A Semantic Web Service Framework to Intelligent Distributed Manufacturing

Boonserm (Serm) Kulvatunyou*, Hyunbo Cho**, Young Jun Son***
* MSI Division, National Institute of Standards & Technology
Gaithersburg, MD 20899, USA
** Department of Industrial Engineering, Pohang University of Science and Technology
Pohang 790-784, Republic of Korea
*** Systems and Industrial Engineering Department, The University of Arizona
Tucson, AZ 85721-0020, USA

Abstract

As markets become unexpectedly turbulent with a shortened product life cycle and a power shift towards buyers, the need for methods to rapidly and cost-effectively develop products, production facilities and supporting software is becoming urgent. The use of a virtual enterprise plays a vital role in surviving turbulent markets. However, its success requires reliable and large-scale interoperation among trading partners via a semantic web of trading partners' services whose properties, capabilities, and interfaces are encoded in an unambiguous as well as computer-understandable form. This paper demonstrates a promising approach to integration and interoperation between a design house and a manufacturer by developing semantic web services for business and engineering transactions. To this end, detailed activity and information flow diagrams are developed, in which the two trading partners exchange messages and documents. The properties and capabilities of the manufacturer sites are defined using DARPA Agent Markup Language (DAML) ontology definition language. The prototype development of semantic webs shows that enterprises can widely interoperate in an unambiguous and autonomous manner; hence, virtual enterprise is realizable at a low cost.

Keywords: Semantic web, Distributed planning and manufacturing, DAML Ontology, Virtual Enterprise, Distributed Manufacturing

1. Introduction

In the past, most companies were able to reduce manufacturing costs and sustain consistent quality by mass production because of stable demands, homogeneous markets, and long product life cycles [11]. More recently, markets have been characterized by unexpectedly turbulent and volatile environments. Typically, product life cycle becomes shortened, marketing powers are shifted towards buyers who require individual customization, and markets become highly diversified and global [15]. While a large and powerful enterprise preyed on weak ones in the past, a speedy enterprise will outlast a slow or negligent one in the future. In order to cope with these fluctuating market situations, the need for methods to rapidly and cost-effectively develop products, production facilities and supporting software including design, process planning, shop floor control, enterprise resource planning, supply chain management, is becoming urgent [3].

In terms of production facilities, agile shop floors can adjust the production process in a timely and optimal manner to respond rapidly to changes in demand or capacity. As far as supporting software is concerned, virtual enterprise concept can be employed that is the cooperation of independently operating enterprises with the aim to design, manufacture and sell specific products, which helps the trading partners concentrate on their special core area of business [11]. Enterprises dynamically form temporary alliances, joining their business in order to share their costs, skills and resources. The virtual enterprises use workflows to automate their supply chain operations and integrate their information systems [2]. To achieve the above goals, a low-cost and flexible electronic data interchange mechanism is necessary. The virtual enterprises can evolve to perform autonomous interoperations across the Internet programmatically using standard Internet protocols and representation formats like Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML).

One of the most important issues in data exchange between two trading partners is interoperability in business transactions, including the semantic interpretations related to business processes, messages, and documents. To achieve interoperability, many companies have formed consortia to develop integration frameworks that provide standard functions enabling businesses to communicate efficiently over the Internet, such as Open Application Group Integration (OAGI), Universal Description, Discovery, and Integration (UDDI), RosettaNet, ebXML, and BizTalk. The

problem with these standards, and many others, is that they are incompatible [22]. Furthermore, even the same set of standards implemented in different organizations may not interoperate. The aforementioned efforts have focused primarily on web service-based partner discovery and some aspects of trade execution. On the other hand, reliable and large-scale interoperation among trading partners can be sustainable by creating a semantic web of each trading partner's service whose properties, capabilities, and interfaces are encoded in an unambiguous, computer-understandable form [4, 9, 13, 19]. In order to facilitate web-accessible semantic definitions, web-based ontology-oriented modeling languages are under development, for example, DAML+OIL (DARPA Agent Markup Language, Ontology Inference Layer).

The objective of the paper is to describe a semantic web of manufacturing services to enable business and engineering collaborations between a design house and a manufacturer. The detailed objectives are as follows: First, an overview of distributed manufacturing is given in the web service framework – this is a functional view of the framework. Second, the ontological definition of resource model and the process capability model on the manufacturer side is proposed using DAML encoding for an efficient and effective distributed manufacturing web service. Third, a prototype manufacturer's service profile is presented to demonstrate an interoperable collaboration using semantic webs.

The rest of the paper is organized as follows. Section 2 provides an overview of enabling technologies for electronic business-to-business integration. An overview of the collaboration framework is presented in Section 3. The ontological definition of manufacturing operations and processes is detailed in Section 4. The service capability semantics of the manufacturer are addressed in Section 5. An example implementation for manufacturing web service discovery is described in Section 6. Finally, the conclusion is provided.

2. Enabling Technologies

The beginning of electronic business-to-business integration dates back to 1948 with the use of Electronic Data Interchange (EDI) [12]. This section gives an overview of related technologies that will enable reliable and large-scale electronic business-to-business integration to support the virtual enterprise realization. These technologies as envisaged in this paper are classified into five major specifications including Business Process Specification, Service Profile, Service Execution

Profile, Business and Engineering Contents, and Communication Specification. Figure 1 illustrates how these specifications fit together, and subsections of this section describe each component in further detail. The technologies described in these sections are currently at different levels of maturity, in that some have been adopted and used as industry standards, some are being reviewed, and some are in the research and development phase.

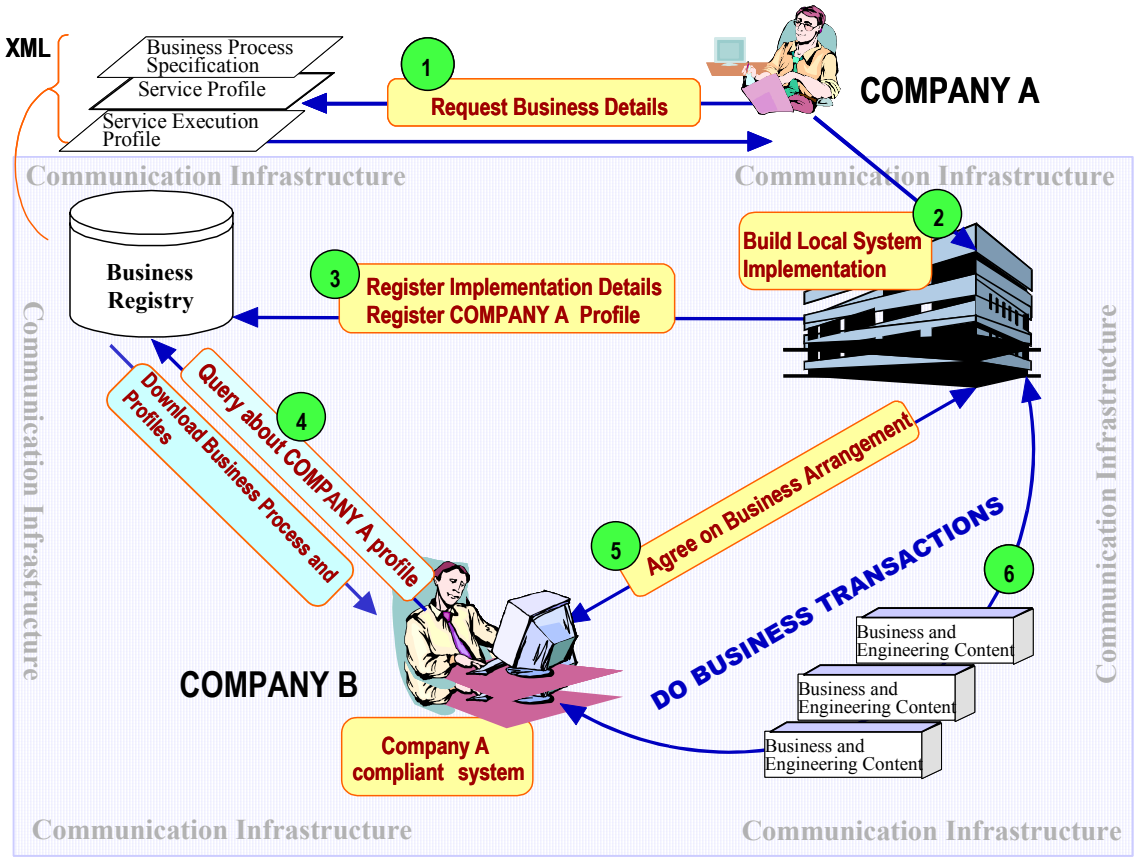

Figure 1: Overview of enabling technologies [modified from 25].

2.1. Business Process Specification (BPS)

The BPS specifies the coordination between partners. It provides a collaboration context for a sequence of business transactions that need to be executed by each participating partner in order to achieve one or more business objectives. Each business transaction typically specifies requirements for the business content (i.e., message), authentication, and confidentiality. Time to perform each transaction is another important business parameter. Each business transaction also implies some

business actions to be taken by the partner. Examples of the business process specification standards are the ebXML Business Process Specification Schema (BPSS) [24] and DAML-S Process Model [6]. RosettaNet also has an embedded business process specification as part of the RosettaNet Partner Interchange Profile [21], though it cannot exist by itself.

2.2. Service Profile

Before the start of business collaboration, the client partner needs to find a partner or partners that may be able to provide the required service. A service profile allows the service provider to advertise its business. Service profiles are stored in open registries, which provide human and computer interfaces to register services and to search for them. A service registry is analogous to the yellow pages. It categorizes the businesses into groups to facilitate the search. Typically, each service profile includes Uniform Resource Locator (URL) pointers to detailed information about the service. The detailed information can include service capability and/or service execution capability. The service execution capability profile specifies how the service might be obtained.

