J Intell Manuf (2006) 17:681–688
DOI 10.1007/s10845-006-0037-x

Manufacturing interoperability

S. R. Ray · A. T. Jones

Received February 2005: /Accepted: January 2006

Abstract As manufacturing and commerce become ever more global, companies are dependent increasingly upon the efficient and effective sharing of information with their partners, wherever they may be. Leading manufacturers perform this sharing with computers, which must therefore have the required software to encode and decode the associated electronic transmissions. Because no single company can dictate that all its partners use the same software, standards for how the information is represented become critical for error-free transmission and translation. The terms interoperability and integration are frequently used to refer to this error-free transmission and translation. This paper summarizes two projects underway at the National Institute of Standards and Technology in the areas of interoperability testing and integration automation. These projects lay the foundation for at tomorrow's standards, which we believe will rely heavily upon the use of formal logic representations, commonly called ontologies.

Keywords Interoperability · Standards · Ontology · Formal logic

Introduction

Supply chains are the dominant organizational structure in manufacturing today. That structure can be viewed as a global network of suppliers, manufacturers, transporters, retailers, and customers who must share technical and business information seamlessly. This information, previously shared in a variety of ways including paper and telephone conversations, must now be passed electronically and correctly among all the partners in the supply chain. The term "interoperability" is commonly used for this capability.

Disparate corporate and national cultures, a plethora of international and regional standards, and numerous commercial products make this task of sharing information all the more difficult. These facts further underscore the need for a clear and unambiguous, standardsbased, interoperability infrastructure. Such an infrastructure does not exist. The resultant penalty paid by industry has been quantified in a number of studies including a 1999 study commissioned by NIST (1999). This study reported that the U.S. automotive sector alone expends one billion dollars per year to resolve interoperability problems.

There are three principal approaches used to reduce these exorbitant costs. In the first approach, a point-to-point customized solution is developed for each pair of partners. This approach is expensive in the long run because each pair of software systems needs a dedicated solution. When there are, for example, ten partners in the chain, this would require up to $90(10 \times 9)$ interfaces. Moreover, should any system provider release a software upgrade, many of the translators would likely need modification.

In the second approach each original equipment manufacture (OEM) mandates that all partners conform to a particular, usually proprietary solution. This has been the practice, for example, in the automotive sector. While this is a cost-effective solution for the OEM, it causes nightmares for the partners because they are forced to purchase and maintain multiple, redundant systems if they want to do business with several major

S. R. Ray (✉) · A. T. Jones
National Institute of Standards and Technology,
100 Bureau Drive, MS 8260
Gaithersburg, MD 20899-8260, USA
e-mail: ray@nist.gov

OEMs. The current trend of global outsourcing has made this approach practically impossible to implement.

In the third approach neutral, open, published standards form the foundation of the infrastructure. By adopting open standards the combinatorial problems associated with the first approach go away—it is now of order N rather than order N2. The nightmares associated with the second approach because partners can buy any software they want, provided the vendors implement the standards. Furthermore, standards also offer stability in the representation of information, an essential property for long-term data retention. Increasingly, this retention issue is recognized as a costly and critical problem for industries with long product life cycles, such as aerospace.

Early efforts to develop interoperability standards focused on system architectures. One such effort culminated in the Open Systems Architecture for CIM (CIMOSA) (AMICE, 1993).

Later efforts focused on content standards such as ISO 10303 (ISO, 1994), a set of standards for the exchange of product model data. The most widely adopted component, ISO 10303–203 (ISO, 1994b), is already conservatively estimated to be saving the transportationequipment- manufacturing community over $150 million per year in mitigation and avoidance costs. This figure is expected to rise to $700 million by 2010 (NIST, 2002).

But the problem is far from solved. Interoperability standards, like communication standards, come in layers (see Fig. 1). All the layers in this interoperability stack must be implemented correctly for interoperability to be achieved. The greatest challenges remain at the top of this stack.

Is xml the answer?

