a language other than XML Schema, such as DTDs.

Transformations may be performed on both schema and instance data resulting in a revised schema suitable for a specific implementation, which we will call an implementation schema, and revised data that corresponds to that schema.

The *Model Piloting* activities may or may not result in changes to the original XML schemas; however, they should surely result in improved artifacts, such as better documentation, better and more robust instance data, and guidelines on how to use the XML Schema in a given business context. Changes to the original schemas may be indicated if shortcomings of those schemas are uncovered.

3.5. Model Registration

The *Model Registration* activity organizes the schemas and related materials according to one or more classification schemes within a registry and stores the material in a repository so that it is accessible to other activities. Multiple classification schemes provide different perspectives of schemas just like the multiple Federal Enterprise Architecture (FEA) reference models. This supports a multi-dimensional and structured search of the registry; hence, discovery of the schemas is more efficient. The registry should not be viewed just as a versioning tool but a repository of stable and usable versions as shown in Figure 7.

An envisioned tool to help support the *Model Registration* activity is the classification assistant. Placing a schema into one or more classifications can be a tedious and error prone task. This task requires that the person understands the semantics of the classification schemas as well as his/her own schemas. Placing a schema in a wrong node in a classification not only makes the schema less accessible but also has a risk of misinterpretation by other users. In addition, placing a schema in too generic a node makes the *Model Discovery A2* activity less efficient by inundating the user with too many schemas. The classification assistant would use technology like a semantic similarity measure to provide suggestions for classification nodes to the user.