Current technologies that specify service profiles include the ebXML Collaboration Protocol Profile (CPP), DAML-S Profile [5], and more. CPP identifies business capabilities using instances of the Business Process Specification Schema (BPSS) and communication binding capabilities. Similarly, DAML-S Profile uses the DAML-S Process Model and Service Grounding. Other related technologies are the service registry specifications, which include the Universal Description, Discovery, and Integration (UDDI) [26] and the ebXML Registry Information Model. Service registry specifications define storage and retrieval meta-data and interface specifications. Current industry standards for service profiles address only the business and communication capabilities of the service provider, while the manufacturing service capabilities, which are needed in the distributed manufacturing framework, have yet to be addressed.

2.3. Service Execution Profile

The service execution profile is discovered or composed with respect to the service profile at runtime. It is a contract that specifies a business process or service to be executed as well as a communication mechanism to be used. For example, the ebXML Collaboration Protocol Agreement (CPA) specifies the execution profile between two trading partners from the intersection of their

CPPs. Similarly, DAML-S Service Grounding associates a DAML-S Process Model with a communication mechanism. Some service execution profiles indicate the operation to invoke the service and its input/output parameters in association with the communication mechanism (encoding protocol). An example of such an execution profile is the Web Service Description Language (WSDL) [29].

2.4. Business and Engineering Contents

A business document is a piece of information passing between trading partners. For example, the business documents that are involved in a simple request-for-quote (RFQ) business collaboration may include the Get RFQ, the Respond RFQ, and the Quote. In addition to the business information, engineering information may need to be passed to indicate engineering requirements for the product.

There are a number of standard consortia addressing the specifications for business content in different domains. The Open Application Group (OAG) has a number of business document specifications, so called Business Object Document (BOD), in several domains including accounting, procurement, inventory management, automotive retail and more [20]. RosettaNet has its specifications specialized in the electronic component domain. Most of the standard bodies have only focused on the business side of the content. However, the content that delivers engineering requirements for the product is needed for distributed process engineering and manufacturing. Although there is an initial work in the OAG to address the business document for product data management (specifically the engineering change request) using Standard for Exchange of Product Data (STEP, ISO 10303), this paper proposes the integration framework that encompasses process requirements in a distributed manner in association with the product data [17].

2.5. Communication Infrastructure

The communication infrastructure refers to the information technology layer that enables trading partners to communicate in such a secure and reliable manner that legal bindings hold for the collaboration. This includes such communication protocols as message packaging, encoding, security, authentication, and authorization. Other than past developments such as Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Secure Sockets Layer (SSL), Multipurpose

Internet Mail Extensions (MIME), recent developments include the Simple Object Access Protocol (SOAP) and SOAP with attachment, Digital Signature, Public and Private Key Certificate, and Message Encryption. A number of standard bodies combine these technologies into e-business messaging standards, such as the United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT) ebXML Message Service Specification and the RosettaNet Implementation Framework (RNIF).

3. Overview of Distributed Planning in Collaborative Manufacturing

This section describes two functional views of the manufacturing web service. The first view shows three Unified Modeling Language (UML) sequence diagrams describing the activity flows between the design house and the manufacturing partners. The second view illustrates a proposed manufacturing information workflow for the manufacturing web service scenario.

3.1. Activity Flow Overview

The UML sequence diagram convention is used to represent the high-level views of the activity flows within the collaborative planning and manufacturing. The arrows show either self-contained actions or interactions between the design house and the service registry as well as the manufacturing partners. The text above the arrows provides a short description of the activity. Information related to the activity is listed in parentheses. There are four basic steps in the collaboration: 1) service discovery, 2) partner-filtering, 3) distributed process plan construction, and 4) contracting.

In the *service discovery* step, the design house discovers the partners that match the necessary service category (e.g., machine shop) from a web service registry and retrieves the manufacturing capability profiles. The *partner-filtering* step then selects the partners whose manufacturing capabilities match the process requirements. The result of this step is a roster of potential manufacturing partners. In the *distributed process plan construction* step, the design house sends out RFQs and receives back quotes (this is the negotiation process) from the potential partners as it searches for the best way to distribute the manufacturing of the product. Figure 2 shows the case where collaboration succeeds without design and process plan revision, while Figure 3 and Figure 4 show the cases where there are infeasibilities and revision of the design and/or process plan is

necessary. After the revision, the collaboration process may loop back to the discovery step or the filtering step depending on the degree of changes in the revision. After the distributed process plan is completely constructed, the design house can start business processes to subcontract each of the selected partners. This last step is beyond the scope of the paper.

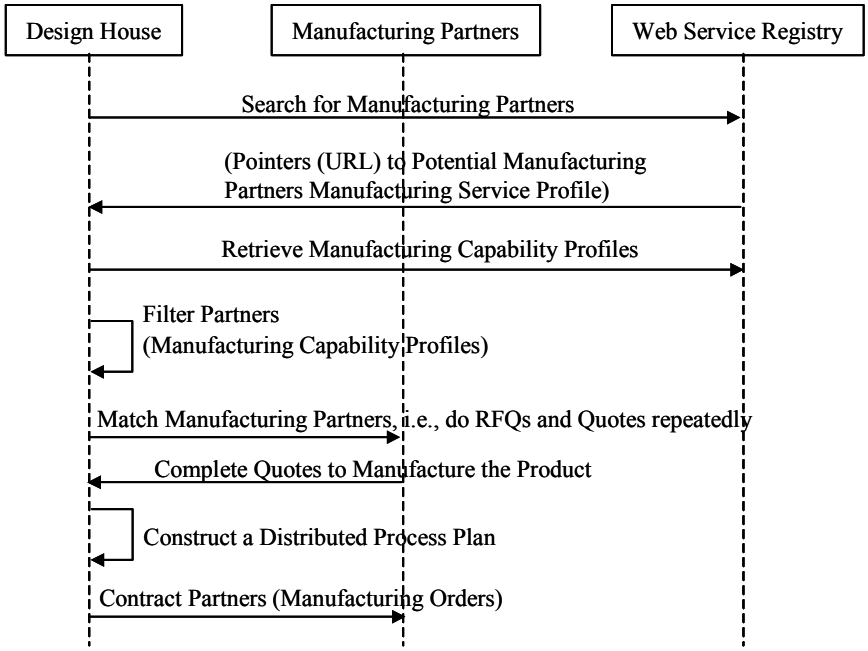

Figure 2: Activity flows without infeasibility in the collaboration.

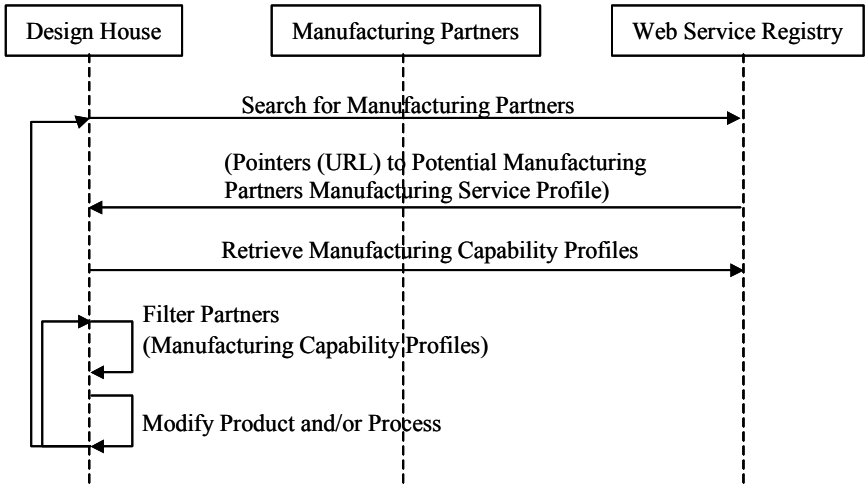

Figure 3: Activity flows with service discovery infeasibility.

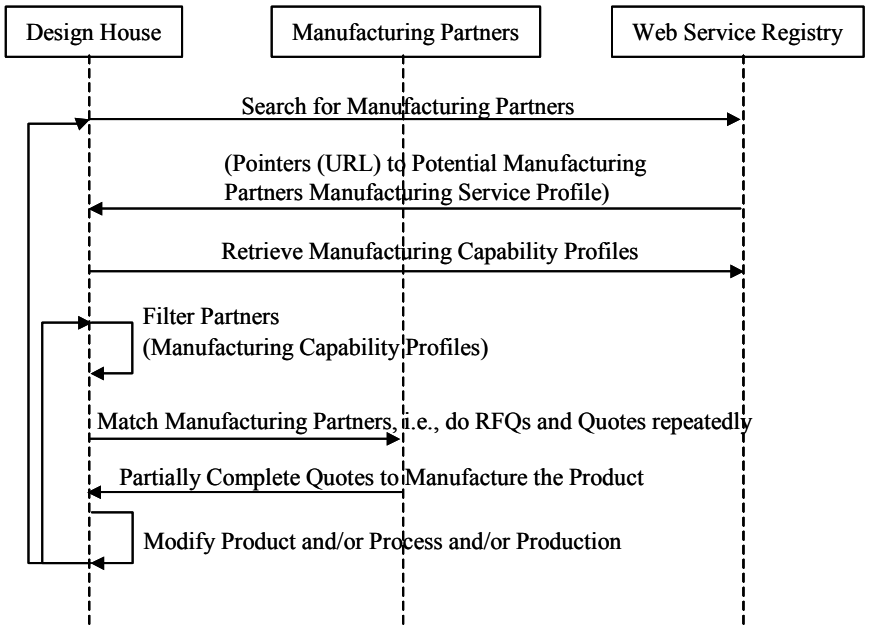

Figure 4: Activity flows with partner matching infeasibility.

3.2. Information Flow Overview

The evolution of the proposed collaboration process in the information centric view is illustrated in Figure 5. The designer designs the part and then prepares the process centric data, so called a ***Resource-Independent Process Plan*** (RIPP), as defined in Definition 1. Its exemplary graph is depicted in

Figure 6a. The RIPP is represented in a two-level process-plan graph. The upper level graph is called an Operation Level Graph (OLG) and the lower level graph is called a Process Level Graph (PLG). The OLG is an AND/OR graph in which each node describes a type of operation, the associated equipment and work-holding capability requirements, and a pointer to the associated PLG. An operation is viewed as an aggregation of processes where the product specification necessitates that they are executed without refixturing. The PLG is an *augmented* AND/OR graph in which each node contains process capability requirements such as type of process, accuracy, and associated geometry, which are derived from the product data. In the AND/OR graph, AND junctions facilitate sequence relationships among nodes. All operations for nodes (or paths) emanating from an AND junction must be done, but they may be done in any order. OR junctions

represent alternative sequences and parallel actions, which imply that only one node (or path) must be done among all of those emanating from the OR junction.