XML, (the eXtensible Markup Language (http://www.w3.org/XML/), or one of its relatives, appears in almost every layer in this stack. In fact, many standards groups have adopted XML, which has resulted in a tremendously positive impact on the interoperability software systems. XML is a markup language that can be used to tag collections of data with labels.

As part of a standardization activity, communities can agree on the names for these labels. An interoperability problem remains, though, if different people have differing understandings of the meaning of these names. Stated more succinctly, XML standardizes syntax; it was never designed to even capture, much less standardize, semantics. This is not necessarily an obstacle for a tightly knit community that operates within a common context, such as the automotive sector or the financial sector. Within a given sector, the meanings associated with a tag are shared and well understood by all. Serious problems can arise, however, in moving data from one sector to another, such as automotive to financial. Without explicit, rigorous definitions of the meaning of terms, misunderstandings are sure to arise. Humans can resolve such misunderstandings; computers cannot. Consequently, the process of achieving interoperability remains a highly manual process, with computers executing only the most basic steps in this process.

Dealing with this limitation of computers is the focus of two projects at the National Institute of Standards and technology: the Automated Methods for Integrating Systems (AMIS) project, and the B2B (Business-to-business) Interoperability Testbed.

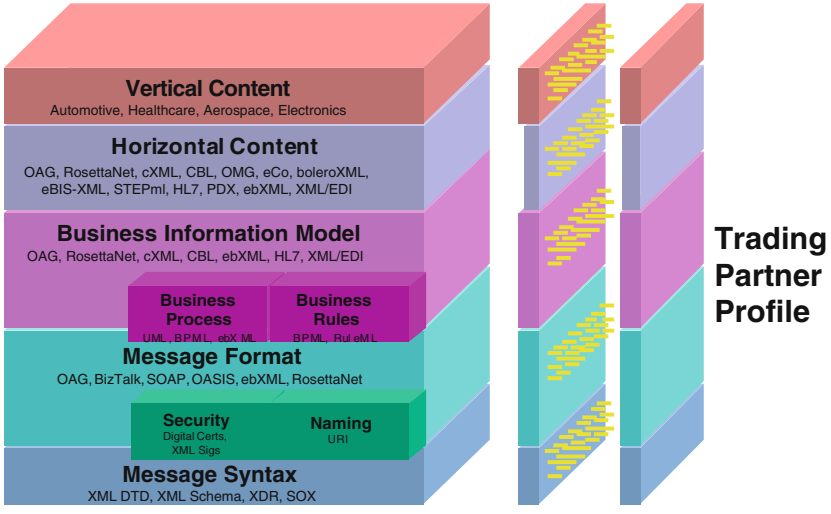

Fig. 1 Interoperability stack

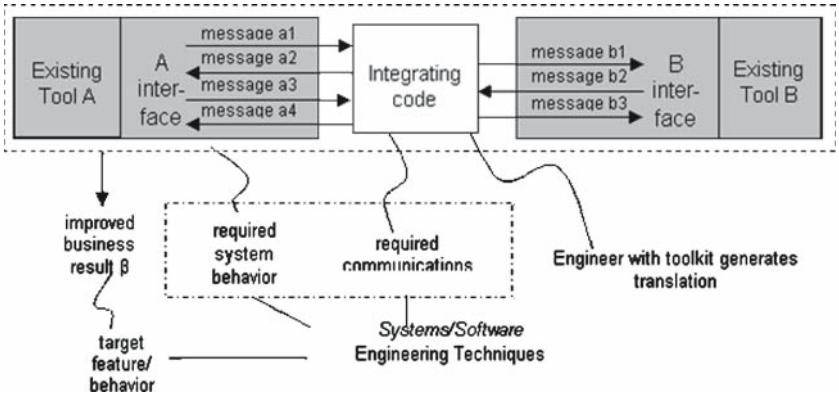

Fig. 2 Generic interoperability problem

Automated methods for integrating systems

From business perspective, an interoperability problem is typically stated as a requirement to produce an improved business result from a set of systems (see Fig. 2) implemented in software tools. Each system that contributes information or functionality to the improved business result exposes interfaces. These interfaces are the communication endpoints by which the systems can be considered to be components that will interact with each other and thereby form a new, integrated system capable of performing new functions. Functions accomplished by such interactions are called joint actions.

