Toward Self-integrating Software Applications for Supply Chain Management

Albert Jones
National Institute of Standards and Technology
100 Nureau Drive, Mail Stop 8260
Gaithersburg, Md 20899-0826
301-975-3554
albert.jones@nist.gov

Nenad Ivezic
National Institute of Standards and Technology
100 Nureau Drive, Mail Stop 8260
Gaithersburg, Md 20899-0826
301-975-3536
nenad.ivezic@nist.gov

Michael Gruninger
National Institute of Standards and Technology
100 Nureau Drive, Mail Stop 8260
Gaithersburg, Md 20899-0826
301-975-6536
michael.gruninger@nist.gov

Abstract

Each paper in this special issue deals with a particular aspect of supply chain management. Nevertheless, they all point to a common need for meaningful and timely information exchange across the supply chain. The Internet and the existing Web technologies have made the exchange of information essentially free and instantaneous. On the other hand, determining the meaning of that information, which we call integration, is still very costly. Millions of dollars and hundreds of man-years have been spent developing and coding interface specifications and software applications to achieve this integration. While this approach has been successful in the past, it is not a viable approach for the future in which the Semantic Web is becoming an important business strategy. In this paper, we discuss a new approach, called self-integration in which software applications are imbedded in an environment that allows them to integrate automatically. We first provide some background information on integration and then focus on our research on semantic querying, semantic mapping, and semantic inferencing.

Key Words: equivalence metric, inference, mappings**,** ontology, self-integration, semantics, supply chain management, translators

Introduction

In the past few years, the world has undergone a digital revolution caused by the evolution of the Internet and the Web. Billions of computers are linked electronically via the Internet; and, the software that drives the Web makes the information on those computers human readable. That's been great for people and it has enabled a rapid growth in electronic commerce -- the buying and selling of goods on the Internet. There is, however, another effort underway to provide a new and different kind of Web. In his recent book, Tim Berners-Lee calls it the "Semantic Web", Berners-Lee (1999). This new Web will enable software applications on one computer to talk directly and intelligently to software applications on any other computer on the Internet. This capability will have a striking impact on people and businesses. One of those businesses is manufacturing.

During the 1980s, manufacturing enterprises around the world tried to emulate the business structure of the most successful economy in the world, Japan. That structure, called *keiretsu*, consisted of tightly integrated, highly closed, wholly Japanese, supplier-manufacturer relationships. It had many advantages, particularly for small suppliers. They had (http://www.businessforum.com/keiretsu.html) "a guaranteed customer, an assured price structure, predictable production and delivery schedules, clear quality and performance standards, and insulation from direct interactions with the marketplace." This captive relationship, however, shielded both the customer and its suppliers from the marketplace. As history has shown, the impact can be positive for a while; but, over time, this shield will weaken the market position of the customer and the capabilities of the suppliers.

After years of observation and emulation, many manufacturers are attempting to build a business structure that will yield the benefits of the keiretsu, but avoid its weaknesses. This structure has many names including virtual enterprise, supply chain, collaborative commerce, and, B2B. In these structures, the suppliers are essentially independent of the customers since each customer receives only a fraction of the total output of a given supplier. This allows customers the freedom to choose the best suppliers and suppliers the ability to maintain their innovative edge. This special issue discusses many techniques related to the optimal design and management of these structures. It also points out the need for a free exchange of meaningful information needed to implement these techniques.

One of the technologies that will enable this free exchange is the Semantic Web. Customers and suppliers are installing a myriad of software applications that they hope will allow them to connect to and conduct business over this new Web. Success is critically dependent upon the ability of these applications to find one another, to establish a dialogue, to exchange information, and to understand that information, more or less automatically. This ability, which we call self-integration, is the focus of this paper. We first give details of the current approaches to integration in several domains. Then we provide some common examples of self-integration. We follow this with a description of

our current and future research efforts in self-integrating systems. We conclude with a summary.

Integration: How It's Done Today

Before describing how self-integration might work in the future, we first summarize how integration is achieved for software applications, FAX machines, and on the Internet.

Manufacturing Software Applications

In a manufacturing context, integration is the timely and meaningful exchange of information between software applications. In the early days of computing, this integration was implemented using point-to-point translators from a specific application on one computer to a specific application on another, possibly, different computer. This required advanced agreement on every detail about the information to be exchanged. As the need to exchange information grew and the types of computers increased, the integration problems exploded. Point-to-point translators became an untenable approach to addressing that explosion.

