Ontology-Based Knowledge Management for Co-operative Supply Chain Configuration

Alexander V. Smirnov[1,2], Charu Chandra[1]

[1]Industrial & Manufacturing Systems Engineering Department
College of Engineering and Computer Science
University of Michigan - Dearborn
4901 Evergreen Road, Dearborn, Michigan, 48128-1491, U.S.A.
e-mail: {charu; smir@umich.edu}

[2]St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
39, 14th Line, St. Petersburg, 199178, Russia
e-mail: smir@mail.iias.spb.su

Abstract

The interest in Supply Chain (SC) decision-making problem is fast growing. There is an increasing use of artificial intelligence (AI)-based engineering and management technologies in supply chain problem solving. Furthermore, the trend is towards product data and knowledge globalisation in the SC network. The purpose of a SC is to transform incomplete information about the market and available production resources into *co-ordinated* plans for production and replenishment of goods and services in the network formed by *co-operating* entities. The value of this information is critical to managing the network, built on principles of a co-operative supply chain (CSC). The goal of SC Knowledge Management (KM) is to facilitate knowledge transfer and sharing in the context of SC structures, thereby integrating the customer and the supplier. Ontology-oriented KM tools for SC configuration utilise reusable components and configure knowledge as needed, in order to interactively assist users (agents) in decision-making. This paper discusses a generic development methodology for information support of supply chain management using an industry example.

Introduction

Knowledge is a critical resource for any business activity. Firms utilise industrial knowledge to manage change in its environment due to product life cycle-time & cost reductions, and variations in product & process specifications. Every firm has collective knowledge in the form of its technological competencies. In order to capitalise on this knowledge base, firms have to organise and manage it in creative and useful ways. As a result, new information technologies, such as Knowledge Management, and Agent Technology are increasingly being used for the purpose.

As firms compete in a global economy, new models of transacting business, such as business-to-business (goods & services), and business-to-consumers have emerged.

Novel and creative business partnerships, such as supply chain (SC), and virtual enterprise networks are being formed. These are highly flexible and co-operative business arrangements among partners in order to stay competitive in a dynamic environment. SC configuration is an approach to "network enterprise" creation and reuse that considers enterprises as assemblies of reusable components (units) defined on a common domain knowledge model, such as a "product – process - resources" model described later in this paper.

The objective of this paper is to describe (a) elements of a generic methodology, and (b) its implementation through a pilot project of an ontology-based CSC configuration knowledge management.

The rest of the paper is organised as follows. First, a co-operative supply chain is defined as a special class of a supply chain network. A systems approach applied to managing it describes a supply chain management system. For efficient management of a supply chain, its design must be capable of being configured flexibly. A scheme for supply chain configuration is described to accomplish this. Configuring a supply chain involves, (a) managing its knowledge, (b) modelling the constraint network, and (c) managing knowledge among agents identified in the network. These are described next in the paper. A pilot project of an ontology-based environment for CSC configuration knowledge management is described following this. Finally, conclusions on this research and plans to extend it are presented.

Co-operative Supply Chain

The concept of supply chain is about managing co-ordinated information and material flows, plant operations, and logistics (Lee and Billington 1993). It provides flexibility and agility in responding to consumer demand

shifts with minimum cost overlays in resource utilisation. The fundamental premise of this philosophy is synchronisation among multiple autonomous business entities represented in it. That is, improved co-ordination *within* and *between* various supply chain members. Co-ordination is achieved within the framework of commitments made by Members to each other. Members negotiate and compromise in a spirit of co-operation, in order to meet these commitments. Hence, the label co-operative supply-chain (CSC). Increased co-ordination can lead to reduction in lead times and costs, alignment of interdependent decision-making processes, and improvement in the overall performance of each Member, as well as the supply chain network (Group) (Chandra 1997, Poirier 1999, Tzafestas and Kapsiotis 1994).