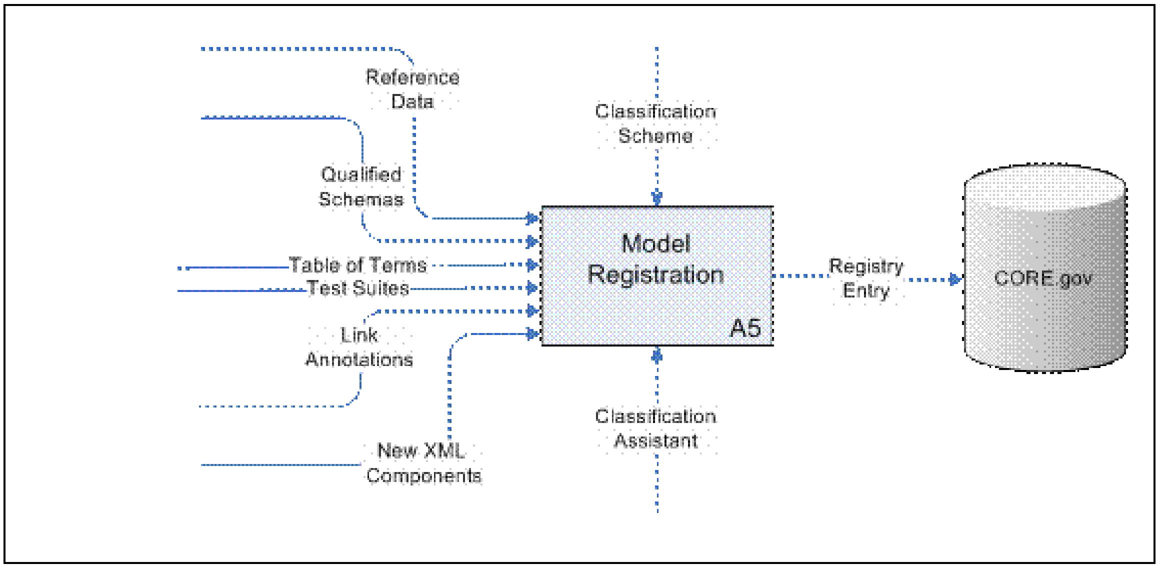

Figure 7: Activity A5 – Model Registration

3.6. *Model Integration*

The *Model Integration* activity is critical to supporting the evolution of an interoperability project. The objective of *Model Integration* is to ensure that new schemas and extensions are semantically coherent in the growing schema registry and repository. The general procedure for model integration is depicted in Figure 8. The first subactivity is to identify new terms and data structures that are semantic duplicates and/or overlaps. The second and third subactivities address how to resolve the duplicates and overlaps. The ultimate goal of model integration is to eliminate duplicates by requesting changes to the original schemas as shown in A6.2; however, when elimination is not desirable, such as when one or more of the schemas is already in use or is a standard controlled by an outside party, one must find alternative ways to handle the duplication such as by creating cross link annotations. Similarly in activity A6.3, the preferred approach to resolving overlaps would be to establish relationships within the schemas; however, that may not be a desirable or an achievable solution for similar reasons. In such case, cross-links between the overlaps should be annotated to ensure that the relationships could be identified and managed. Annotation tools based on XML Linking Language (XLink) [XLink] and Resource Description Framework (RDF) [RDF] may be used to allow computer interpretation.

Model Integration can be complex particularly when there is semantic ambiguity in the model or when part of the model needs to be restructured to accommodate a new relationship in the overlapping semantics. The tools we've conceptualized for the *Model Integration* activity include a *semantic similarity measure* and a *semantic alignment algorithm*. The semantic similarity measure provides assistance in activity A6.1 described above, while the semantic alignment algorithm supports activities A6.3. The semantic similarity measure assists in identifying the semantic duplication and overlaps by providing quantitative guidelines to the semantic proximity of terms. The semantic alignment algorithm could suggest the relationships between the new terms or structures and the existing ones and could also suggest how the existing model should be changed to accommodate the new relationship. Ongoing research such as Stuckenschmidt and

Visser (2000), Peng et al. (2002), and Ambite and Knoblock (1995) provides a basis for these two tools.