Starting in the mid 80s, a new approach to integration appeared that is still in use today. That approach is to decompose integration into two parts: communication protocols and interface specifications. Communication protocols govern the physical exchange of bits and bytes between the computers on which the software applications execute. There are many national and international standards for these protocols. These standards can now be implemented in computer hardware and system software that is separate from the manufacturing applications. Interface specifications govern the syntax and semantics of every piece of information exchanged by those applications. For any particular specification, there may be none, one, or many standards. The availability of such standards has potential to reduce the number of required translators from $O(N^2)$ to $O(N)$ -- where $O(\cdot)$ means "on the order of", and N is the number of computers.

Largely, the benefits from the communication standards that govern Internet and network communications have been realized. Bits and bytes can be sent cheaply, quickly, securely, and accurately, from one computer to another anywhere in world. Similar benefits have not accrued from standardizing interface specifications for information exchanges. The development, testing, and implementation cycle for these specifications can take years. Furthermore, since numerous organizations and consortia develop them, multiple, even conflicting, standards arise for the same information. This leads to costly and time-consuming harmonization efforts, which produce longer and more costly cycles.

As manufacturing enterprises implement the previously mentioned business structures, they are making the Internet a critical part of their business strategy. Consequently, the number of information exchanges will increase exponentially. In this environment, the current approach to integration will not work. Businesses cannot wait years for standards to be developed, conflicts to be recognized, and harmonization efforts to be concluded. The key to reducing the time, avoiding these conflicts, and eliminating these efforts lies

in a better understanding and a more formal modeling of the underlying information semantics. As we will see later, this is also the key to self-integration.

FAX machines

FAX, short for facsimile, machines are designed to transmit and receive document images. The original machines, which came on the market in the 1960s, utilized proprietary protocols for analog signaling and image encoding. This meant that each manufacturer's machines could communicate only with other machines made by the same manufacturer. This was, in part, a result of protocol patents -- a manufacturer could not use the same protocols without violating the patent of a competitor, and the competitor had no motivation to license competing products. Since the scanning devices, printing devices, and communication devices all used analog mechanisms, there was also a complex coupling among the technologies used by any FAX manufacturer. This coupling made interoperability nearly impossible.

In the early 1980s, manufacturers incorporated new technologies that could accept any of the commonly used communication protocols. These technologies used "agile" analog circuits that could be tuned by software commands or hardware switches. These circuits examined the properties of the initial signals to determine which protocol the sender wanted to use. While it worked, this agility raised the price of FAX machines considerably at a time when the price of the scanning and printing equipment was falling.

In 1984, the International Telecommunication Union published the first version of v.35 (http://www.itu.int/itudoc/itu-t/rec/obsolete/v/v35_84-fr.html). V.35 was a public modem standard with open licensing that provided for digital-to-analog signal encoding (QPSK) and enabled digital transmission speeds faster than 1-bit-per-Hertz. The intent was to create a wider market for all products by enabling intercommunication of diverse computer systems, terminals, and workstations connected by telephone lines. FAX vendors perceived the value of adopting the new standard and were quick to use v.35 signaling for their systems.

Using v.35 or later modems, the sender and the receiver can negotiate a mutually acceptable transmission speed by sending standard messages until both have consistently acceptable responses. This negotiation automates the process of transmitting messages between machines that are of different ages and capability. In its simplest form, a modern fax machine initiates a connection with an older vintage machine by sending a test signal using its most capable protocol. If it fails to receive a response from the older machine, the sending machine then tries a series of less capable protocols until a response

---

QPSK, quaternary phase-shift keying, is a method of modulating digital signals onto a radio-frequency carrier signal using four phase states to code two digital bits.

is received. Finally, following some handshaking transmissions, the actual facsimile images are transmitted.

Using this negotiation protocol, FAX machines can exchange information timely and accurately; but, they still cannot understand the meaning of that information. This requires a human sender and a human receiver.

Searching and E-commerce

Using the Web, users can transmit messages with one another using a variety of instant-messaging technologies. They can also transmit larger quantities of information to one another using a variety of transfer protocols. The transmissions are done using a collection of communication standards, data representation standards, and security standards. The information is presented on the computers using a collection of textual and imaging standards. The important point is that the principal agents in these activities are people; they generate and they consume the information that is exchanged. The standards simply make the exchange possible.

The Web also enables another kind of information exchange, between people and software applications. In these exchanges, people generate and consume some of the information; the software applications generate and consume the rest. These types of exchanges fall into two main categories: searching and e-commerce.

