Development Life Cycle for Semantically Coherent Data Exchange Specification

Abstract

In enterprise integration, a data-exchange specification is an architectural artifact that evolves along with the business. Maintaining a coherent, data-exchange, semantic model is an important, yet non-trivial task. A coherent, semantic model of data- exchange specifications supports reuse, promotes interoperability, and, consequently, reduces integration costs. Components of data-exchange specifications must be consistent and valid in terms of agreed upon standards and guidelines. In this paper, we propose an activity model for the creation, test, and maintenance of a shared semantic model that is coherent and supports scalable standards-based enterprise integration. While it frames our research and the development of tools to support those activities for semantic models implemented using XML (Extensible Markup Language) Schema, the activity model presented in this paper is independent of the data-exchange technology.

1. Introduction

The motivation for this work comes from experience in working with industries to develop standards for data exchange [15]. We have found that data-exchange models or specifications evolve in a fragmented and distributed fashion. To make integration and interoperability more efficient and scalable, the fragmented specifications need to fit into a coherent, semantic model. That is, they need to be logically consistent and contain minimal duplication. Additionally, semantically overlapping data structures should be

related or annotated. In our previous work [15], we proposed an activity model capturing the activities involved in creation and maintenance of a coherent, semantic model of data-exchange specifications (DES). The focus of this paper is a robust life-cycle- activity model that is independent of a particular data-exchange specification and implementation technology. The DES Development Life Cycle (DDLC) is proposed as an activity model represented using the IDEF0 [19] framework.

2. DES Development Life Cycle

The highest-level activity, called the *Manage DES Development Life Cycle* is shown in Figure 1. In IDEF0, each activity box is defined with the inputs on the left, outputs on the right, controls/constraints on the top, and, mechanisms from the bottom. This section describes the inputs, controls, outputs, and mechanisms involved in the activity at their highest level. These are further detailed in subsequent sections. The objective of the activity is to design, create, extend, or modify data-exchange specifications for systems integration projects such that data/content level interoperability is achieved over multiple specifications. This, we believe, will reduce integration costs over the long run. To achieve that reduction, we assert that the activity must produce and maintain a library of semantically coherent DES as the integration projects evolve.

2.1. Inputs

The Data Exchange Requirement (*DER*) input includes detailed information requirements for integration. The DER may be captured in a number of different data models including class diagrams, database schema, and, entity-relationship diagrams. We note that DER only refers to the data model in this discussion context; other kinds of related models

such as activity, interaction, or use-case models are considered as part of the Integration Requirement Document described below.

The *Sample Exchanged Data* is representative data to be exchanged in the actual integration scenario. The more sample exchange data in hand the better the quality of the DES produced.

<Insert Figure 1 here.>

2.2. Outputs

The *Library of Semantically Coherent DES* output is a collection of data-interchange terms and structures captured in a computer interpretable representation such as XML Schema [31]. Terms and data structures should contain unique semantics, overlapping semantics, or properly annotated duplicate semantics. Overlapping semantics must be clearly represented. Where direct relationships cannot be established internally via the DES normative representation or duplications cannot be eliminated, they must be properly annotated.

The *DES Supporting Materials* include information kept along with the DES to help maximize the reusability and comprehensibility of its terms and data structures. Supporting data include, but are not limited to, a table of terms (controlled vocabulary), classification schemes for categorization, DER and integration requirements documents, DES documentation, sample exchange data, and, more expressive semantic models. Further details of the supporting materials will be apparent when we describe the subactivities in later sections.

The final output, *Test Suites*, is important for enabling standards-based integration[1]. Testing is an indispensable activity in systems integration [8, 10]. The test suite provides test data and other materials needed for integration testing.

2.3. Controls

Controls (represented by the arrows entering from the top of the activity box) are the relevant materials needed to constrain and guide the DES outputs.

The *Target Applications* are the systems to be integrated. They provide detailed data-exchange requirements such as sample exchange data. Target applications are also used to pilot test the DES to determine requirement satisfaction.

The *DES Meta-Model and Design Guide* refers to syntax, semantics, and best practices for encoding/capturing the DES as a normative representation. For example, if the DES is encoded in the XML schema, its syntax and semantics specification must conform toW3C XML Schema specification. Best practices may come from a design guide such as the UN/CEFACT Naming and Design Rules (NDR) [26]. The design guide may suggest a subset of features in the DES representation to use, and/or specific ways to construct data structures and terminology that support comprehension, reuse, maintenance, and ultimately, interoperability. We note that a selected DES representation limits the expressiveness with which the relationships between overlapping data structures can be modeled. Based on the selection, it may be necessary to supplement the

[1] We refer to standards-based integration as the integration approach that relies on implicitly agreed upon DES semantics (i.e., complete and formally expressed semantics of the DES is not available).

DES with another representation to annotate the relationships. Recall that these annotations are part of the DES supporting materials output.

The *Integration Requirement Document* includes typical artifacts and models captured in traditional software development and systems integration projects. Examples include use-case models, activity models, interaction models, and their documentation. The data models themselves are factored out separately as an input to the activity.

The *External Ontologies* can be used to further explicate the semantics of the DES. Within this paper, an ontology is a set of logical statements providing formal descriptions of terms and data structures. A mechanism to link these external ontologies to the DES is required [6].

The *Classification Schemes* are taxonomies of categories and attributes. Examples of classification schemes are business area classifications, subject classifications used in the library, and, products and services classifications. They constrain how DES components should be related and where they should be stored in the DES library.

The *Business Rules* (also known as context rules) capture usage and data constraints when a DES is used in different transactions and environments. For example, the same purchase order DES can be used in several different countries. Nevertheless, one country may require sales tax while another country may not. Business rules must capture such differences.