As shown in Fig. 2, tool A and tool B each expose interfaces for communication in a number of possible forms. A systems engineer reasons from the desired business result to determine the target feature or behavior the new system must yield—the joint action the two systems are to perform. From the required system behavior and the exposed interfaces, the system engineer determines the required communications between tool A and tool B. To enable these communications, the engineer generates the specifications to translate the relevant messages between the two tools using the interfaces they support and forms that are meaningful. From these specifications, a software developer can generate integrating code (translators) to produce the necessary translations and obtain the new system behavior. To be successful, the systems engineer and the software developer must understand the meaning of the information contained in those specifications and interfaces.

The research question is: can we automate capturing meaning and building translators directly from the interface specifications? To answer this question, the AMIS project has embarked on three main areas of work: a joint action model, semantic mappers, and connector transformations.

Joint action model formulation

The JAM (Joint Action Model) is a requirements model for the intended joint action. The JAM is abstracted from the relevant concepts in the envisioned business process. It specifies the required actions of interactions between the component systems, their conceptual interactions with roles for each, and a shared model of the business entities pertinent to those interactions. In modeling joint action and communication, two distinct levels of abstraction are significant: the conceptual model and the engineering model. These views are important because interoperability solutions are conceived in business/conceptual terms and implemented in engineering terms.

The conceptual model describes concepts, rules, and relationships of the business. The conceptual model of the joint action sees the interacting agents as playing specific roles in the business process, such as buyer and seller. The model characterizes the interaction as flows of information and service requests. At this level, the model is defined using three terms:

- business actions: functions and behaviors that implement the roles of each agent
- business entities: objects that are discussed in the communications and used or modified by the joint action
- transactions: notifications, requests for information, requests for functions or services, and responses

An engineering model of an existing system includes models of the interfaces supporting possible interactions with other systems or agents. The engineering model of the joint action sees interacting agents as software components communicating by one or more mechanisms - file transfer, database access, operation invocation, or queued messaging. For each unit of communication there is a mechanism, and each agent plays

a specific role with respect to that mechanism. And for each unit of communication, there is a message, which is the set of information objects transferred by that communication unit. There is a further level of detail associated with engineering models, which defines detailed communications protocols and binary representations for each data item. Engineering models also contain terms for message types, operations, information objects, and so on. In simple terms, they are the interfaces and communications provided by the software applications.

Semantic mapping

Although conceptual views and engineering views serve different roles, they are not used in isolation. Inter-model relationships between elements from these views link the relationships between an activity or entity expressed in business terms and an engineering means of implementing that activity or representing the entity. We refer to an engineering model linked to concepts in the conceptual model as a local interaction model (LIM).

After the LIMs and the JAM have been produced, the relevant notions in the LIMs must be semantically mapped to their corresponding notions in the JAM (See Fig. 3). The AMIS approach is to build tools once to automate the creation of these maps. This is the difficult research problem—automating the creation, derivation, or extraction of semantic relationships between models. We believe that using formal models, such as ontologies, for the LIMs and JAM is one approach to solving this problem. We discuss this more in Section "Combining AMIS, testing, and ontologies".

Given the links between the JAM and the LIMs, the engineering model for the joint action can be built. This model uses the interfaces identified in the engineering models of the participating tools to achieve the goal described by the JAM.