Search engines are ubiquitous on the Web; their main jobs are to find and return information in response to a query. The returned information can be in a variety of forms that depends on the nature and complexity of the query. Some queries return specific information such as names and telephone numbers. Most queries, however, return a collection of uniform resource locators (URLs) for other Web pages. These URLs may or may not be relevant to the original query, because the search engines understand neither the meaning of terms in the query nor the needs of the user making the query. Consequently, users may have to conduct many searches, repeatedly refining the original query, before the returned URLs match their needs.

Simply stated, e-commerce is buying or selling goods and services over the Internet. It has transformed the Web from a place where people execute transactions to search for or exchange information to a place where people can also execute transactions to conduct business. Each of these transactions involves numerous interactions between software applications and people. Typically, however, the software applications provide raw data, pre-defined forms, or filtered information only; the people interpret the data, fill in the forms, and give meaning to the information. These interactions take place in a very controlled environment and bounded domain, such as such the purchase of a book from an online bookstore. Consequently, the repeated query refinement, typical of general searches, rarely occurs in e-commerce transactions.

Remarks

Clearly, people provide the meaning necessary for the successful exchange of information over the Web -- among themselves and with software applications.

Consequently, people play an essential role in software integration today. As noted above, this will not be the case in the future management of complex supply chains; the integration will need to occur between software applications. We believe that the only viable approach will be to use the Semantic Web as a foundation for self-integration of those software applications. The remainder of this paper describes our current efforts to implement this approach.

Achieving Self-integration in a Supply Chain

If self-integration is to become a reality in the Web-based, supply-chain environment, then we must find a way for supply-chain software applications to convey and understand semantics. Some recent research attempts to provide semantics to information exchanged over the Internet (http://www.ontoknowledge.org/oil/TR/oil.long.html, www.daml.org) and Decker (2000). The most relevant is the joint effort by the Worldwide Web Consortium's (W3C's) Semantic Web Activity, the DARPA Agent Markup Language Project, and the resulting daml+oil specifications (http://www.darpa.mil). The technologies developed in this effort are aimed primarily at making Web pages that programs understand. Nevertheless, they may serve as both a language standard and theoretical foundation for resolving semantic differences.

We have been involved in three research efforts to develop the additional methodologies and mechanisms to demonstrate the feasibility of self- integration: an environment, mappings, and, a testbed. We describe them in more detail in the following sections.

An Environment for Self-Integration

Our strategy has been to start with a simple supply chain scenario involving two software agents: a buyer agent, A1 and a seller agent, A2. We assume initially that A1 and A2 share a common ontology ONT-0. ONT-0 contains the semantics of some very basic terms related to buying and selling, and generic part names associated with the products to be bought and sold. In addition, each agent has its own specialized ontology: ONT-1 gives the semantics of products to order for A1, and ONT-2 gives the semantics for the product catalog for A2. We assume that terms in ONT-1 and ONT-2 are defined on top of ONT-0, and thus can be reduced to terms in ONT-0. In our later research, we will relax these assumptions.

Suppose A1 sends a Request For Quote (RFQ) to A2 for a certain number of a product called "High_Power_Laptop", which is a term defined in ONT-1. Before A2 can determine a quote, it needs to understand what A1 means by this term. We have defined the required ontologies following daml+oil convention in which each term/concept is defined as a class in a frame-like structure. Each ontology forms a hierarchy according to super and sub-class relations and class-instance relation, and is associated with a unique namespace and URL. For example, the term *High_Power_Laptop* in ONT-1 (namespace *ns1*) can be defined as a frame below (omitting the details of daml+oil)

```
<"ns1:High_Power_Laptop"
  <SubClassOf "ns1:Laptop">
```