2.4. Mechanisms

Mechanisms are methodologies and/or tools facilitating DDLC activities.

Editing/Encoding Tools refer to those tools involved in the construction of the DES. They

may include tools that automatically generate a DES from a graphical representation, or tools that support authoring and validating the DES directly. A number of commercial tools are available.

Rule Engines provide a utility to test for conformance to DES syntax, semantics, design guidelines, test suites, and other requirements. Schematron [14] is a specific example of a rule engine that is used with XML standards. A traditional, rule-based expert system such as the Java-based Expert System Shell (JESS) [9] may be used as well.

Validation Tools provide functions similar to the rule engines. The difference lies in the techniques they use. Validation tools use techniques designed to verify conformance based on a model rather than rules. XML Schema and XML validation parsers are good examples. Parsers verify conformance of an XML instance based on a grammatical model specified by an XML schema. The term “validation tools” refers both to DES validation and DES instance validation tools.

Semantic Analysis Tools mean those tools capable of providing analysis of semantic similarity between terms and data structures. This analysis may be complicated by different modeling perspectives, modeling approaches, and usage contexts. This term encapsulates the Semantic Aware Look up Tools, Semantic Similarity Measures, and Semantic Alignment Tools used in subactivities.

Test Suite Development Tools refer to functionalities used to generate and validate test cases and analyze test coverage. Test- case generation may be manual, automated, or semi-automated. Typically, multiple steps are recommended before obtaining test cases such as functional-requirements extraction and test-assertions authoring.

Documentation Tools are tools that help create supporting materials for the purpose of enhancing the understanding of the DES in either a formal or an informal form. This can be anything from a word-processing tool to a graphical-representation tool. This term encapsulates Annotation Tools and the Diagramming Tools described in subactivities.

Transformation Engines are utilities used to create DES variations. A transformation typically amounts to a change in the format including the look, labels, or structures of the DES rather than any change in semantics. This may involve going from a textual representation to a spreadsheet or hyperlinked webpage.

Test Tools refer to those tools used in pilot testing. The functionality may include choreographing test scenarios, initiating test messages, simulating counterpart application, logging and monitoring message traffic, and verifying compliance of DES messages.

3. Activities of the Manage DES Development Life Cycle

The manage DES development life cycle activity (*A0*) is decomposed into the five subactivities shown in Figure 2.

<Insert Figure 2 here.>

3.1. Discover DES

The DES discovery activity targets reuse of existing DES. Reusing DES is strategically important for minimizing long-term interoperability costs. A new *DER, Requirement Gaps,* or *Change Requests* might be needed when using the results of other activities as inputs to the discovery activity. The activity is decomposed into three subactivities: *Select DES for Reuse, Extend/Adjust DES,* and *Create New DES* as depicted in Figure 3.

The select DES for reuse activity searches the *Library of Semantically Coherent DES* for modes that closely match the new DER or requirement gaps propagated from a requirement coverage analysis. We envision having an intelligent *Semantic Aware Lookup Tool* to assist the domain expert or system engineer with this activity. The tool should exploit other information associated with the DER and DES in finding such matches. That information includes text descriptions, integration models, sample data, *Classification Schemes*, and other expressive logical axioms describing the semantics of the DES data elements in the *External Ontologies*.

One outcome of the DES selection activity is that the DES can be used as-is. This outcome is preferable, because there will not be a new DES to maintain. In addition, the integration will be eased, because the new interface implementation can be readily connected to the same interfaces already implemented.

In many cases, the existing DES from the library only partially supports the new DER. In such cases, a partially *Reusable DES* is extended or is adjusted to meet the *Uncovered Requirements*. This is the objective of the *Extend/Adjust DES* activity. An extension is an addition to an existing DES. An adjustment is a direct modification to the reusable DES, such as a relaxation of some constraints to accommodate a new type of data. For uncovered requirements that have no relationship to existing DES, a *New DES* is created.

For both the extend/adjust DES and create new DES activities, the DES documentation helps the domain expert or system engineer to perform the activities by providing additional descriptions of the existing data elements in the DES. The *DES NDR guidelines* ensure that the DES is produced with best practices and consistent with the

existing DES library. The *DES grammar* enforces and constrains the syntax and semantics of the representation used to encode the DES. The *DES encoding tools* implement the DES grammar and DES NDR guidelines to aid the user in creating the DES. Some tools generate a DES from a graphical model or spreadsheet.

<Insert Figure 3 here.>

3.2. *Validate DES*

The objective of the *Validate DES* activity [17] is to ensure the quality of the DES including that the DES satisfies its requirements as laid out in the discovery activity. Figure 4 illustrates subactivities of the DES validation.

The *Qualify DES* activity assures that the DES follows the *DES NDR Guidelines*, does not violate the *DES Grammar*, and is logically consistent - no conflicts or unsatisfiable specifications - according to the *DES Schema Semantics*. These controls are part of the DES meta-model and design guide explained in the higher-level activity A0 in section 2.3.

This activity may seem redundant with the creation activity, which has the same controls; however, it is a very practical step. If the DES has been developed using a specific tool that enforces the grammar and NDR Guidelines then this step may seem unnecessary; however, it is particularly useful when different tools are being used by project partners. The qualification activity checks that a DES is compatible not only with the tools used in its development but also with others that will be used by other activities in the project before the DES is disseminated widely.

To avoid rework, the NDR guidelines should be established, documented, and enforced as early as possible in the *DES* development. The DES qualification may also be viewed as an activity performed by a group maintaining the library of semantically coherent DES. Only a high-quality DES that conforms to the NDR and is in other ways error free should be allowed into the library for reuse by others. These guidelines ensure that modeling practices are used consistently. They enhance the specification's understandability and help avoid confusion during the pilot and implementation phases of the integration project.