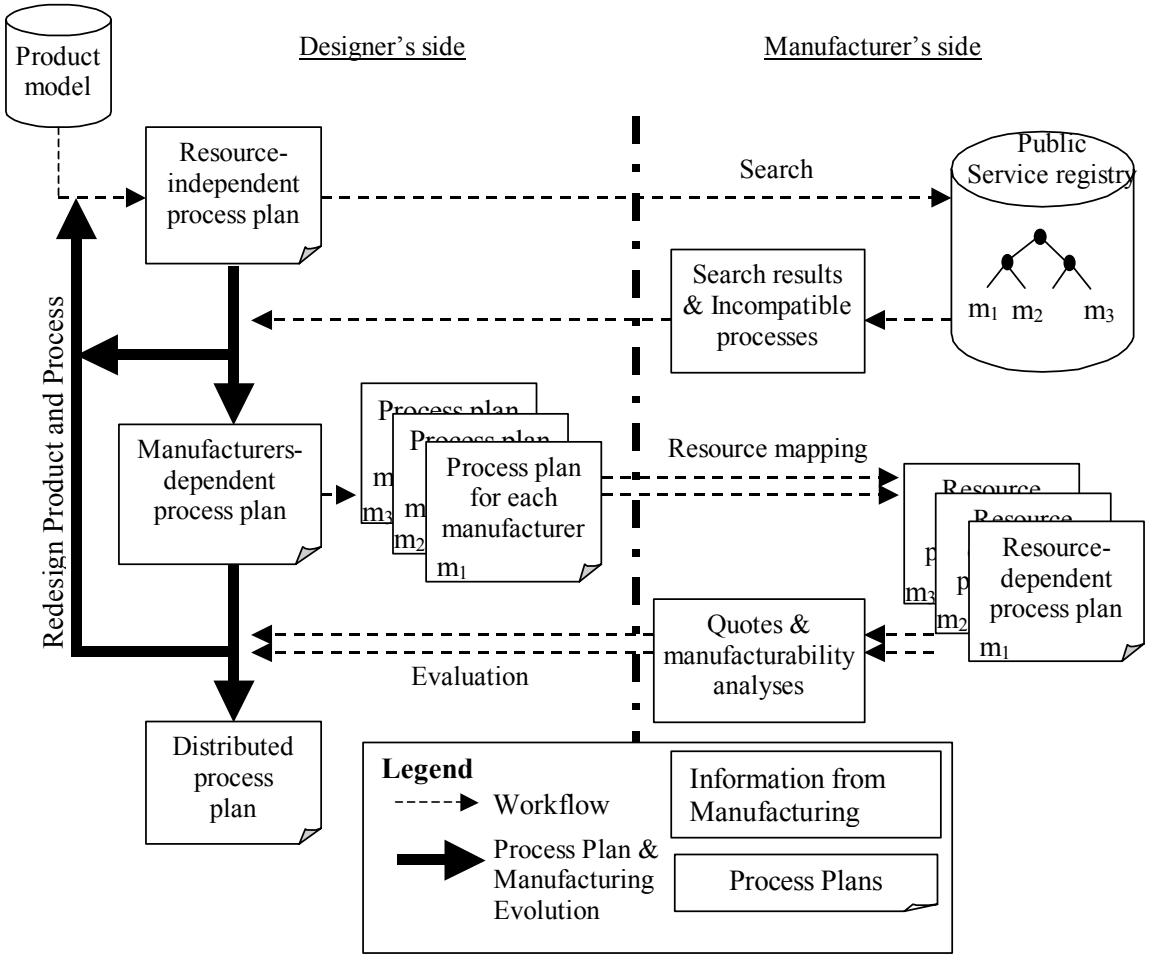

Figure 5: Process information workflow for collaborative planning and manufacturing

Definition 1. A ***resource-independent process plan*** (**RIPP**) $G_i = (V_i, E_i)$ is an AND/OR graph where V_i is a finite, non-empty set whose members are nodes containing process level graphs and E_i is a finite, non-empty set whose members represent precedence among nodes.

After finishing preparing the RIPP for the designed part, the design house searches for one or more relevant manufacturers that possess the operation specified in each node from the manufacturing web service registry. The registry contains a number of manufacturers categorized based on particular processes (typically only a high-level classification) that they can perform. The search returns meta-data for manufacturing web services, which consist of manufacturer names,

pointers (e.g., URL) to the manufacturing capability profiles, and the Internet addresses where each service can be invoked. It may return no entry for some processes in the RIPP; in which case, a new revision of design and/or the process plan will need to be created.

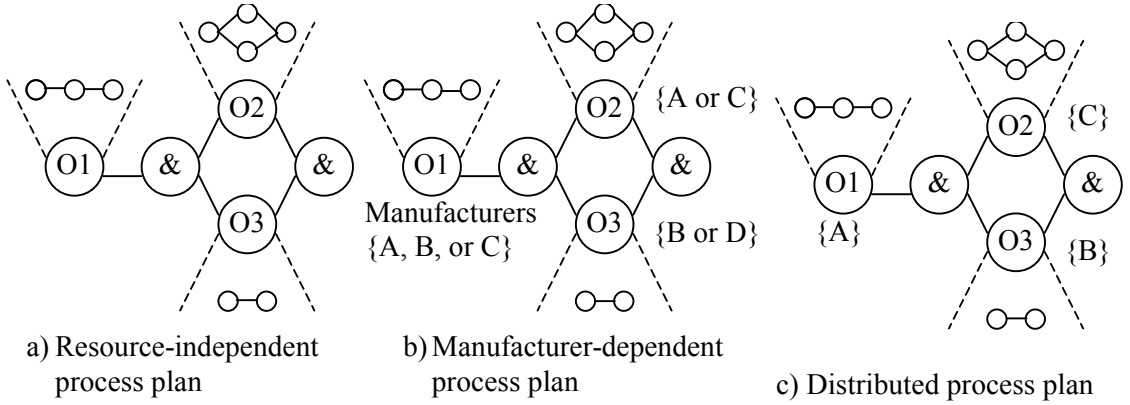

Figure 6: Process plan evolution illustration.

Once manufacturers are identified for all of the operations specified in the RIPP, the RIPP is transformed into a manufacturer-dependent process plan (MDPP) as defined in the Definition 2. Its exemplary graph is depicted in

Figure 6b. As the number of manufacturers returned from the search increases, the complexity of the MDPP increases. Two manufacturers who can perform an identical operation can be represented as two alternatives attached to each node for that operation. The process plan for each manufacturer, which is represented as a node in the MDPP, is conveyed to the related manufacturer with a request for quote.

Definition 2. A ***manufacturer-dependent process plan*** (**MDPP**) $G_m = (V_m, E_m)$ is a resource-independent process plan with alternative manufacturers attached to each node. The set of alternative manufacturers that can perform the operation defined in each node are identified.

The manufacturer maps its own resources to the conveyed process plan, which results in a Resource-Dependent Process Plan (RDPP) as defined in Definition 3. In order to win the request for quote issued from the designer, the manufacturer minimizes the manufacturing cost and time subject to the constraints of detailed surface finishes, tolerances, etc. Each manufacturer returns detailed quotes and/or a list of the problems that occur when mapping resources.

Definition 3. A ***resource-dependent process plan*** (**RDPP**) $G_r = (V_r, E_r)$ is an AND/OR graph where V_r is a finite, non-empty set, in which each member is a node containing a machining feature attached to a set of resources for creating it, and E_r is a finite, non-empty set whose members represent precedence among nodes.

Once the design house has received and evaluated manufacturers' quotes a Distribution Process Plan (DPP) is generated, in which a single manufacturer is selected for each node in the manufacturer-dependent process plan. The DPP is defined in Definition 4. Its exemplary graph is depicted in

Figure 6c. In the evaluation stage, the designer selects the best plan based on the quotes; otherwise, it re-plans the part if there is no feasible plan. When evaluating the quotes, the design house must consider material transportation costs to minimize overall production costs. It is noted that if the design house wants to request for quotes for multiple parts, the evaluation strategies might be different.

Definition 4. A ***distribution process plan*** (**DPP**) $G_d = (V_d, E_d)$ is exactly the same as the manufacturer-dependent process plan with an exception that each node is attached with a single manufacturer.

4. Ontological Definition of Manufacturing Operations and Processes

4.1. Approaches

Universal Description, Discovery, and Integration (UDDI) registry allows the business to register their information including business/service type and geographical location. UDDI suggests the use of business/service type coding standards United Nations Standard Products and Services Code (UN/SPSC) [27], North American Industry Classification System (NAICS) [28], and Data Universal Numbering System (DUNS) number [8]. These standards retain very limited semantic description about the service. For example, a machine shop may use the UN/SPSC code '73.12.16.1' to indicate a 'Metal Cutting Service' and may add the code '73.12.15.6' to indicate a pre-finishing metal processing service and/or '73.12.15.7' to indicate a finishing metal processing service. These numbers have their classification limited to four levels as indicated by the four dotted-separation fields. Even worse, there is no way that the client of the service can know what

size of work piece that the machine shop can accept, whether it is just a small/capability-specific job shop (e.g., deep hole drilling, non-traditional machining), the machine shop's quality assurance compliance, etc. If the client of the service knows these pieces of information in advance, service execution is more efficient because only a small set of manufacturers will be queried during the service execution process. A richer service profile also benefits the distributed process plan construction in the collaborative manufacturing scenario, since each discovered manufacturing partner can be matched with each node in the resource independent process plan.