<ns0:CPU "ns1:High_End_Pentium”>
<ns0:Memory "256M">
<ns0:Disk "SCASI">
... >

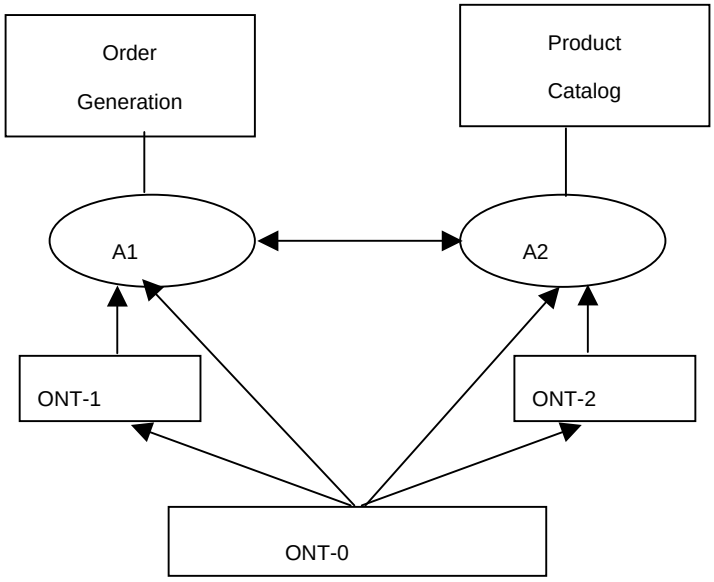

Figure 1. Simple RFQ scenario involving two agents

We use phrase "self-integration environment" to include everything that one application needs to integrate with another. The ontologies described above will be part of that environment. To date, we have identified three processes in that environment: querying, mapping, and inferencing.

Semantic Querying: For A2 to understand the term *High_Power_Laptop*, it needs to understand all the relevant terms used in ONT-1, including slot name and filler name of each slot. The definition of this term can be obtained by querying A1. Since A2 only knows the terms in ONT-0 or ONT-2, it wouldn't understand the term *ns1:Laptop* or the term *ns1:High_End_Pentium*. The process proceeds to query the definitions of these terms, and it may propagate to any number of classes in ONT-1. Vertical query propagation is particularly interesting because some properties that are used to define a class are inherited from its super-classes, and each ontology may have a different hierarchical structure. After successful completion of this step, A2 can build a frame *ns2:High_Power_Laptop* that defines this term by a list of slots/fillers it thinks it understands.

We have identified three major research questions that we are pursuing:

- What methods are needed to structure the many ontologies on the Semantic Web so as to maximize the likelihood of achieving semantic understanding?
- What policies are needed to control query proliferation so that the queries will not continue endlessly after the relevant semantic information is obtained?
- What requirements would this process place on the existing agent communication languages?

Semantic Mapping: In this step A2, attempts to match this frame with those defined in its own ontology, ONT-2. The mapping can be done using logic-based reasoning. For example, as suggested in Stuckenschmidt and Visser (2000), the term *ns2:High_Power_Laptop* matches another term *ns2:X* if it logically follows from the properties of *ns2:High_Power_Laptop*. We may also use some description logic system if *ns2:High_Power_Laptop* can be subsumed by *ns2:X*, taking advantage of their efficient procedures for subsumption and classification operations, Decker (2000). These approaches work best when an exact match can be found in the target ontology, Stuckenschmidt (2000). In a supply-chain environment, it is far more likely that semantically similar terms from two ontologies will agree on many, but not all, properties. Moreover, the number of slots in definitions of similar terms may differ, possibly due to different levels of granularity of their respective hierarchies. Therefore, approximate or partial matching becomes a necessary part of the reasoning process for semantic mapping. Our research in this area is summarized below.

Semantic Inferencing: Whenever two terms do not match exactly, the question of how closely they match must be answered before any mapping can be developed. Put another way, the question is "Is there a quantitative metric that measures the difference in meaning between two information objects?" To answer this question, we turn to the world of physical objects to see how it is answered there.

Measurement comparisons of physical objects have been made for years. The key to successful comparisons is a single approach to the computation and reporting of the measurement uncertainties. That approach is described in the ISO Guide to the Expression of Uncertainty in Measurement, the GUM ISO (1993). The two important concepts in the GUM are measurand and measurement uncertainty. There are two views of these concepts: frequentist and Bayesian.

In a frequentist view, probability is the true, observable value of a physical phenomenon that is subject to randomness. This value and the randomness are inherent properties of the phenomenon. If one could just conduct enough direct observations -- an infinite number is required -- one could eliminate the randomness and find the true value. Consequently, one treats the value of the measurand as an unknown constant and the uncertainty about measurement data as a random variable. The output of a typical frequentist analysis is the estimated true value, an estimated standard deviation, and a confidence interval - all computed using traditional statistical formulas.