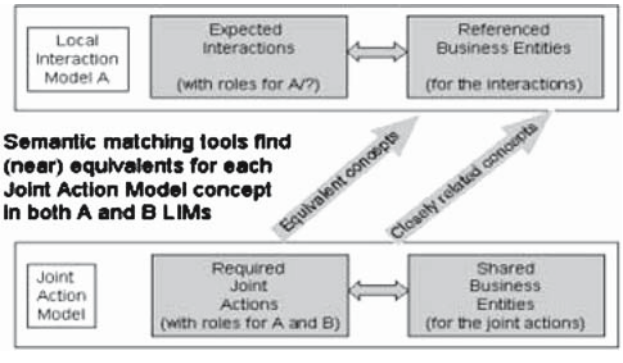

Fig. 3 Conceptual view of semantic mapping

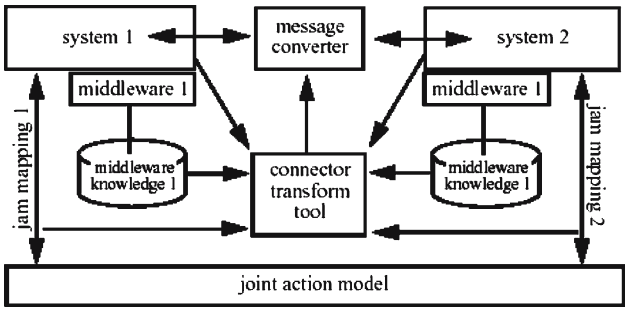

Fig. 4 Role of connector transform

Connector transformation

Given a sufficiently detailed semantic mapping, it is theoretically possible to build a tool that generates translations corresponding to the mappings. This is shown in Fig. 4. To achieve arbitrary transformations of syntax, structure and interactions to the lowest levels of abstraction requires that all the information be formalized. Generation of message converters is then reduced to a search problem: find the composition of available components that can transform the input available into the desired output. There are two significantly different engineering problems here: conversions of messages and message elements, and dealing with differences in the actual communications protocols.

B2B interoperability testbed

The B2B Interoperability Testbed complements the tools and methods emerging from the AMIS project. It provides on-demand interoperability demonstration and testing of B2B information exchange for three stakeholders: software vendors, manufacturing companies, and standards organizations.

Motivation

A number of standards organizations are busy creating these B2B standards and vendors are busy using the latest technologies to implement these standards in their products. Before manufacturers buy these products, they want to be sure that (1) the standards are implemented correctly and (2) they can use the products to do business with their partners around the world. Before vendors can sell their products to manufacturers and their partners, they must show that these products conform to existing standards. This requires substantial demonstrations that involve both users and vendors. Users provide the manufacturing scenarios and test data; vendors

modify their products to implement the required standards, the given scenarios, and the chosen data. Such demonstrations are typically very costly and time consuming because the software infrastructure must be recreated and procedural rules reinvented for each new demonstration. A number of users and vendors suggested that NIST create a persistent environment, tools, and test suites for demonstration and testing. In response, NIST spawned the B2B Interoperability Testbed.

Approach

The testing approach, and the types of tools being developed to implement that approach, are shown in Fig. 5. Currently, the OEM and the supplier represent virtual trading partners from the automotive industry, exchanging messages using different vendor products. NIST and it partners including vendors, users, and researchers from the U.S., Europe and Korea, are developing the tools. These tools include a reflector, process checkers, content checkers, syntax checkers, and grammar checkers.

The Reflector supports both disconnected and connected testing scenarios while allowing for the transactions to be routed to the specified end points, reflected to the originator, and stored in a permanent transaction log.

- The Process Checker performs conformance checking for choreographed transactions between business partners. It currently supports ebXML BPSS and CPA standards but it is being expanded to support Web services. It uses a Web-based, graphical, user interface to monitor the business interactions in real time. It takes the ebXML BPSS and CPA instances as input and produce a graphical presentation of the collaboration as an output. It verifies that each message has the right sender and receiver and that messages come in the right order. It also verifies that any time constraint has not been violated. Should a violation occur, the tool raises a flag indicating that the collaboration has failed.
- The Content Checking tool enables specification and execution of content constraints. It allows standard developers, users, and implementers to precisely specify, extend, and test for conformance with, semantics of a common data dictionary (lexicon). The content tool allows a user to create a profile and specify test cases using a Web-based interface. The test cases are then stored in a customer's repository. Users can select and execute test cases associated with a selected customer by posting an XML document.
- The Syntax Checker is essentially an XML parser that checks the structure of a message against a standard such as the W3C XML Schema. It also verifies that all necessary elements are present and in the right order as specified in the XML Schema instance for that business document.
- The Grammar Checker may be thought of as a superset of the Syntax Checker responsible for enforcing business document structural rules that (1) are defined in some application domain and business context, or (2) can be expressed through grammatical rules of composition and structure generation, but are not 'general' and cannot be easily abstracted from the application context. These rules are not easily expressible in the form of XML Schema instances and require additional expressive capability.