Our methodology is to extend an industry standard registry such as UDDI or ebXML business registry with a semantic markup of manufacturing capability profile. A registry entry typically consists of a product and service type identification, the geographical location of the service, and a pointer to the service. The pointer to the service can be one or more entries of contact information such as an Internet address, an email address, a personal phone number, or a fax number. Manufacturing partners are first discovered through the matching of product and service type. Our methodology then assumes that each registered manufacturing partner has one of the pointers directed to the service point, an Internet address, where a markup of manufacturing capability can be retrieved. The client of the service (design house) collects these capability profiles and uses its own matching algorithm to filter the manufacturing partners retrieved from the standard registry. It should be noted that another approach could be taken -- to create a manufacturing-oriented capability profile registry to guide each manufacturer through creating its capability profile. A public interface is then provided to filter the manufacturing partners. To facilitate either approach, domain ontology is required to instantiate a manufacturing capability profile.

The Resource Description Framework (RDF) is a possible candidate language to represent the web-based domain ontology. The RDF model and syntax is based on a triple representation consisting of *subject*, *verb*, and *object*. The *subject* and *object* can be any resource, which is limited to any intangible web content either part or whole such as an image, an HTML document, or an XML element. The *verb* can be interpreted as a predicate, a property, a relation between resources, or a restriction of the relation. The *verb* itself can also be a resource. Resources are identified by using the Uniform Resource Identifiers (URIs) [1]. The triple representation can be represented as a labeled directed graph as shown in Figure 7, where the solid line points from a *subject* to an *object* with a label of a *verb* and the dotted line indicates an instance relationship. In fact, the dotted line is a solid line labeled with *type*. The semantic definitions of those properties (e.g. 'domain','range')

and the relationship between those properties and other resources (e.g. 'subClassOf', 'subPropertyOf') are specified in the RDF schema. The *subClassOf* implies the parent/child relationship of any two classes, i.e., an instance (resource) of a child class is also an instance (resource) of the parent class. The *subPropertyOf* implies the parent/child relationship between two properties. The *domain* and *range* properties constrain the relationship between resources and properties. The *domain* constrains the subject of the property, while the *range* constrains the object of the property. However, RDF is designed only for markup of web content meta-data; therefore, it has limited semantics. The DARPA Agent Markup Language (DAML) initiative, which is based on RDF, defines additional semantics for web-based knowledge representation and sharing.

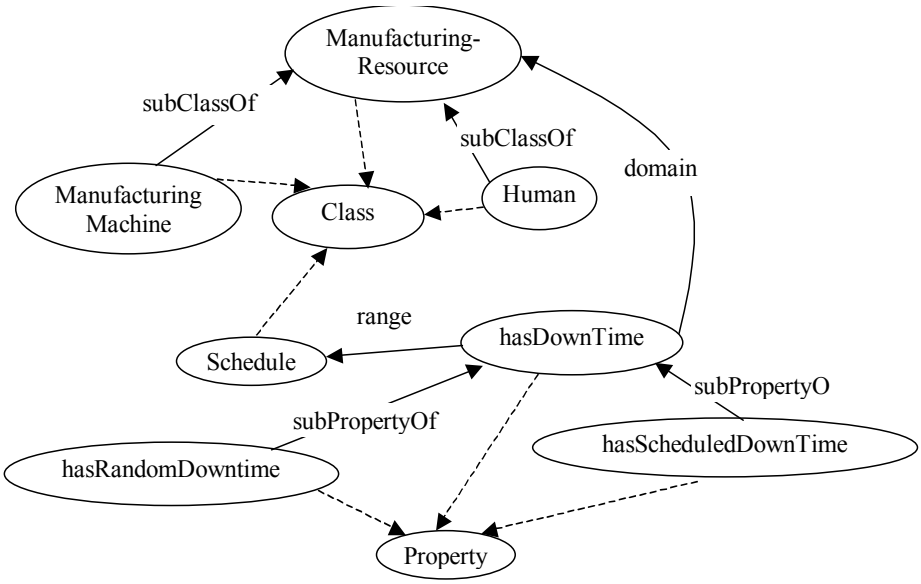

Figure 7:Graphical Representation of an example RDF model.

DAML syntax and model theoretic extends RDF with the description logics [18]. DAML enables the creation of ontologies for any domain and the instantiation of these ontologies in the description of specific Web sites. After the first release of DAML-ONT, DAML merged with the European effort, Ontology Inference Layer (OIL) for a new version, DAML+OIL. The capabilities of DAML lie not only in its Web-based knowledge-sharing scheme (i.e., Semantic Web), but also in their ability to represent logical relationships between data. This allows developers to annotate the web content and enrich the web with distributed relational meta-data in order to enable a machine-understandable web. DAML is employed to represent the manufacturing service capability profile.

4.2. Ontological Definition of Manufacturing Operations

The ontology will focus on the machining domain; however, the construct should also provide abstract layers that are the basis for other manufacturing domains. Using EXPRESS-G format, Figure 8 illustrates the manufacturing domain ontology at the operation level. EXPRESS is an ISO standard as a formal language to describe language-neutral information model [14]. EXPRESS-G is a graphical version of the text-based EXPRESS. Figure 9 is its partial DAML encoding of the *MaterialRemovalOperation* class. Briefly, the ontology states that any operation can be classified as a business operation, transportation operation, or manufacturing operation. The ontology focuses on the manufacturing operation by further classifying it as shown in Figure 8.

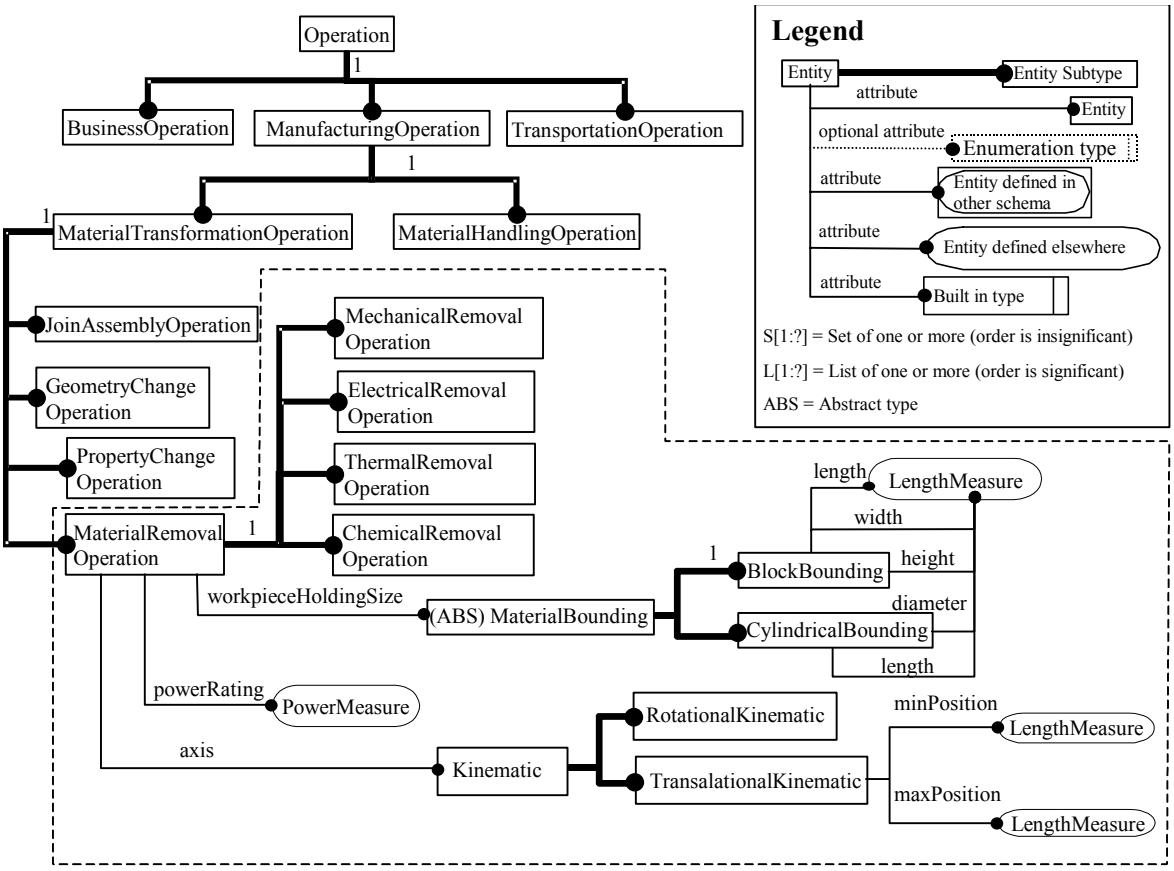

Figure 8: Manufacturing domain ontology

```
<!-- This is stored at "http://msi.postech.ac.kr/OperationOntology.daml" -->
<!-- Low level concepts such as measurement elements are assumed to be defined in
the "http://msi.postech.ac.kr/SupportOntology.daml" -->
<rdf:RDF
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:support="http://msi.postech.ac.kr/SupportOntology#"
    xmlns     ="http://www.daml.org/2001/03/daml+oil#">
<Class rdf:ID="MaterialRemovalOperation">
    <rdfs:subClassOf rdf:resource="#MaterialTransformationOperation"/>
    <disjointUnionOf ParseType="collection">
        <Class rdf:about="#MechanicalRemovalOperation"/>
        <Class rdf:about="#ElectricalRemovalOperation"/>
        <Class rdf:about="#ThermalRemovalOperation"/>
        <Class rdf:about="#ChemicalRemovalOperation"/>
    </disjointUnionOf>
</Class>
<Class rdf:ID="MechanicalRemovalOperation">
    <rdfs:subClassOf rdf:resource="#MaterialRemovalOperation"/>
</Class>
<!-- Other MaterialRemovalOperation subclasses could be added -->
<ObjectProperty rdf:ID="axis">
    <rdfs:domain rdf:resource="#MaterialRemovalOperation
    <rdfs:range rdf:resource="#Kinematic"/>
</ObjectProperty>
<ObjectProperty rdf:ID="powerRating">
    <rdfs:domain rdf:resource="#MaterialRemovalOperation"/>
    <rdfs:range rdf:resource="support:#PowerMeasure"/>
</ObjectProperty>
<Class rdf:ID="Kinematic">
    <disjointUnionOf ParseType="collection">
        <Class rdf:about="#RotationalKinematic"/>
        <Class rdf:about="#TranslationalKinematic"/>
    </disjointUnionOf>
</Class>
<Class rdf:ID="TranslationalKinematic">
    <rdfs:subClassOf rdf:resource="#Kinematic"/>
    <rdfs:subClassOf>
        <Restriction daml:cardinality="1">
            <onProperty rdf:resource="#minPosition"/>
            <onProperty rdf:resource="#maxPosition"/>
        </Restriction>
    </rdfs:subClassOf>
</Class>
<Class rdf:ID="RotationalKinematic">
    <rdfs:subClassOf rdf:resource="#Kinematic"/>
</Class>
<ObjectProperty rdf:ID="minPosition">
    <rdfs:domain rdf:resource="#TranslationalKinematic"/>
    <rdfs:range rdf:resource="support:#LengthMeasure"/>
</ObjectProperty>
<ObjectProperty rdf:ID="maxPosition">
    <rdfs:domain rdf:resource="#TranslationalKinematic"/>
    <rdfs:range rdf:resource="support:#LengthMeasure"/>
</ObjectProperty>
<!-- workpieceHoldingSize property could be added -->
</rdf:RDF>
```