In a Bayesian view, probability is a measure of the state of knowledge of a physical phenomenon conditioned on all that is known about that phenomenon. It is not a

property of the phenomenon and it requires no special notion such as randomness. Whenever any relevant information is acquired, our state of knowledge can be updated. One treats the value of the measurand as a random variable and attempts to find a probability distribution that quantifies the current state-of-knowledge. The uncertainty is then a measure of the dispersion of that distribution. The output of a typical Bayesian analysis is a posterior probability distribution, which is computed from the prior distribution using the measured data and Bayes theorem.

The Guide combines the frequentist view and the Bayesian view into a single coherent view of measurement. That view is based on a new concept called the measurement equation, $Y = c_1X_1 + c_2X_2 + \ldots + c_NX_N$, where Y is the measurand and the X_i are the input variables. Each variable for which measurement data is available is analyzed from a frequentist view using sampling theory and the well-known formulas from classical statistics. For all others, a Bayesian view is used. Then the results are combined using the measurement equation and the root-sum-of-squares method.

With information objects, there is no underlying population; there is no way to make repeated observations; there is no randomness; and, there is no sampling distribution. We have one message, and we must determine if we understand that message. Therefore, we believe that the Bayesian view is the correct view for semantic inferencing. We have initiated a research project to determine how to implement that view. That project is addressing three major questions:

- What is the underlying state space?
- How do we assign prior probabilities to the elements in that state space?
- How do we use the posterior probabilities in the inferencing process?

Prior Work on Semantic Mappings

Our prior work on developing semantic mappings between ontologies is based on the Process Specification Language (PSL), Schlenoff et al (1999b). PSL facilitates complete and correct exchange of process information among manufacturing applications. The initial demonstration of integration using PSL was between a commercial scheduling application and a commercial process planning application, Schlenoff et al (1999a). That demonstration was based on the PSL ontology.

The PSL ontology organized into PSL-Core and a set of extensions. PSL-Core is the set of axioms written in the Knowledge Interchange Format (KIF) and using only the nonlogical lexicon of PSL-Core (http://logic.stanford.edu/kif/specification.html). The extensions form a lattice of consistent extensions to PSL-Core.

The purpose of PSL-Core is to axiomatise a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes. This characterization of basic processes makes few assumptions about their nature; consequently, the Core is rather weak in terms of logical expressiveness. In particular,

PSL-Core is not strong enough to provide definitions of the many auxiliary notions that become necessary to describe all intuitions about manufacturing processes.

To supplement the concepts of PSL-Core, the ontology includes a set of extensions that introduce new terminology. A PSL extension provides the logical expressiveness to express information that involves concepts not explicitly specified in PSL-Core.

For each symbol in a non-logical lexicon of an extension of PSL, there exist one or more axioms, written in KIF, that constrain its interpretation. A distinction is drawn between definitional and non-definitional extensions of PSL-Core. A definitional extension is an extension whose non-logical lexicon can be completely defined in terms of PSL-Core. Definitional extensions add no new expressive power to PSL-Core. Non-definitional extensions involve at least one new primitive not contained in PSL-Core.

Translators: Translators, or mappings, for PSL consist of two parts -- a semantic translator and a syntactic translator (see Figure 2).