Supply Chain Management System

Supply chain management is the art and science of managing this complex network of interrelated systems and their components. It encompasses identifying goals, and objectives of the supply chain and outlining policies, strategies, and controls for its effective and efficient implementation. A formal mechanism to organise these cohesively requires a systems approach. A supply chain management system (SCMS) is proposed for this purpose. It is a system (S) that describes Member (M) entities in a supply chain and their relationship (R) to one another. Notationally, S = (M, R), where M = $\{m_1, m_2, ..., m_n\}$, and R = $\{[(m_1,m_2)\ (r_1,r_2,..,r_n)], ..., [(m_1,m_n)\ (r_1,r_2,..,r_n)], \ldots, [(m_{n-1},m_n)\ (r_1,r_2,..,r_n)]\}$. Likewise, such a relationship can also be expressed for components of a Member (m_i); at a function (or business) m_i (b_i); process $\{(m_i,b_i)\ (p_i)\}$; and activity $\{(m_i,b_i,p_i)\ (a_i)\}$; i = 1,2,...,n. Relationships in a SCMS describe, (a) actions between its members (and their components) involving exchange of information, controls, and resources, and (b) meta (or logical) systems (that is, Members and their components), decentralisation, and specialisation of autonomous Members (Bond and Gasser 1998, Gasser 1991, Moulin and Chaib-Draa 1996).

Figure 1 depicts a decentralised view of a textile supply chain. This CSC is comprised of a Group and more than one Member.

The network is arranged in the order of flow of materials, processes, and information between its Members. In this example, consumer demand is relayed by retailer to apparel maker, textile manufacturer, fibre manufacturer, and to cotton grower. Similarly, flow of material occurs in transforming cotton to yarn by fibre manufacturer, fibre to fabric by textile manufacturer, textile to apparel by apparel maker, and a name brand garment by retailer. The interaction between Members occurs as a *consumer* and a *provider*. Thus, an apparel maker assumes the role of a provider (of apparels) in its dealings with retailer (a consumer of apparels). However, the apparel maker acts as a consumer of fabric while dealing with a textile manufacturer (a provider of fabric).

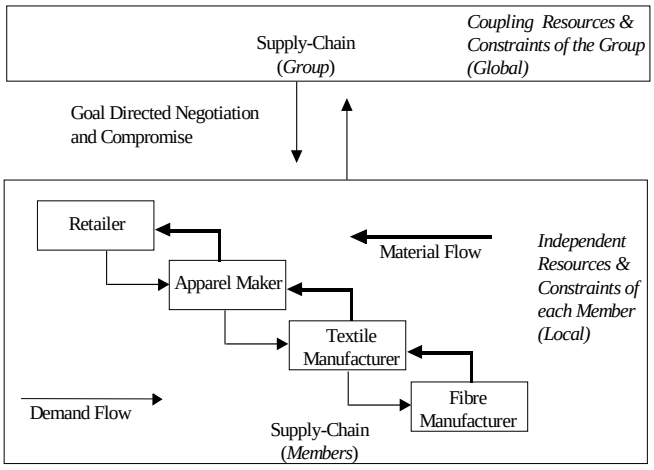

Figure 1. A co-operative supply chain decentralised enterprise view

A decentralised CSC is a physically and logically distributed system of interacting components and elements of autonomous business entities (Members) (Hillier and Lieberman 1990, and Taha 1987). In this distributed problem-solving environment, the task of solving a problem is divided among a number of modules or nodes (autonomous business entities and their systems). Members co-operatively decompose and share knowledge on the problem and its evolving solutions. Interactions between Members in the form of co-operation and co-ordination are incorporated as problem-solving strategies for the system. Entity Group is responsible for co-ordination throughout the supply chain. Entity Member brings specialised expert knowledge and product and process technology(ies) to the supply chain. The decision-making process is centralised for the Group. The Group enforces common goals and policies for the supply chain on Members. However, decision-making at Member is decentralised. Each Member pursues its own goals, objectives, and policies conceptually, independently of the Group, but pragmatically in congruence with Group goals. A common knowledge base supports the CSC structure (Chandra 1997). Knowledge is assimilated for an *activity* (the lowest level of information) in a specific domain and aggregated for various decision-making levels in the enterprise. Main concepts in *activity modelling* in a CSC decentralised enterprise are described below.