Figure 9: Partial DAML encoding of the material removal operation ontology.

In Figure 9, the DAML semantics, *disjointUnionOf*, states that the members of each of the four material removal operation subtypes are not members of one another. A number of properties are associated with the material removal operation (and its subtypes). For example, one statement states that the *axis* property is characterized by the *Kinematic* class. It should be noted that in high-level ontology some properties and their relationships to classes (domain and range) are identified without cardinality restrictions. These properties have context sensitive effects to the class semantics. For instance, the ontology does not specify that the *axis* property is required for the *MaterialRemovalOperation* class.

4.3. Ontological Definitions of Manufacturing Processes

Similar to the operation ontology, Figure 10 partially illustrates the upper level process ontology with details shown for the hole making process. It should be noted that each operation is viewed as an aggregate of processes. This portion of the ontology states that the Manufacturing Process can be classified into two disjoint sets called the Material Transformation Process and the Material Transportation Process. The Material Removal Process is a subtype of the Material Transformation Process. Furthermore, the Hole Making Process, Roughing Process, and Face Making Process are subtypes of the Material Removal Process. It should be noted once again that there are no cardinality requirements specified for the Hole Making Process at this level. Only the properties and valid values are identified.

In the next section, DAML-S is extended to describe not only business process capability, but also the manufacturing capability. The manufacturing capability represents the real business capability of the service. The domain ontology illustrated in this section is utilized for the extension. Note that this ontology definition is assumed to be stored at "http://msi.postech.ac.kr/ProcessOntology.daml".

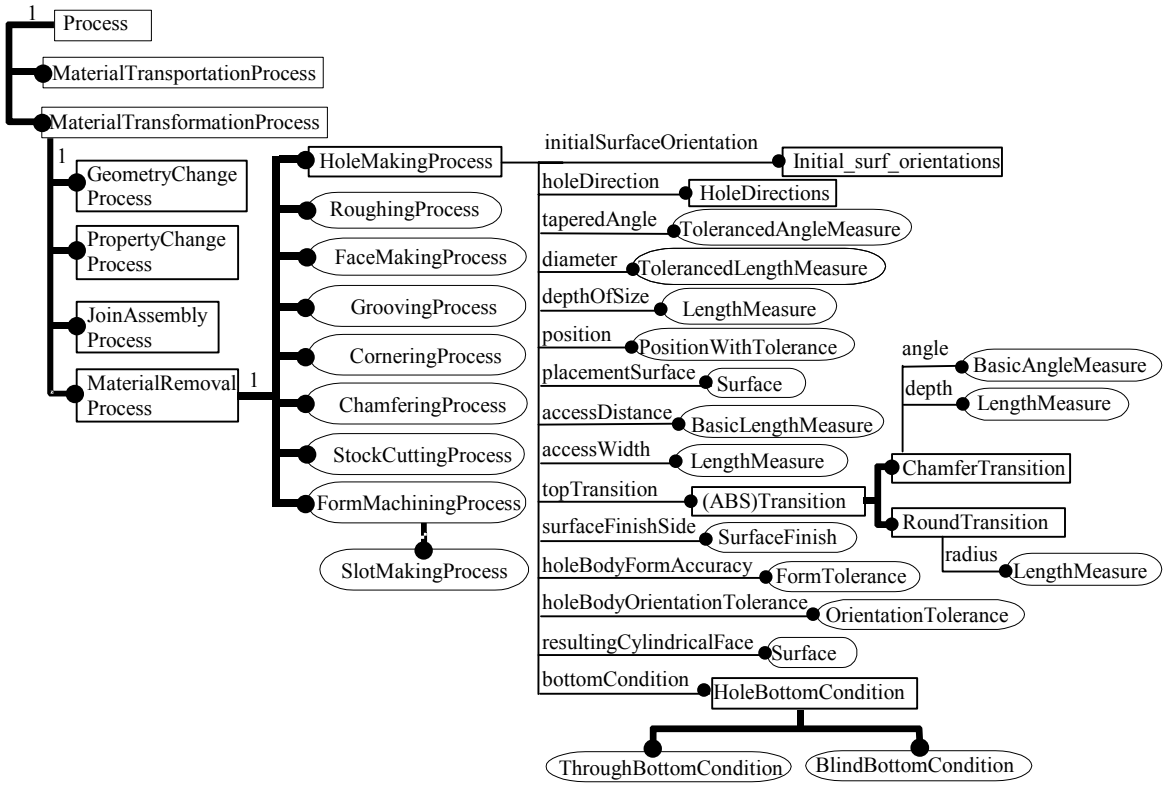

Figure 10: Upper level process ontology and extended hole making process ontology.

5. Web Service Markup for Manufacturing Capability

5.1. Approaches

The manufacturing service capability profile is different from the Service Profile and the Service Execution Profile used for the business transaction and information technology capabilities. Industry standard registries such as UDDI and ebXML list very limited partner information. Due to our need for more partner information, we extend these registries with semantic markup of service capability profile using DAML-S domain ontology specification language as a basis. This approach allows for more effective manufacturing web service execution.

DAML-S seeks to exploit the semantic capability of DAML. It groups Web Service details into three components and defines respective ontologies called *ServiceProfile*, *ServiceModel*, and *ServiceGrounding* [5]. The Service Profile ontology aims at describing service capabilities, service inputs, service outputs, and preconditions. It is similar to other Web Service architecture

components such as the ebXML collaboration protocol profile (CPP) and the UDDI; yet, it is more expressive and extensible. We are interested in extending the *serviceCategory* property, that is, the properties of services that may be offered (e.g., products, problem solving capabilities, commercial services information). The Service Model defines the process ontology and process control ontology for the interactions between trading partners. The Service Profile and Service Model enable the service discovery and composition. The Service Grounding indicates an ontology to encapsulate technical requirements to communicate, interact, and execute the service.

```
<!-- This is stored at "http://msi.postech.ac.kr/ServiceCategory.daml" -->
<rdf:RDF
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns     ="http://www.daml.org/2001/03/daml+oil#">
    xmlns:damls="http://www.daml.org/services/daml-s/2001/10/Service#"
    xmlns:sp="http://www.daml.org/services/daml-s/2001/10/Profile#"
    xmlns:oper="http://msi.postech.ac.kr/OperationOntology#"
    xmlns:proc="http://msi.postech.ac.kr/ProcessOntology#"
<rdfs:Property rdf:ID="domainServiceCategory">
    <rdfs:subPropertyOf rdf:resource="sp:#serviceCategory"/>
</rdfs:Property>
<Class rdf:about="damls:#ServiceProfile">
    <rdfs:subClassOf>
        <Restriction daml:minCardinality="0">
            <onProperty rdf:resource="#domainServiceCategory"/>
            <toClass rdf:resource="#DomainServiceCategory"/>
        </Restriction>
    </rdfs:subClassOf>
</Class>
<Class rdf:ID="DomainServiceCategory"/>
<Class rdf:ID="OperationalService"/>
    <rdfs:subClassOf rdf:resource="DomainServiceCategory"/>
    <rdfs:subClassOf>
        <Restriction minCardinality="1">
            <onProperty="#hasOperationalCapability"/>
        </Restriction>
    </rdfs:subClassOf>
</Class>
<ObjectProperty rdf:ID="hasOperationalCapability">
    <rdfs:domain rdf:resource="OperationalService"/>
</ObjectProperty>
<ObjectProperty rdf:ID="hasMaterialRemovalCapability>
    <rdfs:subPropertyOf rdf:resource="#hasOperationalCapability">
    <rdfs:range rdf:resource="oper:#MaterialRemovalOperation"/>
</ObjectProperty>
<ObjectProperty rdf:ID="hasMaterialRemovalProcessCapability">
    <rdfs:domain rdf:resource="oper:#MaterialRemovalOperation"/>
    <rdfs:range rdf:resource="proc:#MaterialRemovalProcess"/>
</ObjectProperty>
</rdf:RDF>
```

Figure 11: Partial DAML encoding of service category ontology.

5.2. Manufacturing Service Markup

The extension of the *domainServiceCategory* as a subproperty of the *serviceCategory* is shown in Figure 12. The subproperty is used in order to indicate that a service may fall into multiple service categories with different classification schemes. The domain service category corresponds to industry domains, such as retail, information service, operational service, etc. Operational service is described by the operational capability (*hasOperationalCapability*). Since our focus is on the *MaterialRemovalOperation*, we extend the *hasOperationalCapability* property with the *hasMaterialRemovalCapability*, which allows the range restriction to the *MaterialRemovalOperation*. The process ontology, particularly the *MaterialRemovalProcess*, is then used to increase the expressiveness of the operation. It should be noted that other operation areas could describe their capabilities in a similar manner.