DES grammatical compliance is assured by the *DES Validation Tools*. These are parsers or parsing functions in DES editing/encoding tools. The *Rule Engines* are useful for capturing and ensuring compliance to the DES NDR guidelines. They may also be used for capturing and ensuring compliance to the DES schema semantics, which require expressivity beyond parsers' functions. The *Naming Assistant* is a tool specifically designed to validate DES element names. It can be used in conjunction with a *Table of Terms* (also called controlled vocabulary or data dictionary).

<Insert Figure 4 here.>

The other DES validation subactivity is the *Analyze Data Coverage*. Its purpose is to ensure the new, extended, adjusted, and/or selected DES has sufficient data structures to capture the intended DER. The most direct approach to implement this activity is to analyze the relationship between the DES and the *Sample Exchange Data* (i.e., the application data). The analysis validates that the DES can be instantiated with the sample exchange data and the DES is not constrained (over specified) to the point that the sample data is invalidated. For this approach, the *Data Production Tools* include data editing

and/or data generation tools that generate sample exchange data from target applications in the DES compliant representation. The *DES Validation Tools* are then used to determine if the sample data is invalidated. *Business Forms,* which contain existing data, are one source of sample exchange data. Another popular approach is to analyze the coverage only at the DES (schema) level. Typically, in this approach, a spreadsheet is created to map DES data elements to the DER data elements. Although this approach ensures the data element coverage, it does not verify whether the DES is potentially over specified for the DER. An alternative approach is to translate known DER constraints into constraints represented as DES data elements and then determine if there are any logical conflicts with existing DES constraints.

Model Validation (A2) can take much iteration, but the end result is a *Validated DES* meeting a given set of qualification criteria along with other artifacts illustrating the DES and how it is to be used. These artifacts are the *Validated DES Instance Data* created as a reference from the sample exchange data and the table of terms containing controlled vocabulary and data-element definitions.

3.3. Maintain Semantic Coherence

The *Maintain Semantic Coherence* activity is important as a long-term interoperability strategy. The activity can be viewed as a monitoring or certification function before the DES and associated artifacts are made available for reuse. The overall goal of the activity is to ensure (1) that the new, extended, or adjusted DES does not conflict with existing uses; and, (2) new terms or data structures that are semantic duplicates or overlaps with existing ones are not created without proper relationship and documentation.

Consequently, this overall goal is decomposed into two subactivities, the *Analyze DES Compatibility* and *Integrate DES,* as shown in Figure 5.

<Insert Figure 5 here.>

DES compatibility analysis ensures that the *Validated DES* produced from the earlier activity is compatible with any existing uses. A DES versioning scheme should be documented in the NDR guidelines to help ensure compatibility. In some circumstances, compatibility is broken in order to achieve semantic coherence. In such cases, clear versioning indicates potential incompatibility and additional intelligence can be built into the corresponding interfaces to handle this. The output from the compatibility analysis is either a *Compatible DES* when there are no compatibility issues or a *Change Request* to fit the new DES into the broader semantic model. As described earlier, compatibility may be left broken. The decision, which is a business as well as technical decision, depends on the long-term impact on interoperability, migration strategy, development stage, and cycle time. Further discussion of this decision is beyond the scope of this paper.

Two approaches to the compatibility analysis have been suggested in the diagram. The first approach is an empirical one that uses a *DES Instance Validation Tool* and a *Library of DES Instance Data*. The existing DES instance data is validated against the adjusted/extended DES. If the existing instance data is not invalidated and it fully covers the applications, it is likely that there are no compatibility issues. The other approach is to perform a subsumption test. If the adjusted/extended DES subsumes the previous version of the DES in the *Library of Semantically Coherent DES*, then there is no compatibility issue - the new version is backward compatible with the previous version. Note that 'A'

subsumes ‘B’, if all possible instances of ‘B’ are also instances of ‘A’. An example is when a structure ‘A’ is simply a less restrictive version of ‘B’.

The compatible DES is fed into the *Integrate DES* activity to analyze and maintain the semantic coherence with the existing DES. The activity seeks to ensure that the DES does not create semantically duplicate terms or data structures and that any semantically overlapping term or data structure are properly related. Typically, the activity would first identify terms and data structures that are semantic duplicates and/or overlaps. Where possible, duplicates should be eliminated by sending a *Change Request* to the activity A1 to reuse. When elimination is not possible, such as when the DES is already in use or when it is a standard controlled by an outside party, *Link Annotations* are created across the terms or structures. Similarly, a preferred approach to resolving overlaps would be to restructure and establish a relationship using a schema construct available in the DES. When that is not possible, cross-links between the overlaps should be annotated to ensure that the relationships can be identified and managed. Consequently, if there is no change request to the earlier activity, the compatible DES and table of terms are output from the activity along with the link annotations where necessary.

Cross-link annotations may be assisted by *Annotation tools*. Depending on the DES representation, they may be simply a formatted documentation or they may be based on such technologies as XML Linking Language (XLink) [30] and Resource Description Framework (RDF) [28]. Using these technologies can make the cross-link annotation computer interpretable, consequently facilitating the Discover and Integrate DES activities.

Analyzing semantic duplicates and overlaps can be a complex and tedious task particularly when there is semantic ambiguity in the model. A manual approach would require the domain expert to comb through the whole library of DES for each term and structure in the new DES. We envision semantic analysis tools, such as the *Semantic Similarity Measures* aggregated in the *Semantic Alignment Tools*, to assist with this task. The semantic similarity measure assists in identifying semantic duplication and overlaps by providing quantitative guidelines for assessing the semantic proximity of terms and structures. Semantic alignment tools would (1) discover the relationships between the new terms or structures and the existing ones, and (2) suggest changes to accommodate the new relationships. Much research is on-going in semantic similarity measures [1, 3, 7, 21, 23, 24] and semantic alignment [2, 25]. These tools use information such as *External Ontologies* and *DES Documentation* to get more clues when comparing the target DES with the *Library of Semantically Coherence DES*. Since DES integration is a topic of ongoing research, the list of tools and reference materials here is by no means exhaustive.