- *Activity* represents transformation of input (in the form of a technology) to an output (or product) through use of resources of an enterprise, such as a supply chain.
- *Process* is a collection of activities representing various forms of technologies mobilised by the enterprise in generating an output.

- *Supply Chain Management* comprises of activities or processes. These entities when associated to a user assume unique ontological forms.
- *Ontology* is a unique form of representing knowledge applied in various domains. It is useful in creating unique models of a CSC by creating specialised knowledge bases specific to various supply chain problem domains.
- *Representation*. An *activity* represents the lowest level of interaction in the supply chain model. It is synonymous with a "Member" for modelling business process, and an "agent" for knowledge management environment. It is classified into various *activity types* depending on unique *service*(s) they provide. Activity (ies) is used in relation to an aggregation. An activity possesses *attribute*(s), which describe its characteristics or features. An attribute assumes *parametric* values in relation to an aggregation model. Activities communicate with each other by exchanging *message*(s). Communication occurs based on a *protocol* whose boundaries are set by a *control* matrix prescribing level of *resource*(s) to be utilised by an activity, policies to be pursued, and objectives to be met in providing the service(s).
- *Aggregation*. Aggregation represents a *system* form, depicted in figure 2 for a textile operation. It has seven components -- input, process sequence, output, mechanism, agent, environment, and function, described in (Nadler 1970), which are defined by four matrices, namely, resource, performance, technology, and input/output. Each aggregation (system) has its own control matrix to define relationships between its components. Aggregation can take on many forms manifested by the orientation it is based upon, that is aggregation "within" system(s), or aggregation "between" systems. For example, a material-life-cycle-flow and order-life-cycle represent horizontal aggregation between systems. Building decision models across the enterprise represents a vertical aggregation between systems. Similarly, aggregating all activities within a Member function represents "within" systems integration.
- *Protocol*. Protocols for each aggregation (system) describe conventions governing communication between activities, services rendered by activities to one another, and controls for that system.
- *Communication* between activities occurs in the form of *message*(s) exchanged to request a service.
- *Services* are of resource and information types.

Supply Chain Configuration

As the business environment is changing rapidly, supply chains are being designed innovatively to offer competitive advantage. Some of the potential uses of flexible supply chain are:

- Dealing with changing production demand, and
- Trading product models on business networks.

The efficient management of this SC, therefore, requires that its design be properly configured. An important facet of configuration is to understand the relationship between supply chain system components that define its structure, that is products, and associated processes & resources.

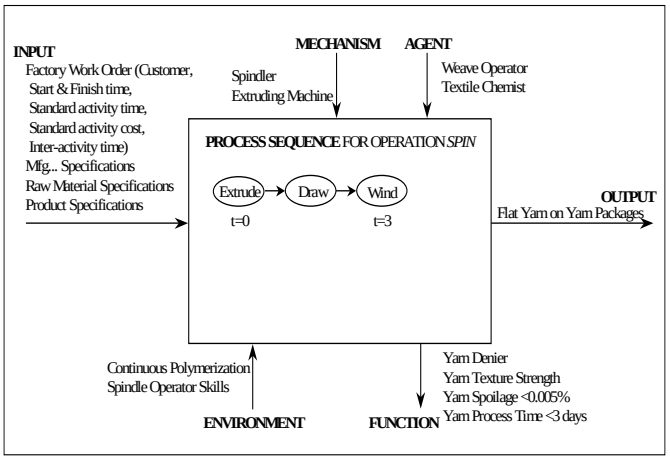

Figure 2. A generic system representation in a co-operative supply chain

The objective of designing SC utilising configuration principles is to generate customised solutions based on standard components, such as templates, baselines, and models. There are two aspects to configuration:

1. Configuring / reconfiguring, and
2. Configuration maintenance.

Configuring deals with creating configuration solutions and selecting components and ways to configure these. Configuration maintenance deals with maintaining a consistent configuration under changing environment. This requires consistency among selected components and decisions. When a decision for selected component changes, configuration maintenance must trace all related decisions and revise them, if necessary. Thus, consistency among components and decisions is maintained. A corporate knowledge management (KM) spanning the supply chain is desired to implement SC configuration.