The *MaterialRemovalOperation* from the operation ontology and its subtypes are used under the "capability" context (describing manufacturing capabilities) provided by the property *hasMaterialRemovalCapability*. Similary, the *MaterialRemovalProcess* and its subtypes are used under the "capability" context provided by the property *hasMaterialRemovalProcessCapability*. Additional restrictions and properties for these contexts are specified in Figure 13. Similar extensions are required for the RIPP ontology, which uses the Operation and Process Ontology in the "requirement" context (describing requirements to manufacture the product). Shematron assertions [23] in Figure 14 add semantic definitions for those terms and associated properties relative to a requirement document (the "RIPP.daml" represents a requirement document as described in Section 6.2). For example, it provides the definition of *minDiameter* property that it is the value that the *diameter* in the requirement document must equal or exceed. It should be noted that the units for the capability and the requirement are assumed to be the same. The Schematron assertion limitation should also be noted. The assertion does not take into account the subsumption logics. For example, the assertions should apply to any requirement that is a subtype of the *HoleMakingProcess*; however, a typical Schematron processor will not produce such an effect.

```
<!-- This is stored at "http://msi.postech.ac.kr/ServiceCategory.daml" -->
<rdf:RDF  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns     ="http://www.daml.org/2001/03/daml+oil#">
    xmlns:damls="http://www.daml.org/services/daml-s/2001/10/Service#"
    xmlns:sp="http://www.daml.org/services/daml-s/2001/10/Profile#"
    xmlns:oper="http://msi.postech.ac.kr/OperationOntology#"
    xmlns:proc="http://msi.postech.ac.kr/ProcessOntology#"
<rdfs:Property rdf:ID="domainServiceCategory">
    <rdfs:subPropertyOf rdf:resource="sp:#serviceCategory"/>
</rdfs:Property>
<Class rdf:about="damls:#ServiceProfile">
    <rdfs:subClassOf><Restriction daml:minCardinality="0">
           <onProperty rdf:resource="#domainServiceCategory"/>
           <toClass rdf:resource="#DomainServiceCategory"/>
    </Restriction></rdfs:subClassOf></Class>
<Class rdf:ID="DomainServiceCategory"/>
<Class rdf:ID="OperationalService"/>
    <rdfs:subClassOf rdf:resource="DomainServiceCategory"/>
    <rdfs:subClassOf><Restriction minCardinality="1">
           <onProperty="#hasOperationalCapability"/>
    </Restriction></rdfs:subClassOf></Class>
<ObjectProperty rdf:ID="hasOperationalCapability">
    <rdfs:domain rdf:resource="OperationalService"/>
</ObjectProperty>
<ObjectProperty rdf:ID="hasMaterialRemovalCapability>
    <rdfs:subPropertyOf rdf:resource="#hasOperationalCapability">
    <rdfs:range rdf:resource="oper:#MaterialRemovalOperation"/>
</ObjectProperty>
<ObjectProperty rdf:ID="hasMaterialRemovalProcessCapability">
    <rdfs:domain rdf:resource="oper:#MaterialRemovalOperation"/>
    <rdfs:range rdf:resource="proc:#MaterialRemovalProcess"/>
</ObjectProperty>
</rdf:RDF>
```

Figure 12: Partial DAML encoding of service category ontology.

```
<!--The MaterialRemovalOperation is extended for the "Capability" context-->
<!-- This is stored at "http://msi.postech.ac.kr/OperationCapability.daml" -->
<rdf:RDF  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns     =http://www.daml.org/2001/03/daml+oil#
    xmlns:oper="http://msi.postech.ac.kr/OperationOntology#">
<Class about="oper:#MaterialRemovalOperation">
    <rdfs:subClassOf><Restriction minCardinality="1">
           <onProperty resource="#hasMaterialRemovalProcess"/>
    </Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><Restriction minCardinality="1">
           <onProperty resource="oper:#axis"/>
    </Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><Restriction cardinality="1">
           <onProperty resource="oper:#powerRating"/>
    </Restriction></rdfs:subClassOf>
    <rdfs:subClassOf><Restriction cardinality="1">
           <onProperty resource="oper:#workpieceHoldingSize"/>
    </Restriction></rdfs:subClassOf></Class>
</rdf:RDF>
```

Figure 13: Partial DAML encoding of operation capability ontology

```
<!--***The HoleMakingProcess is extended for the "Capability" context. ***-->
<!-- This is stored at "http://msi.postech.ac.kr/ProcessCapability.daml" -->
<rdf:RDF  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns     ="http://www.daml.org/2001/03/daml+oil#"
   xmlns:support="http://msi.postech.ac.kr/SupportOntology#"
   xmlns:proc="http://msi.postech.ac.kr/ProcessOntology#">
<!--Extend existing properties-->
<ObjectProperty rdf:ID="minDiameter">
   <rdfs:subPropertyOf rdf:resource="proc:#diameter"/>
   <rdfs:domain rdf:resource="proc:#HoleMakingProcess"/>
   <rdfs:range rdf:resource="support:#LengthMeasure"/></ObjectProperty>
<ObjectProperty rdf:ID="maxDiameter">
   <rdfs:subPropertyOf rdf:resource="proc:#diameter"/>
   <rdfs:domain rdf:resource="proc:#HoleMakingProcess"/>
   <rdfs:range rdf:resource="support:#LengthMeasure"/></ObjectProperty>
<!--New property-->
<ObjectProperty rdf:ID="diametricAccuracy">
   <rdfs:domain rdf:resource="proc:#HoleMakingProcess"/>
   <rdfs:range rdf:resource="support:#LengthMeasure"/></ObjectProperty>
<daml:Class about="proc:#HoleMakingProcess">
   <rdfs:subClassOf><Restriction cardinality="1">
      <onProperty resource="#minDiameter"/>
   </Restriction></rdfs:subClassOf>
   <rdfs:subClassOf><Restriction cardinality="1">
      <onProperty resource="#maxDiamter"/>
   </Restriction></rdfs:subClassOf>
   <rdfs:subClassOf><Restriction cardinality="1">
      <onProperty resource="#diametricAccuracy"/>
   </Restriction></rdfs:subClassOf>
</daml:Class><!-- A number of properties should be extended and defined with
cardinality similarly such as minTaperedAngle, maxTaperedAngle, etc.-->

<!--Schematron Semantic Definitions for the Operation and Process Ontology under
the Capability Context. -->
<scmt:schema xmlns:scmt="http://www.ascc.net/xml/schematron">
   <scmt:pattern scmt:name="MaterialRemovalOperation Capability Semantics">
      <scmt:rule scmt:context="hasMaterialRemovalCapability">
         <scmt:assert scmt:test = "document('RIPP.daml')//MaterialRemoval
Operation/powerRating/PowerMeasure/value < powerRating/PowerMeasure/value">
Insufficient power to operate.</assert> <!-- More assertions should be added for
the MaterialRemovalOperation capability context. --></scmt:rule></scmt:pattern>
   <scmt:pattern scmt:name="HoleMakingProcess Capability Semantics">
      <scmt:rule
         scmt:context="hasMaterialRemovalProcessCapability/HoleMakingProcess">
         <scmt:assert scmt:test = "document('RIPP.daml')//HoleMakingProcess/
diameter/TolerancedLengthMeasure/nominalValue <= maxDiameter/LengthMeasure/
nominalValue">Can't produce a hole that big.</assert>
         <scmt:assert scmt:test = "document('processRequirement.daml')//
HoleMakingProcess/diameter/TolerancedLengthMeasure/nominalValue >=
minDiameter/LengthMeasure/nominalValue">Can't produce a hole that small.</assert>
         <scmt:assert scmt:test="document('RIPP.daml')//HoleMakingProcess/
diameter/TolerancedLengthMeasure/tolerancePlus + document('RIPP.daml')//
HoleMakingProcess/diameter/TolerancedLengthMeasure/toleranceMinus >=
diametricAccuracy">Can't make a hole with desired diametric accuracy.</assert>
<!-- More assertions should be added for the HoleMakingProcess
capability context. --></scmt:rule></scmt:pattern></scmt:schema></rdf:RDF>
```

Figure 14: Partial DAML encoding of process capability ontology.

6. Discovery and Filtering of Manufacturing Partners

This section describes the partner filtering procedure implemented in this research. We first create an exemplary manufacturing service profile and then use that to illustrate an approach where a traditional expert system is utilized to filter the partners discovered from the business registry.

6.1. Approaches

It is more space-efficient that the service capability profile be an aggregate capability of the shop than for it to be a list of capabilities associated with each piece of equipment. Each operation and process combination should represent the best attainable capability of the shop in order to maintain consistent levels of ambiguity and explicitness. This approach is a recommendation but not a requirement. A shop can opt to list the combinations of each operation and process that it possesses, but although this increases the filtering efficiency, doing so might narrow down its business opportunity.