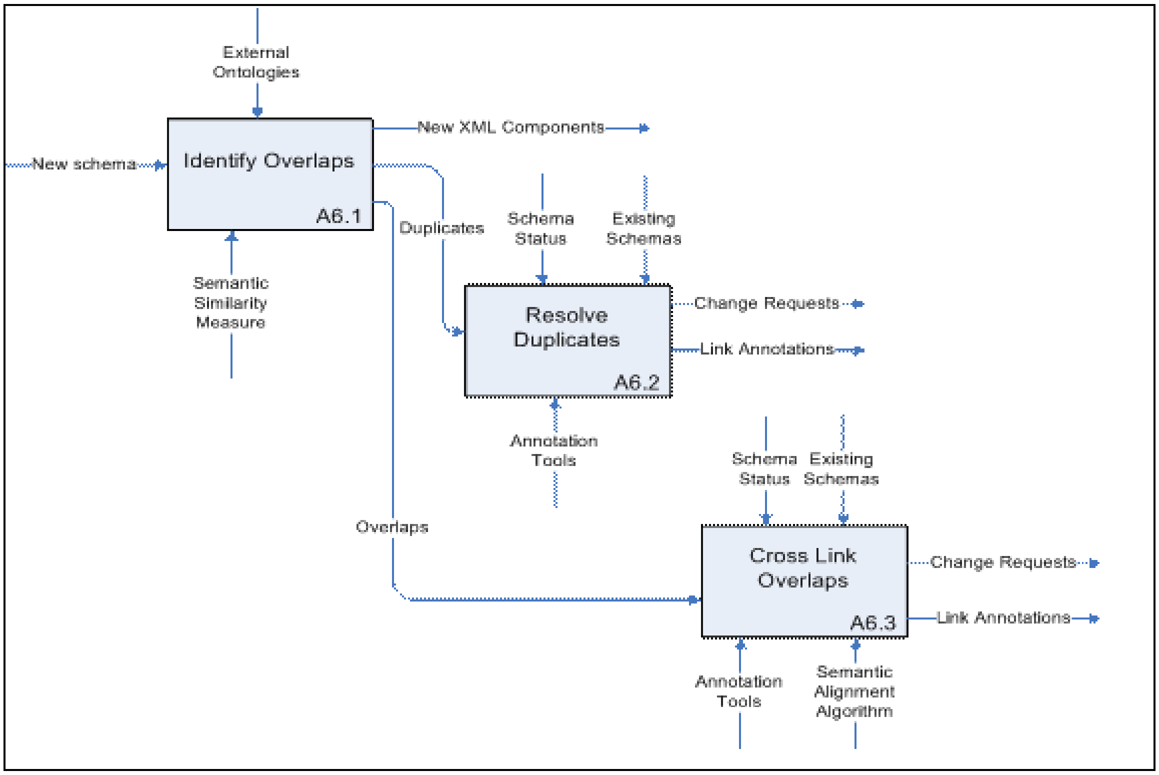

Figure 8: Activity A6 - Model Integration

4. Supporting Tools and Functionalities

The supporting software useful in the Model Development Validation Process is summarized in Table 1 and described below. Table 1 lists the tools needed by the process, the stage of development of those tools, and the source for the tools. The four stages of development in order of increasing maturity are research, prototype, beta, and production. Tools in the research stage are conceptualizations and may include some understanding of a basic design. Prototypes are proof-of-concept implementations of the tool. A beta stage tool is one that has been used by outside groups and NIST would be able to make source code available or support a limited number of users in some other way. Production tools are more available for mass consumption.

Tool	**Stage**	**Source**
XML Schema tools	Varied	Commercial and public domain
XML Validation page	Varied	Numerous generic pages; NIST has a prototype linking the validation feature with a repository of AEX schemas
Schematron engines	Production	Public domain
Schematron Editor	Beta	NIST
Naming Assister	Prototype	NIST
Content checking tool	Prototype	NIST
Schema Quality Assessment Tool	Prototype	NIST
Model transformation tool	Prototype	NIST
Classification assistant	Research	NIST and academia
Semantic lookup assistant	Research	NIST
Semantic integrity measure	Research	NIST and academia
Semantic alignment algorithm	Research	NIST and academia

Table 1: Tools supporting the Model Development Life Cycle

- XML Schema editors, parsers, validators, and related tools (XSLT engine) – these are readily available as both public domain and commercial tools.
- XML Validation page – numerous generic pages are available but these have limitations; NIST has a prototype linking the validation feature with a repository of AEX schemas. This supports both XML instance data validation and XML Schema extension.
- Schematron and the Schematron Editor – Schematron is a publicly available tool / language that we have found useful in augmenting information contained in XML Schema files. NIST has prototyped an editor for writing Schematron scripts.
- Naming Assister – a Naming Assister is under development at NIST with a prototype complete. This tool was originally written to identify naming inconsistencies within the AEX Testbed's XML schemas, and to assist in establishing a table of terms.
- Semantic checking tool – NIST has prototyped a tool (available through the Web) for specifying constraints on data and testing XML instance files against those constraints. This tool addresses concerns of interoperability between partners using different systems for enforcing constraints in their data. [b2btestbed]
- Schema Quality Assessment Tool – NIST has prototyped a quality of design tool which checks an XML Schema for use based on recommended design patterns [Kulvatunyou 2004]. This tool is diagnostic based on a number of "best practice"[Best Practices]

guidelines for XML Schema. Rule-based engines are used to specify and execute the design guidelines. We have used JESS [Friedman-Hill 2002] and Schematron [Jelliffe 2003] for prototyping activities.

- Model transformation tool – A tool called the Simplifier is being developed to transform schemas and test data according to proscribed design patterns. For example, the Simplifier "flattens" schema definitions using multiple namespaces into a single namespace. This is useful for exposing potential naming conflicts and inconsistencies. The Simplifier was originally developed to create a parallel set of schemas and data for schemas used in the AEX Testbed. Work is ongoing to make the Simplifier more generic so that it can be used for other applications.
- Classification assistant – NIST is actively researching the concepts for this tool and evaluating the requirements and complexity.
- Semantic lookup assistant – From monitoring business content specification forums and from interactions with implementers NIST has gathered requirements for the semantic lookup assistant tool. We see a significant need for a tool to assist users in identifying the appropriate XML constructs for their requirements and how to use those constructs in their own context.
- Semantic similarity measure – NIST has funded a few academic researches in this area and is still promoting the advancement of this technology. The initial research produced a quantitative measure for similarity between terms in object classifications.
- Semantic alignment algorithm – NIST is in the initial stages of investigating the potential of this technology. Most of the existing works today is in the academic arena.