3.4. Pilot DES

To solve a real integration problem, we must exchange information between specific software applications, which may impose additional requirements on the discovered DES. While the discovered DES presumably covers most of the DER, certain additions or modifications may be necessary (this is typically the result of maintaining a semantically coherent DES for reuse by others). For one, additional usage criteria specific to the applications being integrated may be needed. For another, adjustments to the deployment environment or community where the applications will be integrated may be needed. The

DES piloting activity deals with these issues. Figure 6 illustrates the four subactivities of piloting.

<Insert Figure 6 here.>

The first activity is *Enhance DES Comprehensibility*. Arbitrary DES representations can be difficult re for the systems integrator to understand. Providing a graphical representation can help reduce lead time in the development. Also a tabular representation of the DES elements that maps to the DER is often intuitive to developers. A combination of these diagrams with *HTML Documentation* is also popular. The HTML allows dynamic browsing through the DES elements and definitions. A number of *Diagramming Tools* with embedded *Conversion Rules* are available for producing such presentations from XML or other specification forms [4, 32].

The process of integrating a DES into the broader semantic model can leave the DES too generic. The *Augment DES* activity captures and codifies transaction-specific requirements on the DES. For example, the concept of a person in one application domain - like customer relationship management - may require more data elements than a person concept in a much simpler domain such as an address book. Consequently, when using the person data structure to exchange the address book, the augment DES activity codifies only the small number of elements that are used in the exchange transaction. The *Documentation Tools*, *Test Suite Development Tools*, and the *Rule Engines* help develop and verify these transaction-specific rules that may be based on *Business Rules* and the overall *Integration Requirement Document* where DES use cases are documented. The *Validated DES Instance Data* from A2 verifies that the output of the augmentation is not over or under specified. (Note that the *Transformed DES Instance Data* from A4.3 may

be used if transformation is needed for the implementation). The outputs from the A4.1 activity are modeled as inputs to the A4.2 activity because we assume that they will be aggregated into the *DES Documentation* along with the DES itself. The purpose of the DES documentation, often called an implementation guide, is to provide a single point of reference for systems integrators. The *Test Suites* output is an aggregation of the transaction-specific requirements, test scenarios derived from the integration requirements document, and DES instance data to be used as test data. The test suite has two main purposes: for the pilot test DES activity and for conformance and interoperability testing of applications using the same transaction context. The information contained in the DES documentation and test suites is mostly the same. However, the former is tailored to human comprehension during the interface development while the latter is computer interpretable for run-time testing.

The *Transform DES* activity may be necessary in certain environments. This activity may include DES simplification, terminology transformation, or different DES representation forms. For example, we may need to simplify the XML schema by flattening its namespace to make it work with specific integration software or middleware tools. Or, we might need to use domain-specific terminology in order to maintain the DES semantic coherence. Finally, different DES representations may be required if the run-time data exchange is in EDI (Electronic Data Interchange) [27] but the broader semantic library uses XML syntax. These specific requirements should be documented in the *Integration Requirement Document*. The DES output from this activity is called the *Transformed DES*. The *Validated DES Instance Data* is also an input to this activity, because the instance data similarly should be transformed (into the *Transformed DES*

Instance Data) and used in the DES documentation and test suites as well. These functions are assisted by *Transformation Engines*. Today's popular transformation technologies are based on the open standard extensible stylesheet language (XSL) family [29].

The other important subactivity of Pilot DES is to actually *Pilot Test DES* with *Target Applications*. In the pilot test DES activity, application developers follow the *DES Documentation* to implement the data-exchange interfaces. They then perform integration testing using the data in the test suites. Issues that may be discovered include (1) the DER was not documented correctly and (2) DES documentation is unclear or ambiguous, among others. A *Change Request* document that summarizes the findings from the test is generated and fed back to earlier activities.

3.5. Register DES

The *Register DES* activity organizes the DES and related materials within a registry and stores them in a repository that is accessible to other activities and users (see Figure 7). The inputs to this activity are those materials that are stored and maintained in the repository. Their definitions are as described in previous activities. Other supplemental information such as version, dependencies, associative semantics, and context information may be stored as well [33]. *Classification Schemes*, which are taxonomies, are typically used to categorize registered information. The taxonomies are typically domain specific. A piece of related information may be classified according to multiple schemes. This supports a multi-dimensional and structured search of the registry in order to make the discovery of DES more efficient.

Placing a DES and associated information into one or more classifications can be a tedious and error-prone task. Placing them in a wrong node in a classification not only makes them less accessible but poses the risk of misinterpretation by other users. In addition, placing a schema in a node that is too generic makes the DES discovery activity (A1) less efficient by inundating the user with too many options. Correct placement involves extensive understanding of the semantics of the classification scheme as well as the DES.

An envisioned tool to support the DES registration activity is the *Classification Assistant Tool*. This tool would use the semantic similarity measure described previously to suggest classification nodes to the user by matching the DES and associated information to a detailed definition in the classification scheme. This would narrow down the choices of classification nodes. The tool would support the user's decision-making process, which currently is based solely on node labels.

<Insert Figure 7 here.>

4. Conclusion and Future Work

We have developed many tools that support an XML Schema-based implementation of the DDLC with particular emphasis on the qualification activity [18]. Research to develop the semantic tools described above - such as the semantic lookup assistant and alignment tools - is beginning. This future work will allow the registry to be an active component in the enterprise data architecture rather than a static file store.