An example profile in Figure 15 uses all the ontology developed in previous sections to illustrate a simple markup of a manufacturing service profile. The example states that *PennStateFMS* is a manufacturing service provider who provides only *MechanicalRemovalOperation*. Its manufacturing capabilities include *HoleMakingProcess* and *MillingProcess*. It can be seen that the term *MillingProcess* is not included in the domain ontology defined in the previous sections. However, the *PennStateFMS* provides a definition for the *MillingProcess* in terms of the domain ontology that the *MillingProcess* inherits the properties of both *FaceMakingProcess* and *RoughingProcess* (in other words, *MillingProcess* subsumes *FaceMakingProcess* and *RoughingProcess*). This implies that the manufacturer having milling process capability has both face making and roughing process capabilities. This interlingua capability allows the service to be discovered even though a vendor specific term is used.

```
<!--This exemplary service profile is stored in the
"http://cimlab/webservice/serviceprofile.daml"-->
<rdf:RDF  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns     ="http://www.daml.org/2001/03/daml+oil#"
    xmlns:damls="http://www.daml.org/services/daml-s/2001/10/Service#"
    xmlns:sp="http://www.daml.org/services/daml-s/2001/10/Profile#"
    xmlns:oc="http://msi.postech.ac.kr/OperationCapability#"
    xmlns:support="http://msi.postech.ac.kr/SupportOntology#"
    xmlns:sc="http://msi.postech.ac.kr/ServiceCategory#"
    xmlns:proc="http://msi.postech.ac.kr/ProcessOntology#"
    xmlns:pc="http://msi.postech.ac.kr/ProcessCapability#">

<damls:Service rdf:ID="PennStateFMS">
    <damls:isDescribedBy rdf:resource="http://cimlab/#PennStateFMSServiceModel"/>
    <damls:supports rdf:resource="http://cimlab/#PennStateFMSServiceGrouding"/>
<!--the PennStateFMSServiceModel and PennStateFMSServiceGrouding are assumed to
be described a separate files not in the scope of this paper-->
    <damls:presents rdf:resource="#PennStateFMSServiceProfile>
</damls:Service>
<damls:ServiceProfile rdf:ID="PennStateFMSServiceProfile">
    <damls:isPresentedBy rdf:resource="http://cimlab/#PennStateFMS"/>
    <sp:serviceName>Penn State Manufacturing Service</damls:ServiceName>
    <sp:serviceType>B2B</serviceType>
    <sc:domainServiceCategory>
        <sc:OperationalService rdf:ID="PennStateFMSOperationalService">
            <sc:hasMaterialRemovalCapability>
                <oc:MechanicalRemovalOperation rdf:ID="MachanicalRemovalOp1">
                    <oc:powerRating>
                        <support:PowerMeasure><support:value>5</support:value>
                            <support:unit><support:PowerUnit>HP</support:PowerUnit>
                            </support:unit>
                        </support:PowerMeasure>
                    </oc:PowerRating>
                    <sc:hasMaterialRemovalProcess>
                        <pc:HoleMakingProcess rdf:ID="HoleMakingProcess1">
                            <pc:minDiameter><support:LengthMeasure>
                                    <support:nominalValue>0.125</support:nominalValue>
                                    <support:unit>
                                        <support:LengthUnit>inch</support:LengthUnit>
                                    </support:unit>
                                </support:LengthMeasure></pc:minDiameter>
                        </pc:HoleMakingProcess>
                    </sc:hasMaterialRemovalProcess>
                    <sc:hasMaterialRemovalProcess>
                        <MillingProcess rdf:ID="MillingProcess1"/>
                    </sc:hasMaterialRemovalProcess>
                </oc:MechanicalRemovalOperation>
            </sc:hasMaterialRemovalCapability>
        </sc:OperationalService>
    </sc:domainServiceCategory>
    <daml:Class rdf:ID="MillingProcess">
        <rdfs:subClassOf rdf:resource="proc:#FaceMakingProcess"/>
        <rdfs:subClassOf rdf:resource="proc:#RoughingProcess"/>
    </daml:Class>
</damls:ServiceProfile>
</rdf:RDF>
```

Figure 15: Exemplary manufacturing service profile.

6.2. Discovery and Filtering Procedure

An overview of the partner filtering procedure is shown in Figure 16. After the partners are discovered with pointers to their service web content, an expert system can read in the DAML Service Profiles and transform them into the manufacturing capability facts asserted in the knowledge base. A query generated from each node in the RIPP graph is fed into the expert system to match the capability requirements.

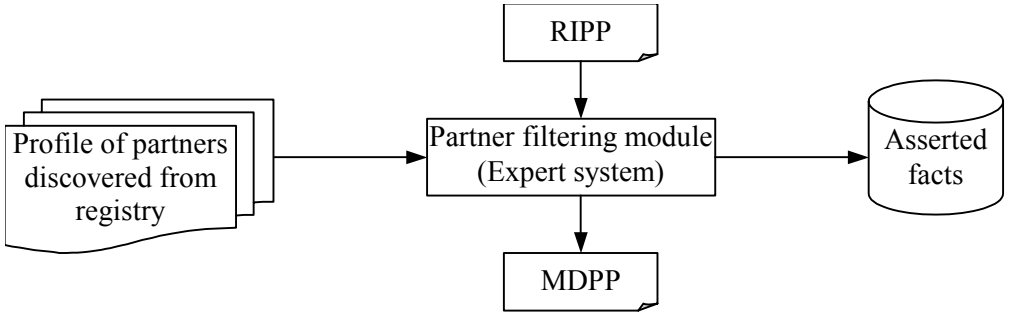

Figure 16: Overview of partner filtering.

In particular, the Java-based Expert System (JESS) [10] is employed to enable the filtering based on description logics of DAML (e.g., subsumption, equality, and set definitions) [7]. DAML-JESS API [16] is used to translate DAML descriptions into JESS assertions. JESS translates DAML descriptions into a set of predicates (ordered-facts) consisting of property or verb, subject, and object. An object instance in DAML is translated into an ordered-fact consisting of *(PropertyValue type <id> <class>)*, where the *PropertyValue* is the head of JESS predicate, the *<id>* is a unique idetification of the object and is generated for anonymous instance and, the *<class> is the class of which the object is an instance*. A property of an object instance in DAML is translated into an ordered-fact consisting of *(PropertyValue <property> <id> <value>)*, where the *PropertyValue* is the head of JESS predicate, the *<id>* is an identification of the object to which the property belongs, the *<property>* is the property term, the *<value>* is the value assigned to the property term. The resulting facts are asserted in the JESS knowledge base as shown in Table 1. The DAML-JESS translator also maintains a set of production rules representing the description logic semantics, a portion of which are shown in Table 2. The rules are fired after the translation, which results in additional assertions. Table 3 illustrates assertions that result from the application of the rules in Table 2 to the facts in Table 1.

Table 1: JESS assertions derived from the service profile in Figure 15.

No.	Fact
Fact-1	`(PropertyValue type #PennStateFMS damls:Service)`
Fact-2	`(PropertyValue damls:presents #PennStateFMS #PennStateFMSServiceProfile)`
Fact-3	`(PropertyValue type #PennStateFMSServiceProfile damls:ServiceProfile)`
Fact-4	`(PropertyValue sc:domainServiceCategory #PennStateFMSServiceProfile #PennStateFMSOperationalService)`
Fact-5	`(PropertyValue type #PennStateFMSOperationalService sc:OperationalService)`
Fact-6	`(PropertyValue sc:hasMaterialRemovalCapability #PennStateFMSOperationalService #MachanicalRemovalOperation1)`
Fact-7	`(PropertyValue type #MachanicalRemovalOperation1 oper:MachanicalRemovalOperation)`
Fact-8	`(PropertyValue sc:hasMaterialRemovalProcess #MachanicalRemovalOperation1 #HoleMakingProcess1)`
Fact-9	`(PropertyValue type #HoleMakingProcess1 proc:HoleMakingProcess)`
Fact-10	`(PropertyValue type #MillingProcess1 #MillingProcess)`
Fact-11	`(PropertyValue subClassOf #MillingProcess proc:FaceMakingProcess)`
Fact-12	`(PropertyValue subClassOf #MillingProcess proc:RoughingProcess)`

Table 2: Production Rules Representing the DAML Subsumption Semantics.

No.	Subsumption Production Rule	Description

Rule-1	`(defrule subclassInstances`
	`  (PropertyValue subClassOf ?child ?parent)`
	`  (PropertyValue type ?instance ?child)`
	`➔(assert (PropertyValue type ?instance ?parent)))`
		An instance of a subclass is an instance of the parent class. This enforces and makes meaningful the rdfs:subClassOf relationship.

Rule-2	`(defrule subPropertyInstances`
	`  (PropertyValue subPropertyOf ?childProperty ?parentProperty)`
	`  (PropertyValue ?childProperty ?classInstance ?value)`
	`➔(assert (PropertyValue ?parentProperty ?classInstance ?value)))`
		An object having a value of a child property also has the same value of the parent property. This asserts the rdfs:subPropertyOf relationship.

Table 3: Facts resulting from the subsumption semantics

No.	Resulted From	Fact
Fact-13	Fact-6, Rule-2, Service Category Ontology	(PropertyValue hasOperationalCapability PennStateFMSOperationalService MachanicalRemovalOperation1)
Fact-14	Fact-7, Rule-1, Operation Ontology	(PropertyValue type, MachanicalRemovalOperation1 MaterialRemovalOperation) (PropertyValue type MachanicalRemovalOperation1 MaterialTransformationOperation), etc.
Fact-15	Fact-10, Fact-11, Rule-1	(PropertyValue type #MillingProcess1 proc:FaceMakingProcess)
Fact-16	Fact-10, Fact-12, Rule-1	PropertyValue type #MillingProcess1 proc:RoughingProcess)

Suppose that RIPP uses the same semantics from the operation and process ontology. The filtering procedure is divided into two passes. In the first pass, a query to the knowledge base is generated to filter only manufacturing partners that have the appropriate operation and process type. In the second pass, Schematron is used to check detailed manufacturing capability against the requirements. JESS query for the first pass can be constructed as shown in Figure 17. The query is applied to each node of the RIPP graph. Only the manufacturing partners that provide appropriate operation and process types are assigned to each node and are forwarded to the second pass. Note that each node may contain an operation requiring several types of processes, in which case the query should be run for each process.

```
(defquery findManufacturingPartner
"Find manufacturing partner as specified by operation type and process type."
"Note that ?serviceID is the returned value. The ?operationTypeVar and the
?processTypeVar are parameters given to the query."
    (declare (variables ?operationTypeVar ?processTypeVar))
    (PropertyValue type ?serviceID damls:Service)
    (PropertyValue damls:presents ?serviceID ?serviceProfileID)
    (PropertyValue type ?serviceProfileID damls:serviceProfile)
    (PropertyValue sc:domainServiceCategory ?serviceProfileID
       ?OperationalServiceID)
    (PropertyValue sc:hasMaterialRemovalCapability ?operationalServiceID
       ?operationTypeID)
    (PropertyValue type ?operationTypeID ?operationTypeVar)
    (PropertyValue sc:hasMaterialRemovalProcess ?operationTypeID ?processTypeID)
    (PropertyValue type ?ProcessTypeID ?processTypeVar)
)
```

Figure 17: JESS operation and process type query for filtering in the first pass.