Combining AMIS, testing, and ontologies

We have conducted two experiments that combine ontologies, the philosophy of AMIS, and the B2B testbed. The first focuses on the process specification language and the second focuses on the Web ontology language.

An integration experiment based on PSL

The Process Specification Language (PSL) is a neutral, standard language for defining a process (Gruninger & Menzel, 2003). As an interchange language, PSL is unique due to the formal semantic definitions (the ontology) that underlie the language. PSL's underlying grammar used for PSL is based roughly on the grammar of KIF (Knowledge Interchange Format) (http://logic.stanford.edu/kif/kif.html). Since KIF is a formal language based on first-order logic, it provides the level of rigor necessary to define concepts in the ontology unambiguously. Like KIF, PSL uses BNF (Backus–Naur form) (http://catb.org/~esr/jargon/html/B/BNF.html), to provide a rigorous and precise recursive definition of the class of grammatically correct expressions of the PSL language. Because of these explicit and unambiguous definitions, information exchange can be achieved, sometimes automatically, without relying on hidden assumptions or subjective mappings.

NIST researchers conducted an integration experiment using PSL as an Interlingua between two software applications: process planning and scheduling. Both of these applications had well-defined interfaces, but neither had an explicitly axiomatized ontology associated with those interfaces. They each used a set of terms, but they didn't provide any axioms to constrain the interpretation of their terms. NIST used an in-house tool to generate semantic mappings, which effectively axiomatizes the terms in KIF, using PSL as the mediating ontology. These mappings form the basis for the integration. More

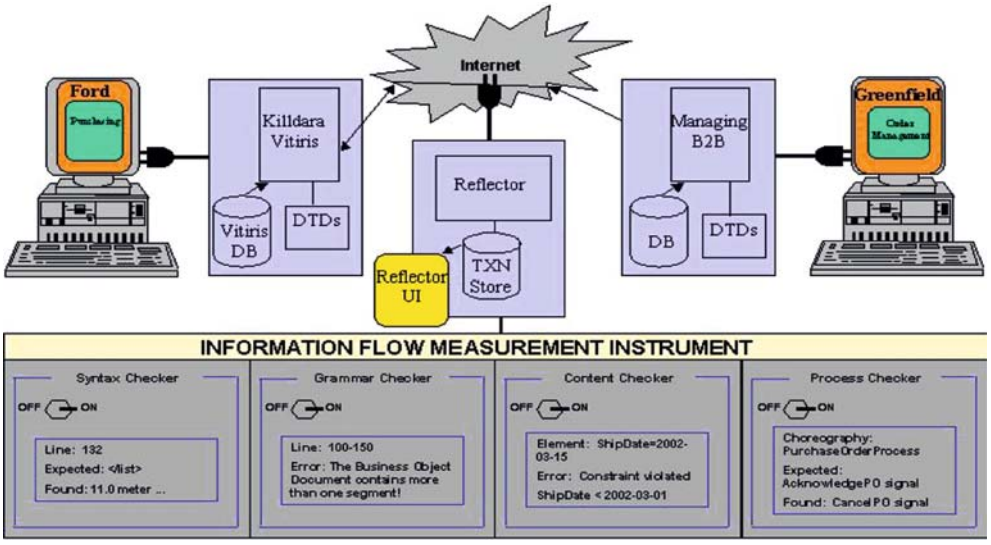

Fig. 5 Schematic of testbed approach

details about this experiment can be found in (Schlenoff, Ciocoiu, Libes, & Gruninger, 1999).