Supply Chain Knowledge Management

An important requirement for collaborative system is the ability to capture knowledge from multiple domains and store it in a form that facilitates reuse and sharing (Neches et al. 1991, Patil et al. 1992). KM could be identified by four factors behind successful KM systems (Donkin 1998):

1. An understanding by employees as to why knowledge sharing is important,
2. Recognition by employees,
3. Legacy of existing practices, and
4. Support mechanism or safety net that allows employees to experiment.

KM is 90 per cent people and 10 per cent technology. General functions of KM are -- externalisation, internalisation, intermediation, and Cognition (Delphigroup 1998). These describe the relation "user – knowledge / databases".

The approach suggested in this paper is limited to designing SC configurations for manufacturing systems and focused on using ontological descriptions. It is based on GERAM, the Generalised Enterprise Reference Architecture and Methodology (ISO TC 184/SC 5/WG 1 1997). This approach is one of several key approaches evaluated and recommended by the IFAC/IFIP Task Force on Enterprise Integration that facilitates developing an overall definition of a generalised architecture. It could be implemented using a variety of Enterprise modelling languages, such as ARIS, CIMOSA, GRAI/GIM, IEM, and the IDEF family of languages.

Applying the GERAM approach enables forming the conceptual model of the SC system. This is accomplished by modelling its product, process, and resource components to satisfy manufacturing constraints in its environment.

The SC configuration stage is represented by the following relation:

> "*configuring the product* (product structure, materials bill) → *configuring the business process* (process structure, operation types) → *configuring the resource* (structure of system, equipment and staff types)".

The implementation of SC approach is based on the shareable information environment that supports the "product - process - resource" model (PPR-model) used for integration and co-ordination of user's activity. This model is studied from different viewpoints or user groups as depicted in figure 3.

Enterprise models may be defined in various ways. In increasing order of formality, generic enterprise models may be defined as (ISO TC 184/SC 5/WG 1 1997):

- Natural language explanation of the meaning of modelling concepts (*glossaries of terms*),
- Some forms of meta models for example, *entity relationship, meta schemas, conceptual models of terminology, and component modelling languages* describing the relationship among modelling concepts available in enterprise modelling languages, and
- Ontological theories defining the meaning (*semantics*) of enterprise modelling languages to improve the analytic capability of engineering tools and through them the usefulness of enterprise models.

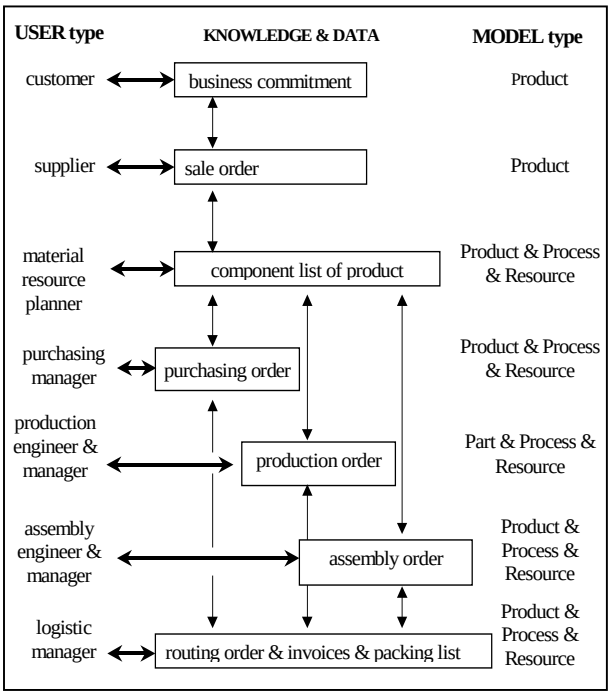

Figure 3. Links of users to knowledge and data in SC

Ontologies are created similar to how knowledge domain models are built. Formal steps in creating ontologies are:

- Defining entities,
- Assigning attributes to entities with regard to its domain,
- Identifying relationships between entities and domain,
- Identifying relationships between entities,
- Describing levels of constraints, and
- Labelling ontologies by classes of entities.