In the development of the DDLC we reviewed several widely available models with overlapping scope. The DDLC is similar to a software development life cycle [5, 8, 10]

and also shares commonality with some of the standards for development life cycles that we reviewed [11, 12, 13, 20, 22]. DDLC is unique in that it is somewhat a merger of these two perspectives. It focuses on the definition and description of a DES as well as software to support those processes and make use of the DES. While this review influenced the DDLC, a detailed discussion of this analysis is left for another paper.

We are confident that if an enterprise data architecture is designed according to the proposed life-cycle model, long-term interoperability cost will be contained or reduced while integration activities grow. Finding the right technologies for the Transform DES and the Integrate DES activities will allow the data architecture to endure changes. Current, popular, transformation technologies like the XSL [29] are too procedural, making it more difficult to build intelligence into data exchange interfaces, which use different DES forms and technologies to interoperate. Although we envision annotation technology like the XLink and RDF to provide declarative transformation information, we have yet to witness the evidence of major success. This will also be an element of our future work.

Product Disclaimer

Certain commercial software products are identified in this paper. This use does not imply approval or endorsement by National Institute of Standards and Technology, nor does it imply that these products are necessarily the best available for the purpose.

References

1. Alspaugh, T.A.; Anton, A.I.; Barnes, T.; and Mott, B.W. An integrated scenario management strategy. In *Proceedings of the 4th IEEE Symposium on Requirements Engineering (RE99)*, Limerick, 1999, pp. 142-149.

2. 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, 1995.

3. Bhavsar, V.C.; Boley, H.; and Yang, L. A weighted-tree similarity algorithm for multi-agent systems in e-business environments. *Computational Intelligence Journal*, 20, 4 (2004).

4. Bluetetra Software. *XSDdoc 2.0 Software*.

5. Boehm, B.W. Spiral development: Experience, principles, and refinements report. *Spiral Development Workshop*, Pittsburgh, 2000.

6. Dogac, A.; Laleci, G.B.; Kabak, Y.; Unal, S.; Beale, T.; Heard, S.; Elkin, P.; Najmi, F.; Mattocks, C.; Webber, D. Exploiting ebXML registry semantic constructs for handling archetype metadata in healthcare informatics. *International Journal of Metadata, Semantics and Ontologies*, 1, 1, (2004), 21-36.

7. Dong, X.; Halevy, A.; Madhavan, J.; Nemes, E.; and Zhang, J. Similarity search for web services. In *Proceedings of the 30th VLDB Conference*, Toronto, 2004, 827-383.

8. Evans, M.W.; Piazza, P.; and Dolkas, J.P. *Principles of Productive Software Management*. New York: John Wiley & Sons, 1983.

9. Friedman-Hill, E. *Jess the Rule Engine for the Java™ Platform Version 6.0a8*.

10. Gilb, T. *Principles of Software Engineering Management*. Reading: Addison Wesley, 1988.

11. International Standard Organization and International Electrotechnical Commission. *Systems Engineering -- System life cycle processes*. ISO/IEC 15288:2002.

12. Institute of Electrical and Electronics Engineers. *Industry Implementation of International Standard for Information Technology -- Software life cycle processes*. ISO/IEC 12207: 1995.

13. Institute of Electrical and Electronics Engineers. *IEEE Standard for Developing Software Life Cycle Processes*. IEEE Std 1074-1997.

14. Jelliffe, R. *The Schematron Assertion Language 1.5*.

15. Kulvatunyou, B.S.; Morris, K.C.; Buhwan, J.; and Goyal, P. Development life cycle and tools for XML content models. *XML Conference*, Washington DC, 2004.

16. Mandanis, G. and Wyatt A. *Software Project Management Kit for Dummies*. New York: John Wiley & Sons, 1999.

17. Morris, K.C.; Kulvatunyou, B.S.; Frechette, S; Lubell, J.; Goyal, P. *XML Schema Validation Process for CORE.GOV*, NISTIR 7187, 2004.

18. National Institute of Standards and Technology, Manufacturing Systems Integration Division. *MIP XML Testbed Web Site*.

19. National Institute of Standards and Technology. *Integration Definition for Function Modeling (IDEF0), Draft Federal Information Processing Standards Publication 183.* NIST, 1993.

20. Palmer, M.E., *Guidelines for the Development and Approval of STEP Application Protocol, version 1.2*. ISO TC184/SC4 N433, 1996.

21. Peng, Y.; Zou, Y.; Luan, X.; Ivezic, N.; Gruninger, M.; and Jones, A. Towards semantic-based integration for e-business. In *International Symposium on Manufacturing and Applications*, Orlando, 2002.

22. Tantara Inc. *Applicability of ISO 9001 to Software Development*.

23. Ryu, T.W., and Eick, C.F. Similarity measures for multi-valued attributes for database clustering. *In Proceedings of Smart Engineering System Design Neural Network, Fuzzy Logic, Evolutionary Programming, Data Mining and Rough Sets (ANNIE'98)*, St. Louis, 1998.

24. Schallehn, E., and Sattler, K.U. Using similarity-based operations for resolving data-level conflicts. In *Proceedings of BNCOD'o3*, Coventry, 2003, pp. 172-189.

25. Stuckenschmidt, H. and Visser, U. Semantic translation based on approximate re-classification. In *Proceedings of the Workshop Semantic Approximation, Granularity and Vagueness KR'00*, Breckenridge, 2000.