5. Summary

NIST researchers are working to formalize the model development lifecycle with emphasis on testing and technological advancement to assist and mange the evolution and consolidation of large inter-organizational integration projects. NIST also has experience relevant to the FEA and CORE.GOV in several yet-to-be-addressed research areas including:

- Testing methods and frameworks
- XML validation/transformation frameworks
- Schema quality tools
- Semantic web technologies (metadata standards, inferencing, rule-based systems)
- Emerging semantic integration technologies

Additionally NIST has experience with and interest in industry outreach to promote reuse and interoperability within and across industries and government.

NIST is developing the tools described above on a small scale and with limited scope but plans to extend these to the larger community. We are also interested in finding the linkage between the model development life cycle and its software implementation counterpart in the pursuit of automating the change propagation from the schemas to associated software implementation.

In addition to the aforementioned tools, NIST is conducting research in automating the implementation phase of systems integration. NIST's AMIS (Automated Methods for Integrating Systems) project seeks to reduce the cost of integration where traditional standards-based approaches are inappropriate or ineffective. Algorithms and tools being developed for AMIS infer interaction models for incompatible systems via the systems' published interface specifications. Interaction models may in some circumstances be used to generate "glue code" needed to achieve integration. An AMIS prototype has been implemented to show automated integration for a Request for Quotation and Quotation Response scenario between a customer using CIDX (Chemical Industry Data Exchange Specification) and a supplier using OAGIS (Open Applications Group Integration Specification) [Libes 2004].

6. Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstrations purposes. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

7. References

[AEX] For a description of the Automated Equipment Exchange (AEX) project see http://www.fiatech.org/projects/idim/aex.htm.

[Ambite and Knoblock 1995] Ambite, J.L. and Knoblock, C.A., Reconciling distributed information sources. In Working Notes of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Palo Alto, CA (1995).

[b2btestbed] B2B Interoperability Testbed: Content Checker Tool; National Institute of Standards and Technology. Available online at http://www.mel.nist.gov/msid/b2btestbed/semantic_checking.html (accessed December 2004).

[CCTS] Core Components Technical Specification’s. Internet Web Site. Available online via http://xml.coverpages.org/CCTSv190-2002.pdf (accessed June 2004).

[Collabnet] Collaborative Software Development on Demand. Available online via http://www.collab.net/.

[CRM] Core Reference Model, Internet Web Site. Available online via <http://www.getf.org/file/wiser/2958.pdf >.

[EDR] Environmental Data Registry, Internet Web Site. Available online via <http://www.epa.gov/edr>.

[EDSC] Environmental Data Standards Council, Internet Web Site. Available online via <http://www.envdatastandards.net>(accessed June 2004).

[FEA] Federal Enterprise Architecture; Federal Enterprise Architecture Program Management Office. Available online at http://www.feapmo.gov/.

[Friedman-Hill 2002] Friedman-Hill, E., Jess the Rule Engine for the JavaTM Platform Version 6.0a8. Internet web site available online via <http://herzberg.ca.sandia.gov/jess/> (accessed July 2002).

[Goyal] Goyal, P., An XML Schema Naming Assister for Elements and Types, NISTIR 7143, (2004).

[hyperModel] XMLmodeling.com, HyperModel, Internet Web Site. Available online via http://www.xmlmodeling.com/.

[IDEF0] Integration Definition for Function Modeling (IDEF0), Draft Federal Information Processing Standards Publication 183, National Institute of Standards and Technology, December 1993. See also http://www.idef.com/.