Ontologies are managed by translation and mapping between different types of attributes and entities.

Ontological theories are formal models of concepts that are used in SC model representations. They capture rules and constraints of the domain of interest, allowing useful inferences to be drawn, analyse, execute, cross check, and validate models.

Ontological translation of an enterprise, such as a supply chain is necessary because networks are multi-ontology classes of entities. Various ontologies for an entity describe its unique characteristics in context with the relationship acquired for a specific purpose or problem. For example, an entity "textiles" may have a multi-ontology representation for a user with a marketing perspective, and for another user with a design perspective, respectively. For the user interested in the marketing perspective of textiles, its attributes of size, denier, and style are important. However, the same textiles'

characteristics may be represented by size, quality, and finish for the user interested in its design specifications.

Object-Oriented Dynamic Constraint Network as Top-Level Ontology Paradigm for Supply Chain Configuration

It is widely accepted that engineering & management activities can be regarded as search involving techniques to satisfy constraint (Giachetti et al. 1997, Hirsch 1995, Tsang 1991). Constraint satisfaction is a fundamental SC problem. Conventional constraint satisfaction procedures are designed for the problem with one constant set of constraints. For example, in engineering for manufacturing systems, such as design for productivity, layout, and scheduling, it is often necessary to solve a dynamic constraint satisfaction problem where applicable constraints depend on the design aspect (Smirnov 1994).

The domain knowledge model of SC contains entities (objects), which can be of different types (classes). Multi-level and multi-aspect/viewpoint are used for PPR-model description. Obviously, each user works with its own ontology-oriented constraint network. KM tools support the conversion of PPR-model, from one ontology to another.

An abstract PPR-model is based on the concept of ontology-oriented constraint networks. Networks are represented by multi-ontology classes of entities, the logic of attributes and the constraint satisfaction problem model. This abstract model unifies main concepts of languages, such as standard object-oriented languages with classes, and constraint programming languages. It supports the declarative representation, efficiency of dynamic constraint solving, as well as problem modelling capability, maintainability, reusability, and extensibility of the object-oriented technology.

Ontology-oriented constraints network model is denoted A = *(St, C)*, where *St* — an ontology structure, *C* — a set of ontology constraints. To deal with the concept schema of configuring process defined in terms of constraints, a dynamic constraint network (DCN) model is applied. A static constraint network (SCN) $A_i = \left(V_i, D_i, C_i\right)$, involves a set of variables $V_i = \left(v_{i1}, v_{i2}, \ldots, v_{iN_i}\right)$, each taking value in its respective domain $D_i = D_{i1} \times D_{i2} \times \ldots \times D_{ij} \times \ldots \times D_{iN_i} = \underset{j=1}{\overset{N_i}{\times}} D_{ij}$, and a set of constraints $C_i = \left\{c_{i1}, c_{i2}, \ldots, c_{ik}\right\}$. A DCN ***N*** is a sequence of SCNs, each resulting from a change in the proceeding one imposed by "the outside world". For description of top-level ontology the following abstract model based on integration of DCN model and object-oriented model (Royer 1992) could be used:

Model: $M_c =$

$= \left[< A_1, F_{11}, \ldots, F_{1n} >, \ldots, < A_n, F_{n1}, \ldots, F_{nn} >\right]$,

where, A_i — an ontology, F_{ik} a conversion from an ontology A_i to A_k, and F_{ii} — an identity.

Ontology: $A_i = \left(V_i, D_i, C_i\right)$ is a SCN.