Suppose that a node in an RIPP graph requires *MechanicalRemovalOperation* with a *HoleMakingProcess* and a *FaceMakingProcess*. First, the *MechanicalRemovalOperation* and the *HoleMakingProcess* are assigned to the variables *?operationTypeVa*r and *?processTypeVar*, respectively. The query will return the *PennStateFMS* due to the crude facts in Table 1. The *PennStateFMS* is also a feasible partner for the second query where the *MechanicalRemovalOperation* and the *HoleMakingProcess* are assigned to the two variables due to the subsumption facts in Table 3.

In the second pass, the service capability profile is checked against the assertions in the capability context. In this case, the content of each node in the RIPP (Resource-independent Process Plan) is extracted into a file called 'RIPP.daml'. The Schematron processor takes the semantic assertions, 'RIPP.daml' file, and the profile of each manufacturing partner from the first pass to execute (see Figure 14). If the test produces an empty string for an operation node and its process level graph, the partner makes it through the second pass and is assigned to that operation node in the MDPP graph. This is done for each operation node of the RIPP and profile from the first pass.

The partner filtering using a semantic web helps increase web service efficiency by adding the rich semantics of service capability profile. Typically, hundreds or thousands of registry entries of manufacturers could turn up from the public registry, which provides only rough service classification. Without these detailed profiles, the service client does not know that a manufacturer does not match its capability requirements until after the service execution. Each service execution requires significant time and communication overhead even if it is automatically executed due to security protocols and agreement establishments required during and before the business interactions. The partner filtering approach described in this paper is passive (i.e., information retrieval). Security and agreement establishments are not necessary; therefore, this approach is more efficient.

7. Conclusion

Approaches and frameworks that enhance current B2B integration technologies to support Distributed Manufacturing in a loosely integrated Virtual Enterprise environment have been presented. The use of graph-based requirement specification of Integrated Product and Process Data to construct a distributed manufacturing plan has been illustrated. A manufacturing service

capability profile has been proposed as a necessary component to discover appropriate manufacturing partners. Upper level semantic web based ontologies, especially for the machine shop domain, to specify such a profile have been illustrated using DAML. The context-based approach using Schematron to unambiguously express the semantic extension from the upper level ontologies has been presented. A Schematron assertion has been found to be limited to a syntactic pattern matching; hence, the subsumption logics of DAML are not fully exploited. Either the scope of XPATH expression used by Schematron has to be extended or a DAML-based rule expression language should be developed. Finally, we presented an expert system approach to utilize the DAML manufacturing capability profile for the service discovery step. The current research is to develop and implement an approach to fully construct distributed process planning and manufacturing using this integration framework.

Reference

1. "Uniform Resource Identifier." *Internet Web Site* (accessed January 2002) available online at http://www.w3c.org/Addressing/.
2. Berfield, A., Chrysanthis, P. K., Tsamardinos, I., Pollack, M. E., and Banerjeez, S., "A scheme for integrating E-Services in establishing virtual enterprises," *Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDEŁ02)*, pp. 134-142, San Jose, California, February 24 - 25, 2002.
3. Cho, H., Jung, M., and Kim, M., "Enabling technologies of agile manufacturing and its related activities in Korea," *Computers & Industrial Engineering*, Vol. 30, No. 3, pp. 323-334, 1996.
4. Ding, Y., Fensel, D., Klein, M., and Omelayenko, B., "The semantic web: yet another hip?" *Data & Knowledge Engineering*, Vol. 41, No. 2, pp. 205-228, 2002.
5. DAML Services Coalition, "DAML-S: Semantic markup for web services," *Proceedings of International Semantic Web Working Symposium*, pp. 411-430, Stanford University, California, July 30 - August 1, 2001.
6. DAML-based Services (DAML-S), Available online via <http://www.daml.org/services> [accessed March 2002].
7. DARPA Agent Markup Language Web Site, Available online via <http://www.daml.org> [accessed July 2001].
8. D&B Corporate Web Site, Available online via http://www.dnb.com [accessed August 2002].
9. Fensel, D., Horrocks, I., Harmelen, F., McGuinness, D. L., and Patel-Schneider, P. F., "The semantic web - oil: an ontology infrastructure for the semantic web," *IEEE Intelligent Systems & Their Applications*, Vol. 16, No. 2, pp. 38-45, 2001.
10. Friedman-Hill, E., "Jess the Rule Engine for the Java™ Platform," Available online via <http://herzberg.ca.sandia.gov/jess/> [accessed July 2002].

11. Furst, K. and Schmidt, T., "Turbulent markets need flexible supply chain communication," *Production Planning & Control*, Vol. 12, No. 5, pp. 525-533, 2001.

12. John, J., "EDI Research," Available online via <http://www.unf.edu/~jinj0001/ediresearch.html> [accessed May 2002].

13. Hendler, J., "The Semantic Web - Agents and the Semantic Web," *IEEE Intelligent Systems & Their Applications*. Vol. 16, No. 2, pp. 30-37, 2001.

14. ISO. "Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual," Technical Report ISO 10303-11:1994(E).

15. Kidd, T., *Agile Manufacturing: Forging New Frontiers*, Addison-Wesley, Reading, Massachusetts, 1994.

16. Kopena, J., "DAMLJessKB". Available online via <http://plan.mcs.drexel.edu/projects/legorobots/design/software/DAMLJessKB/> [accessed June 2001].

17. Kulvatunyou, B., Ivezic, N., Jones, A.T., and Wysk, R.A. (2002). Integrated Product and Process Data for B2B Integration. Submitted to Artificial Intelligence in Engineering Design, Analysis, and Manufacturing Special Issue in New AI Paradigms for Manufacturing.

18. Lassila, O., "The resource description framework," *IEEE Intelligent Systems*, Vol. 15, No. 6, pp. 67-69, 2000.

19. McIlraith, S. A., Son, T. C., and Zeng, H., "The semantic web - semantic web services," *IEEE Intelligent Systems & Their Applications*, Vol. 16, No. 2, pp. 46-53, 2001.

20. Open Application Group Integration Business Object Document Specification (OAGI BOD), Available online via <www.openapplications.org/download/oagidownload.htm> [accessed March 2002].

21. RosettaNet Available online via <http://www.rosettanet.org > [accessed August 2002].

22. Shim, S. S. Y., Pendyala, V. S., Sundaram, M., and Gao, J. Z., "Business-to-business e-commerce frameworks," *Computer*, Vol. 33, No. 10, pp. 40-47, 2000.

23. The Schematron, An XML Schema Validation Language using Patterns in Trees, Available online via <http://www.ascc.net/xml/resource/schematron/schematron.html> [accessed June 2002].

24. UN/CEFACT and OASIS ebXML Business Process Specifications Schema, Available online via <http://www.ebXML.org/specs/index.html> [accessed December 2001].

25. UN/CEFACT ebXML Technical Architecture Specification, Available online via <http://www.ebXML.org/specs/index.html> [accessed December 2001].

26. Universal Description, Discovery, and Execution (UDDI), Available online via <http://www.uddi.org> [accessed February 2002].

27. Universal Standard Products and Services Classification (UNSPSC) – Public Version, Available online via *The Electronic Commerce Code Management Association (ECMMA) Download Web Site* <http://www.eccma.org/downloads.php3> [accessed January 2002].

28. North American Industry Classification System, Available online via *The US Census Web Site* < http://www.census.gov/eos/www/napcs/napcs.htm> [accessed July 2002].
29. Web Service Description Language (WSDL), Available online via <http://www.w3c.org/2002/ws/desc> [accessed March 2002].

Author Biography

BOONSERM KULVATUNYOU is currently a guest researcher at the National Institute of Standards and Technology (NIST) from Oak Ridge Associated University. He recently received his Ph.D. from the Pennsylvania State University, University Park, in 2001. He received his MS from Columbia University, NY, in 1997, and his BS from Chulalongkorn University, Bangkok, Thailand in 1995. He is a senior member of the Society of Manufacturing Engineers (SME) and Computer and Automated Systems Association (CASA) of SME. He is also a member of the American Association for Artificial Intelligence (AAAI). His research interests include computer-integrated manufacturing system, simulation of manufacturing system, enterprise and e-business integration, and information modeling. His email and web addresses are <serm@nist.gov> and <http://serm.ws>.

DR. HYUNBO CHO is an associate professor in the Department of Industrial Engineering at the Pohang University of Science and Technology. He received his B.S. and M.S. degrees in Industrial Engineering from Seoul National University in 1986 and 1988, respectively, and his Ph.D. in Industrial Engineering with a specialization in Manufacturing Systems Engineering from Texas A&M University in 1993. His Ph.D. dissertation was associated with defining and implementing an intelligent workstation controller for CIM. He was a recipient of the SME's 1997 Outstanding Young Manufacturing Engineer Award. His areas of expertise include Shop Floor Control, Process Engineering, and e-Manufacturing. He is an active member of IIE and SME.

DR. YOUNG JUN SON is an assistant professor in the Department of Systems and Industrial Engineering at The University of Arizona. Dr. Son received his BS degree in Industrial Engineering with honors from POSTECH in Korea in 1996 and his MS and Ph.D. degrees in Industrial and Manufacturing Engineering from Penn State University in 1998 and 2000, respectively. His research interests include computer integrated manufacturing, simulation based shop floor control, distributed simulation, virtual manufacturing, and virtual enterprises. Dr. Son was the Rotary International Multi-Year Ambassadorial Scholar in 1996, the Council of Logistics Management Scholar in 1997, and the recipient of the Graham Endowed Fellowship for

Engineering at Penn State University in 1999. He is an associate editor of the International Journal of Modeling and Simulation and a professional member of ASME, IEEE, IIE, INFORMS, and SME. His email and web addresses are <son@sie.arizona.edu> and <www.sie.arizona.edu\faculty\son>.