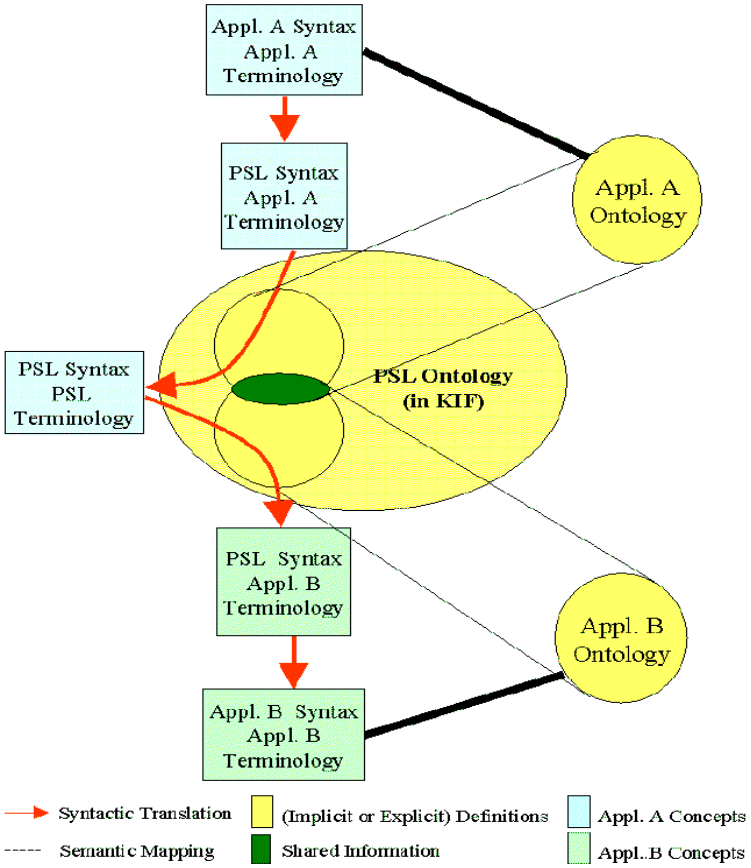

Figure 2. Translation into and out of PSL

The semantic translator maps concepts in the application to concepts in the PSL Ontology by specifying the translation definitions between the terminology of the application ontology and terminology within the corresponding PSL extensions. These translation

definitions have the following form: each relation in the application ontology has a definition using only PSL terminology, and conversely, each relation in the PSL ontology has a definition using only the application's terminology.

For example, the concept *ilcActivity* in the scheduling ontology has following translation definition:

```
(forall (?a)
        (=>    (or     (primitive ?a)
                       (nondet_res_activity ?a))
               (<=>    (activity ?a)
                       (ilcActivity ?a))))
```

That is, the scheduling concept *ilcActivity* is equivalent to the PSL concept *activity* if and only if the activity is primitive or the only non-determinism arises from resource selection.

Syntactic translation of PSL to an application is a mapping from the KIF sentences that specify PSL process descriptions to expressions that satisfy the grammar for the application's process descriptions. The problem of syntactic translation is simplified by the observation that the process descriptions associated with a particular class of activities are not arbitrary sentences. For example, a deterministic activity is defined as one in which all subactivities occur; consequently, any disjunction in the process description for such an activity can only contain ordering literals and not occurrence literals. This approach was exploited in Ciocoiu (1998), where the restricted classes of sentences were used to specify compilation rules that mapped process descriptions in IDEF3 (http://www.idef.com/idef3.html) to process descriptions in PSL.

Challenges: Although the PSL Ontology may faithfully capture the semantics of an application ontology, the converse direction may fail -- in general, there will be concepts that are axiomatized within the PSL Ontology that cannot be axiomatized within the application ontology. For example, most scheduling applications are not able to represent non-deterministic process plans, although these can be defined within the PSL Ontology. Within an interoperability scenario, this leads to the problem of partial translation -- how does the translator support the exchange of process descriptions when information is being lost through translation into a weaker application ontology?

There is also a delicate balance between syntactic translation and semantic translation that arises from the fact that syntactically distinct expressions may be logically equivalent. For example,

```
(forall (?a)
        (=>    (primitive ?a)
               (activity ?a)))
```

is logically equivalent to

```
(forall (?a)
```

```
(or     (not (primitive ?a))
        (activity ?a)))
```

Either canonical forms for expressions must be adopted by the translators, or a first-order theorem-prover must be used during translation to determine whether two expressions are in fact logically equivalent.

Finally, there is the challenge of generating translators automatically. Given the semantic translation definitions, it is possible to generate syntactic translators automatically. Since the process descriptions for each class of activities and resources have a restricted grammar, the semantic translation definitions can be used to match expressions of the application's processes to the grammar of the corresponding process description in PSL.

A vision for generating semantic-mapping definitions automatically is articulated in Ciocoiu (2000, 2001). There, the translation definitions are deduced from the axiomatizations of both the PSL Ontology and the application ontology. Without an explicit axiomatization of the application ontology, however, there is little hope of generating a semantic translator automatically. Without the ontology, there is no specification of the semantics of the application's terminology, and without the semantics, there can be no translator.

An alternative approach is to treat semantic translation not as a mapping between two formal ontologies, but rather as the process of using the standard ontology to attribute semantics to the terminology of the application ontology. In this approach, one may develop computer-assisted engineering tools for explicitly specifying the semantics of an application's ontology through translation definitions.