26. UN/CEFACT Applied Technology Group. *XML Naming and Design Rules Draft 1.1a*. UN/CEFACT, 2005.

27. UNECE United Nation. *UN/EDIFACT* Electronic Data Interchange Standard.

28. World Wide Web Consortium. *RDF Specification Development*.

29. World Wide Web Consortium. *The Extensible Stylesheet Language Family (XSL).*

30. World Wide Web Consortium. *XML Linking Language Recommendation 1.0.*

31. World Wide Web Consortium. *XML Schema 1.0.*

32. XMLmodeling.com. *HyperModel Product*.

33. Xu, Z.; Karlsson, M.; Tang, C.; and Karamanolis, C. Towards a semantic-aware file store. In *Proceedings of HotOS IX: The 9th Workshop on Hot Topics in Operating Systems*, Lihue, 2003, 145-150.

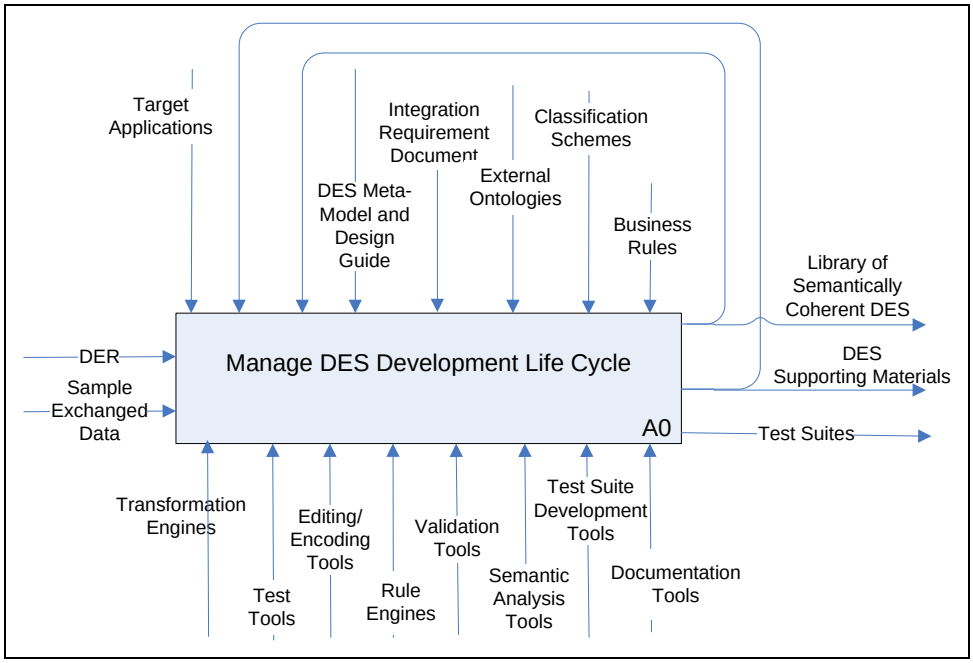

Figure 1: Activity A0 – Manage DES Development Life Cycle

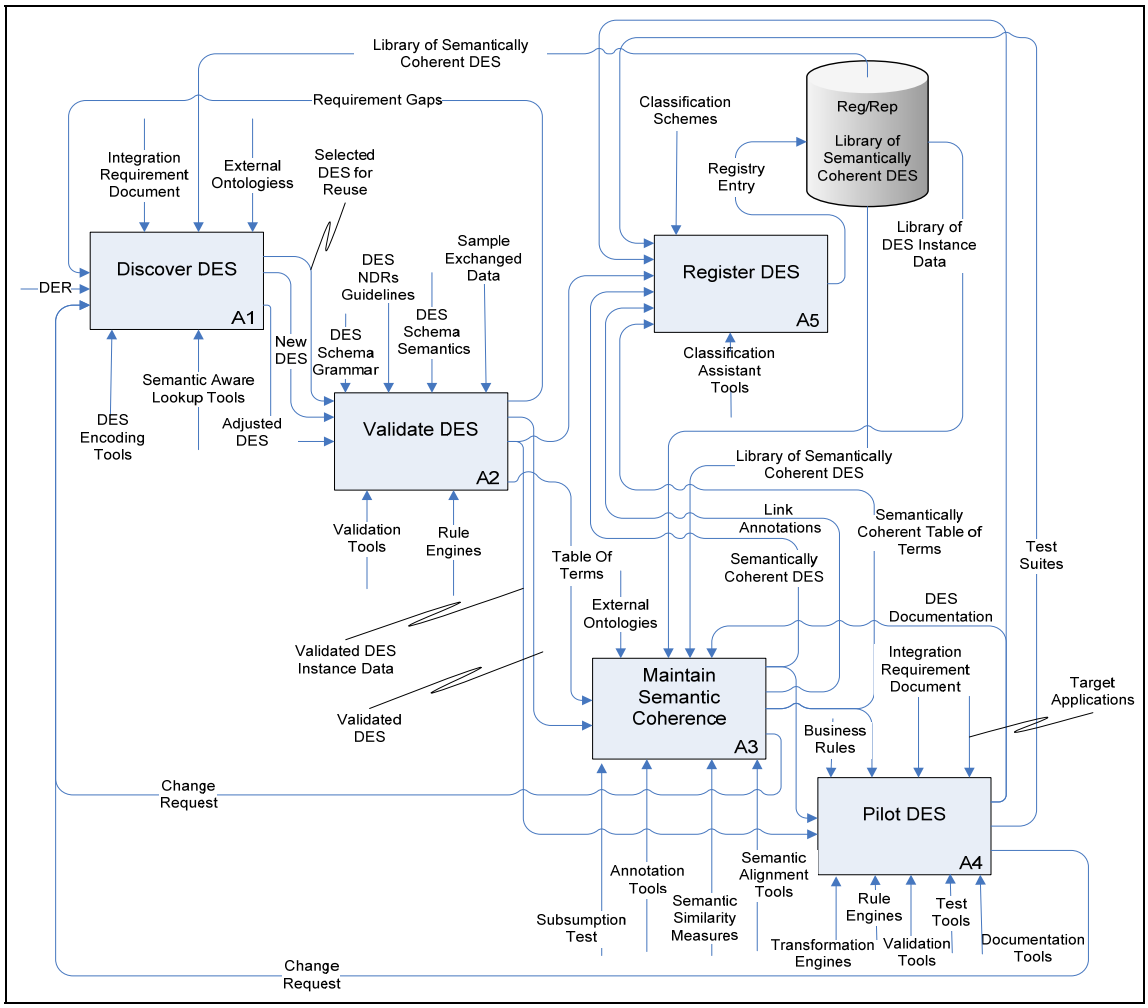

Figure 2: Decomposition of the DES Development Life Cycle