[Jelliffe 2003] Jelliffe, R., The Schematron Assertion Language 1.5, Academia Sinica Computing Center. Internet web site available online via <http://www.ascc.net/xml/resource/schematron/Schematron2000.html>.

[Kulvatunyou 2004] Kulvatunyou, B.S., Ivezic, N., Jeong, B. (July 2004), Testing Requirements to Manage Data Exchange Specifications in Enterprise Integration – A Schema Design Quality Focus. The 8th World Multi-conference on Systemics, Cybernetics, and Informatics, Orlando, FL, USA.

[Libes 2004] Libes, et.al, "The AMIS Approach to Systems Integration: An Overview," NISTIR 7101, May 2004.

Nicholson, Simon, "The XML Assembly Line: Better Living Through Reuse," XML Europe 99, Granada, 1999 http://www.infoloom.com/gcaconfs/WEB/TOC/granada99toc.HTM

[Peng 2002] Peng, Y., Zou, Y., Luan, X., Ivezic, N., Gruninger, M., and Jones, A., Towards semantic-based integration for e-business, International Symposium on manufacturing and Applications, Orlando, FL (June 2002).

[RDF] World Wide Web Consortium. RDF Specification Development page. Internet Web Site, accessed June 14, 2004. Available online via <http://www.w3.org/RDF/#specs>.

[RelaxNG] ISO/IEC 19757-2:2003 Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG; ISO - International Organization for Standardization. See also http://relaxng.org.

[Stuckenschmidt 2000] Stuckenschmidt, H. and Visser, U., Semantic translation based on approximate re-classification. Proceedings of the Workshop "Semantic Approximation, Granularity and Vagueness, KR'00 (2000).

[W3C] The World Wide Web Consortium. Online at http://www.w3.org.

[XML] XML is a project of the World Wide Web Consortium. Available online at http://www.w3.org/XML.

[XSD] XML Schema is a project of the World Wide Web Consortium. Available online at http://www.w3.org/XML/Schema.

[XLink] World Wide Web Consortium. XML Linking LanguageRecommendation 1.0 (June 2001). Available online via <http://www.w3.org/RF/2001/REC-xlink-20010626/>.

[XPATH] World Wide Web Consortium. XML PATH Language Version 1.0 (November 1999). Available online via <http://www.w3.org/TR/xpath>.

[XSDDOC] Bluetetra Software, XSDdoc 2.0, Internet Web Site. Available online via <http://www.bluetetra.com/xsddoc/>.

[BestPractices] **The following references are sources for "Best Practices" guidelines for XML Schema:**

ASC X12C Communications and Controls Subcommittee (October 2002). *ASC X12 Reference Model for XML Design*. ASC X12C/2002-61

ebXML *Technical Architecture Specification v1.0.4*, 16 February 200, available as http://www.ebxml.org/specs/ebTA.pdf.

Korean Institute for Electronic Commerce. *Guidelines for Development of XML Electronic Messages in Korea* (March 2003). Available online via <www.xeni.co.kr/support/KIECGuidelineFinal_english_.pdf>.

Lockheed Martin Federal Systems (October 2002). *Global Combat And Support System – Air Force BOD Developer's Guide Draft Version 1.1*. Department of Air the Force Headquarters Materiel Systems Group (MSG).

OASIS UBL Naming and Design Rules Subcommittee (November 2003). *Universal Business Language (UBL) Naming and Design Rules*. Available online via http://www.oasis-open.org/committees/ubl/ndrsc/.

Roger Costello XML Schemas: Best Practices. Available online via <http://www.xfront.com/BestPracticesHomepage.html>

Roger Costello XML Schema Versioning. Available online via <http://www.xfront.com/Versioning.pdf>

Rowell, M., Feblowitz, M. (2002). *OAGIS 8 Design Document* (Draft 0.93), available online from http://www.openapplications.org/oagis/.

US Federal CIO Council Architecture and Infrastructure Committee, XML Working Group (April 2002). *Draft Federal XML Developer's Guide*. Available online via http://xml.gov/documents/in_progress/developersguide.pdf.