Supply Chain Integration Testbed

The research to achieve the ultimate goal of self-integration is necessarily multi-disciplinary. To be successful, that research must be grounded in the existing standards, technologies, and applications, yet build on evolving technologies such as the Semantic Web. Moreover, to be applicable, the research must be conducted jointly with all stakeholders including software vendors, manufacturers, standards organizations, and university researchers. With these complex research requirements in mind, we have initiated development of a Supply Chain Integration Testbed (SCIT). SCIT will support four major activities:

- to help the Supply Chain Management (SCM) community better understand the capabilities of Semantic Web and other formal methods for self-integrating systems
- to affect the development of Semantic Web technologies and standards while representing needs of the SCM community
- to facilitate introduction and acceptance of those standards by the SCM community
- to demonstrate the potential of the new generation of self-integrating software applications

Testbed Vision: Figure 3 shows our vision for the testbed. The top box shows the commercial portion of the testbed, which is based on a request from the industrial members of the Open Applications Group (OAG) (http://www.openapplications.org) for a neutral testing facility for commercial, supply-chain-integration and B2B software applications. Here, software vendors will provide access to their products and demonstrate interoperability using existing standards specifications and customer-

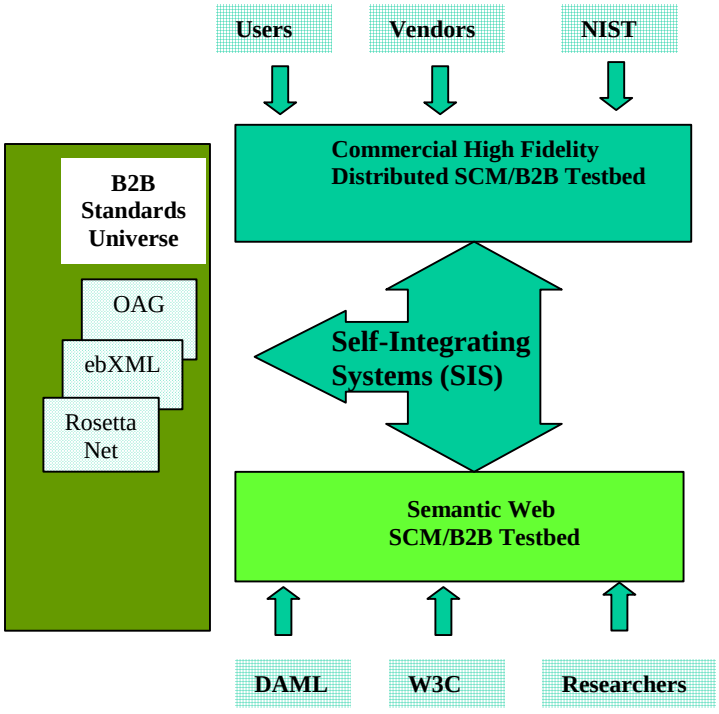

Figure 3. Overall Architecture for SCIT

provided scenarios and testing rules supplied by the National Institute of Standards and Technology (NIST).

The idea for this testing facility came from a recent, one-time event called the Vendor Challenge. This event (http://www.openapplications.org/challenge/index.htm) attracted more than 20 vendors and three major manufacturers -- Boeing, Lucent, and Lockheed Martin. Each manufacturer provided a supply-chain integration scenario that the vendors used to demonstrate interoperability of their software tools based on the OAG interoperability specifications. Unlike the Vendor Challenge, which was a one-time event, this testbed is expected to be persistent. As such, it will have an on-going testing and experimenting capability and will be open for use to a large collection of potential stakeholders.

The left side of Figure 3 identifies the supply chain standards organizations with whom NIST is interacting. At the bottom of Figure 3, is an experimental testbed for Semantic

Web technologies. NIST has started to evaluate these technologies within the self-integrating systems environment. Several projects, which are summarized below, have been initiated. These projects are positioned to link to the existing standards efforts and to provide input to enhancements of the commercial testbed.

Current testbed projects: Three projects have been initiated: Ontologies for Cooperative Product Engineering, Semantic Resolution, and Services Coordination.

The first project investigates next generation of business-to-business and supply-chain frameworks that enable collaborative development of engineered products. The main focus is the potential impact of the *Semantic Web* on these frameworks. Specifically, we will develop an architecture for the vendor-selection task in collaborative product development and discuss how the Semantic Web technologies could affect implementation of the architecture. As part of this effort, we are evaluating the current daml+oil semantic layer from the perspective of that architecture.

The second project, described briefly above, is an on-going research project on resolving semantic differences for multi-agent systems. We are interested to identify guidelines, patterns, processes, and other infrastructure components for the emerging Semantic Web that will ensure that semantic resolution can be addressed. The approach we are taking is an outgrowth of the mapping work described above. First, semantic differences between heterogeneous agents are resolved at runtime and in an incremental fashion. Second, concepts and their classes are represented as frames based on a semantic markup language. Third, the resolution is viewed as an inferencing process and, necessarily, involves approximation and default reasoning.