An integration experiment based on OWL

The Web Ontology Language (OWL) is used by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full (http://www.w3.org/2004/OWL/). OWL builds on RDF and RDF Schema (http://www.w3.org/RDF/) and adds more vocabulary for describing properties and classes. Thos properties include relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes. OWL uses both URIs for naming and the description framework for the Web provided by RDF to add four capabilities to ontologies: the ability to be distributed across many systems, scalability to Web needs, compatibility with Web standards for accessibility and internationalization, and openness and extensibility.

NIST researchers conducted a supply-chain integration experiment that used OWL as a basis for creating ontologies that implement the LIM and JAM concepts described above. Supply chain scenario involved two industrial consortia, STAR (http://www.starstandard.org) and AIAG (http://www.aiag.org/), who base their interface models on the same 'horizontal' document standard—the OAGIS Business Object Documents (BODs) (http://www.openapplications.org). These BODs are syntactic formalisms based XML specifications.

Each consortium independently used the inventory-related BODs to define their own document content models and usage rules. These models and rules were encoded interfaces exposed by the software applications that exchange inventory visibility data. These interfaces are the engineering models in the AMIS architecture. Because these models are based on XML schemas, the integration process, which is shown in Fig. 1, is largely a manual process. To automate this process, NIST researchers used the 12-step approach shown in Fig. 6. We now briefly describe this approach. More details can be found in (Ivezic, Anicic, Jones, & Marjanovic, 2005).

1. Apply automated transformation from the OAG XML schema representation into an OAG OWL-based ontology. The outcome of this step is the generalized ontology of OAG that contains only concept descriptions (not definitions).
2. Calculate concept subsumption and check satisfiability for the new OAG ontology. The outcome is a new subsumption hierarchy of the OAG generalized ontology and an indication from the reasoner that the new ontology is satisfiable (i.e., not contradictory).
3. Create regular terminology that requires human designer input. The original STAR or AIAG schema include free text descriptions of the additional document constraints. In this step, for each of the schema, these constraints are used to specify concept definitions (based on the original concept descriptions). The outcome of this step is a regular STAR or AIAG terminology.
4. Similar to Step 2, we check satisfiability of each individual ontology. The outcome of this step is an

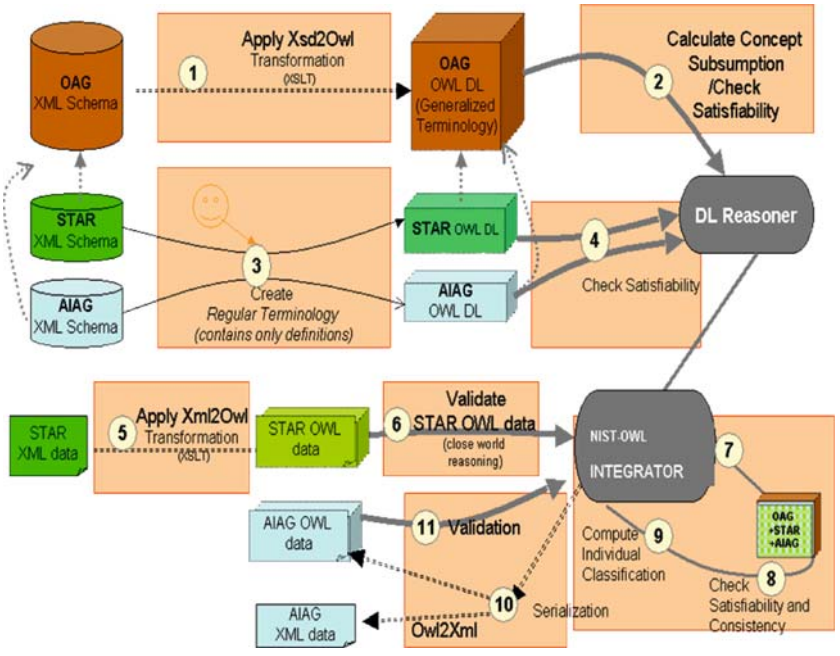

Fig. 6 Interoperability approach using OWL

indication from the reasoner that the individual ontology is satisfiable. In addition, we may choose to check whether the merged ontologies are satisfiable. This is a necessary condition for an individual translation from one to the other ontology. (This step may be omitted here if it is not desired to have this early test of necessary conditions.) In this additional case, the outcome is also an indication from the reasoner whether the merged ontology is satisfiable.