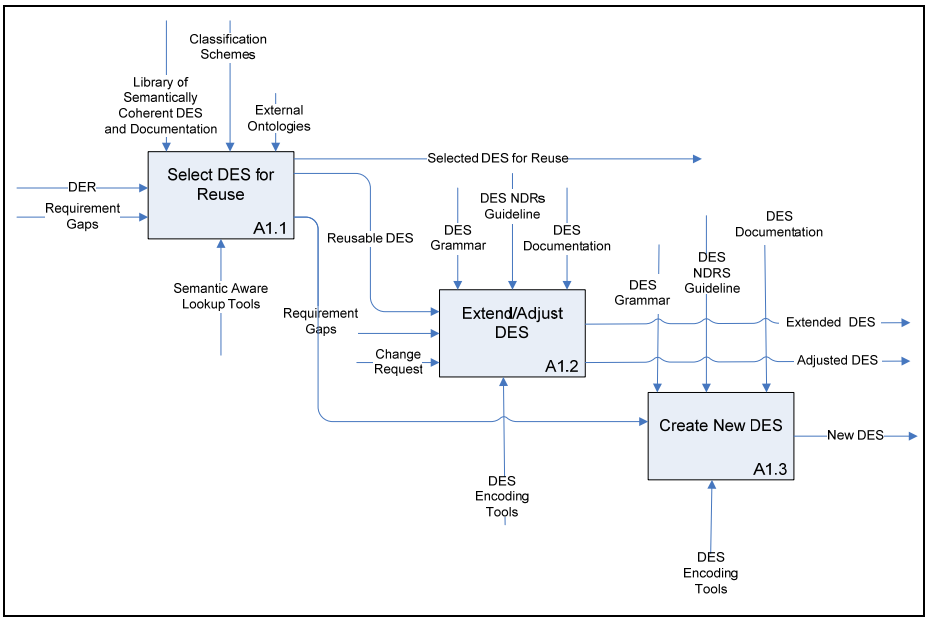

Figure 3: Activity A1 – Discover DES

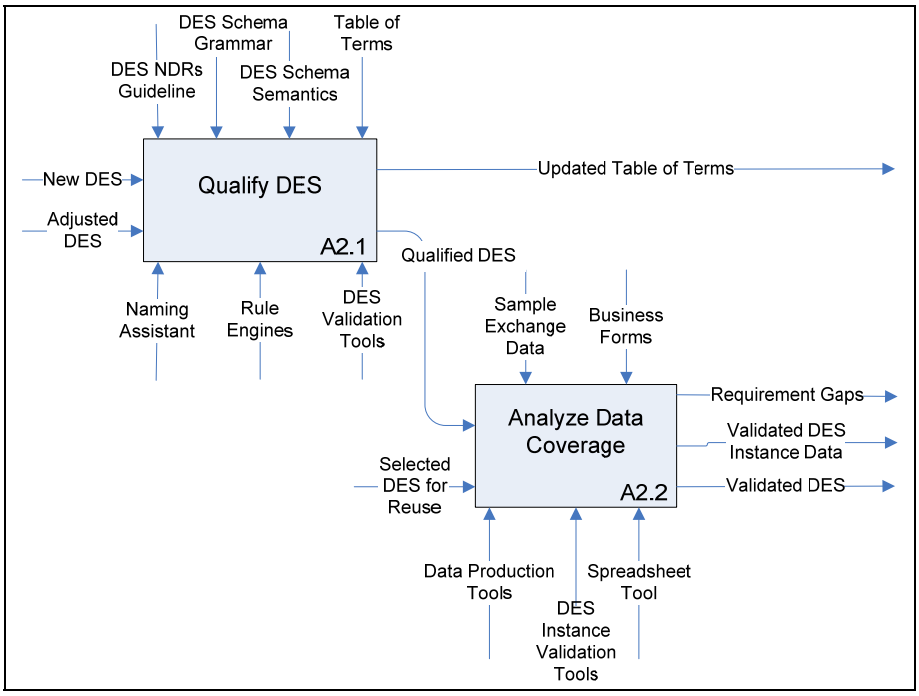

Figure 4: Activity A2- Validate DES

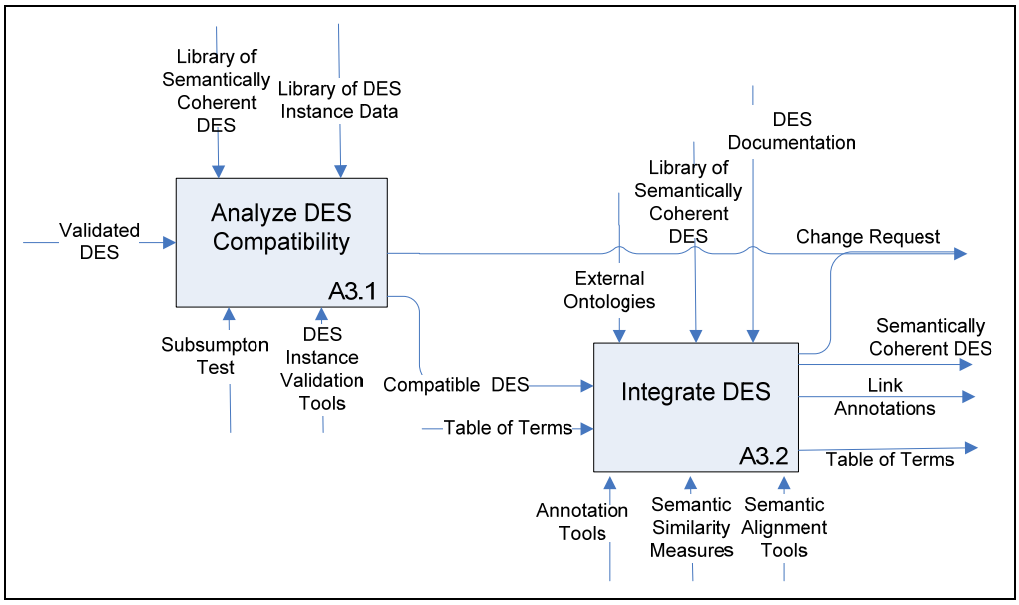

Figure 5: Activity A3 - Maintain Semantic Coherence

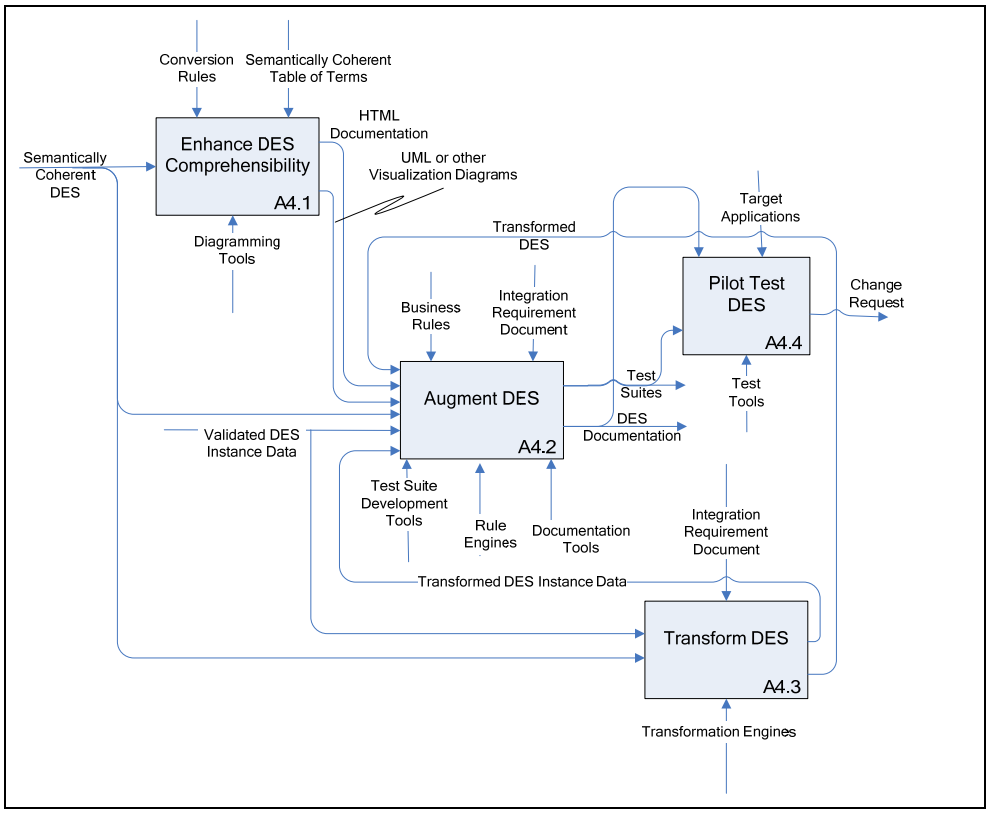

Figure 6: Activity A4 – Pilot DES

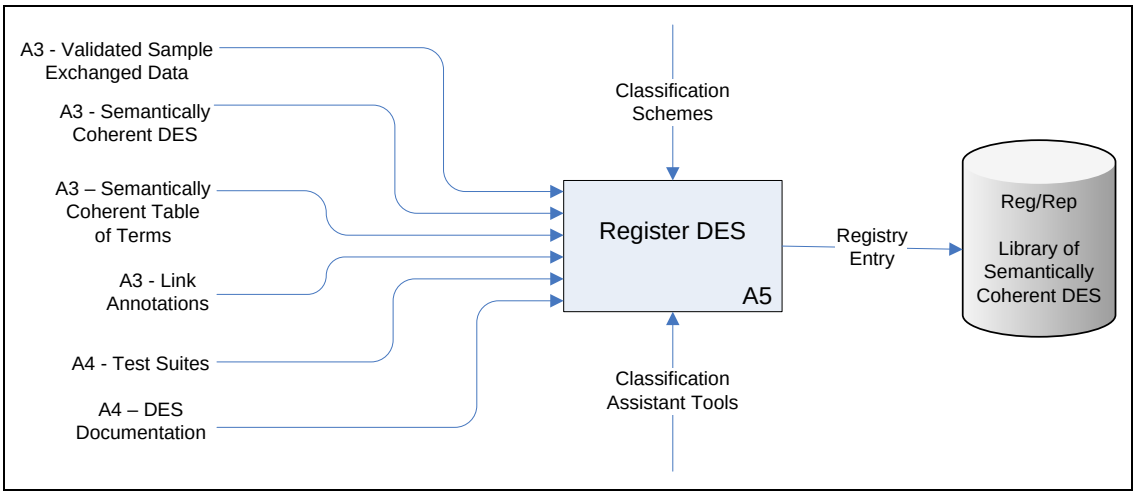

Figure 7: Activity A5 – Register DES