The last project examines new coordination modeling methods that may effectively support development of supply-chain-management standards by (1) capturing semantics from requirement use cases and (2) synthesizing the formal models from these requirements. The new family of coordination modeling methods based on pragmatic linguistic approaches (e.g., discourse analysis) coupled with formal approaches (e.g., temporal logic) shows promise. Recent advances are particularly exciting as they show the possibility of synthesizing formal specification of coordinated behavior from a collection of specific use cases.

Testbed Operation: The testbed will provide an infrastructure for interaction among manufacturing companies, software vendors, and standards organizations. The manufacturing companies will provide interaction scenarios (as a basis for supply chain integration testing) and requirements describing context for integration testing (e.g., messaging protocol specification). We will capture these scenarios and requirements to drive testbed development and make available a scenario repository. Both the manufacturing companies and software vendors will provide and operate nodes of the distributed testbed as no central authority will exist over the testbed operation. The customers and vendors will come together on an as-needed basis to assess, analyze, measure, and demonstrate on-demand integration of software applications that are used to

operate supply chains. We will provide guidance for coordination of the interactions among the different nodes, develop conformance tests, provide test data, conduct integration tests, analyze these tests, and report results.

The current and future projects in self-integration constitute fundamental activities within the testbed. The context in which these projects get created, depends on the expressed needs of the SCM community (i.e., technology pull) and/or on the perceived opportunity for a particular Semantic Web technology (i.e., technology push). Each project will have partners from the academic, business, and standards communities and will conclude with a demonstration. The testbed will provide a mechanism for transferring the results of research projects into commercial products that address real, supply-chain-integration problems using the Semantic Web.

Summary

This special issue discusses many techniques related to the optimal design and management of supply chains. It also points out the need for a free exchange of meaningful information needed to implement these techniques. One of the technologies that will enable this free exchange is the emerging Semantic Web. Customers and suppliers are installing a myriad of software applications that they hope will allow them to connect to and conduct business over this new Web.

Success is critically dependent upon the ability of these applications to find one another, to establish a dialogue, to exchange information, and to understand that information, more or less automatically. This paper proposed a new approach to understanding information called self-integration and it described some of our current and future research efforts aimed at implementing this approach.

Acknowledgements

The authors wish to express their gratitude to Steven Ray for his general comments on this paper and Ed Barkmeyer for his contributions to the section on FAX machines.

References

Berners-Lee, T. and Fischettei, M., *Weaving the Web The Original Design and Ultimate Destiny of the Wrold Wide Web*, Harper, San Francisco, CA, 1999.

Ciocoiu, M., Translating IDEF3 to PSL, *Computer Science Technical Report CS-TR-3950*, University of Maryland, 1998.

Ciocoiu, M., and Nau D., Ontology-Based Semantics, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Breckenbridge, Colorado, April 12-16, 2000.

Ciocoiu, M., Gruninger M., and Nau, D., Ontologies for Integrating Engineering Applications, *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, 12-22, 2001.

Decker, S. et al, Knowledge Representation on the Web, *Proceedings of the 2000 International Workshop on Description Logic*, Aachen, Germany, August 8-12, 2000.

ISO, *Guide to the Expression of Uncertainty in Measurement*, International Organization for Standardization, Case Postal 56, CH 1211, Geneva 20, Switzerland, 1993.

Schlenoff, C., Libes, D., Ciocoiu, M., and Gruninger, M., Process Specification Language (PSL): Results of the First Pilot Implementation, *Proceedings of International Mechanical Engineering Congress and Exposition*, Nashville, TN, USA, November 14-19, 1999a.

Schlenoff, C., Gruninger, M., and Ciocoiu, M., The Essence of the Process Specification Language, *Transactions of the Society for Computer Simulation*, vol.16 no.4, 204-216, 1999b.

Stuckenschmidt, H., Using OIL for Semantic Information Integration, *Proceedings of the ECAI Workshop on Applications of Ontologies and PSMs*, 14th European Conference on Artificial Intelligence, Berlin, Germany, August, 20-25, 2000.

Stuckenschmidt, H. and Visser, U., Semantic translation based on approximate re-classification, *Proceedings of the Workshop on Semantic Approximation, Granularity and Vagueness*, Seventh International Conference on Principles of Knowledge Representation and Reasoning, Breckenridge, CO, USA, April 12-16, 2000.