5. Apply automated transformation from STAR XML data to STAR OWL data. This transformation is dependent on the transformation defined in Step 1. The outcome of this step is transformed STAR OWL data that corresponds to the initial XML data.
6. Validation of STAR OWL data includes 'closing the world' for the purposes of checking the consistency of the STAR OWL data. Reasoning about individuals in OWL-DL assumes 'Open World'. We need to 'close the world' and then to check consistency. The outcome of this step, if successful, is an indication from the reasoner that the STAR OWL data are consistent with respect to the STAR ontology.
7. In order to translate from STAR to AIAG OWL data, we need to create a merged ontology from the two individual ones and calculate a new concept subsumption hierarchy. The outcome of this step is the new merged ontology and the new concept hierarchy.
8. Check satisfiability of the merged ontology and consistency of the STAR data with the new ontology. The outcome of this step is an indication from the reasoner that the merged ontology is satisfiable and, similarly, that all STAR OWL data are consistent with the merged ontology.
9. Compute classification of the STAR OWL data in the AIAG ontology. The outcome of this step is assignment of the STAR OWL data to the specific AIAG class(es). At this point we have a result that the specific STAR XML data (instance) may be successfully translated into target AIAG XML data. This, however, doesn't mean that all data may be successfully translated.
10. Apply serialization of OWL data into AIAG OWL data or transformation into AIAG XML data.
11. Apply validation of new AIAG OWL data. The outcome of this step, if successful, is an indication from the reasoner that the AIAG OWL data are consistent with respect to the AIAG ontology.

Summary

In this paper, we discussed the emerging criticality of interoperability in the arena global supply chains. We argued that the current syntax-based approaches cannot keep pace with industrial need in the long term. We proposed a new semantics-based approach that promises to reduce the costs and difficulties involved in achieving interoperability dramatically. We then briefly described efforts at the National Institute of Standards and Technology that will provide the foundation for realizing that approach.

Product disclaimer

Certain commercial software products are identified in this paper. 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.

References

AMICE (1993). *CIMOSA: Ope3n Systems Architecture for CIM*, 2nd revised and extended version. Berlin: Springer Verlag.

Gruninger, M., & Menzel, C. (2003). Process specification language: Principles and applications. *AI Magazine, 24*, 63–74.

ISO (1994). ISO 10303-1:1994, Industrial automation systems and integration—Product data representation and exchange—Part 1: Overview. International Organization for Standardization, Geneva, Switzerland.

ISO (1994b). ISO 10303-1:1994, Industrial automation systems and integration—Product data representation and exchange—Part 203: Application Protocol: Configuration controlled design.

Ivezic, N., Anicic, N., Jones, A., & Marjanovic, Z. (2005). Toward Semantics-based Supply Chain Integration. Proceedings of the IFIP 5.7 Advances in Production Management Conference, Rockville, Maryland, USA.

NIST (1999). Interoperability Cost Analysis of the U.S. Automotive Supply Chain, (Planning Report #99-1), available at http://www.nist.gov/director/prog-ofc/report99-1.pdf.

NIST (2002). Economic Impact Assessment of the International Standard for the Exchange of Product Model Data (STEP) in Transportation Equipment Industries, (Planning Report #02- 5), 2002, available at http://www.nist.gov/director/prog-ofc/report02-5.pdf.

Schlenoff, C., Ciocoiu, M., Libes, D., & Gruninger, M. (1999). Process Specification Language: Results of the First Pilot Implementation. Proceedings of the International Mechanical Engineering Congress and Exposition, Nashville, Tennessee, USA.