Conversion: $F_{ik} = \underset{j=1}{\overset{N_i}{\times}} v_{ij} = E_{D_{ij}} \rightarrow \underset{l=1}{\overset{N_i}{\times}} v_{kl} = E_{D_{KL}}$,

where, $v_{ij} \in V_i$, $j = 1, \ldots, N_i$, $D_{ij} \subseteq D_i$, and $\underset{j=1}{\overset{N_i}{\times}} v_{ij} = E_{D_{ij}}$ is a set $\left\{\left(v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i}\right) \middle| e_{ij} \in E_{D_{ij}}\right\}$ describing the labelled Cartesian product, built on the structure $\left(V_i, D_i\right)$. The term $E_{D_{ij}}$ designates the values set of D_{ij}.

Constraints:

$C_i : \underset{j=1}{\overset{N_i}{\times}} v_{ij} = E_{D_{ij}} \rightarrow Boolean = \left\{true, false\right\}$.

Instance: $< d_1, \ldots, d_n >$, where $d_i =< A_i, \left(v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i}\right) >$ is a description in ontology A_i, $< v_{i1} = e_{i1}, \ldots, v_{iN_i} = e_{iN_i} >$ — a tuple of equalities in $\underset{j=1}{\overset{N_i}{\times}} v_{ij} = E_{D_{ij}}$.

The above Ontology Management approach is based on two mechanisms:

- Class inheritance mechanism, supported by inheritance of class ontologies (attributes inheritance) and by inheritance of constraints on class attribute values, and
- Constraint inheritance mechanism for inter-ontology conversion, supported by constraint inheritance for general model (constraints strengthening for "top-down" or "begin-end" processes).

Integration of Ontology and Agent

The implementation of the basic principle for the SC, i.e., its collaborative nature is based on distribution of procedures between different users (or different agents). For this purpose, it is obvious to represent the SC configuration KM as a set of interactive autonomous agents. Different configuration problems are treated as separate agent-oriented tasks with embedded constraint satisfaction and consistency support facilities (Sheremetov and Smirnov 1997, Smirnov and Sheremetov 1998). Agent-based distributed constraint satisfaction problem is a

problem in which variables and constraints are distributed among agents. Agents communicate by sending messages. The chart of co-operation cycle for agents has the following structure:

1. Application domain knowledge is shared between agents; knowledge about constraints (user requirements and artefacts) are transmitted to particular agents,
2. Agents solve local configuration tasks on the basis of shared knowledge, and
3. Results are collected and transmitted to all interested agents.

These stages are continuously repeated until the solution is globally confirmed or consensus can not be reached, and constraint relaxation methods are to be applied.

DESO Project: Object-oriented Tool for Corporate Ontology Creation and Maintenance

The goal of DESO (DEsign of Structured Objects) project is to develop a methodology and tools for automated re-use of industrial experience from large collection of knowledge & data in the engineering and business domains. Figure 4 depicts the DESO architecture. It aims at establishing a knowledge platform that enables manufacturing enterprises to achieve reduced lead-time and cost based on customer satisfaction by means of improved availability, communication and quality of product information. DESO follows a decentralised method for intelligent knowledge and solutions access. The configuration process incorporates following features: order-free selection, limits of resources, optimisation (minimisation or maximisation), default values, and freedom to make changes to the SC Model.

An ontology-oriented SC model depicted in figure 5 is an object library composed of the following:

- A set of object classes structured as a taxonomy, i.e., each object is linked to one or more other objects by a sub-class / super-class relationship.
- For each object class, a set of relations is defined linking it to other object classes, as well as a constraint set for each relation, and
- For each object class, a set of attributes and a definition of the intended meaning of each attribute are provided.

The "product-process-resource" model serves as a knowledge repository for manufacturing system. The structure of the DB formed by it is presented in figure 6.

Basic units of the DESO tool database are objects, such as “item”, “machine”, “operation”, “plant” etc. The highest level of object abstraction is object Class, which captures an object’s nature. Classes are defined and entered into database by the developer at the installation stage. The next level of object abstraction is object Type. Each type belongs to one of the classes of the system and contains objects with similar properties. At this level, object attributes (one type of object possesses identical set of attributes) are defined. The user of the tool may create, edit and delete types. At the level of objects, values for each attribute may be entered and edited.

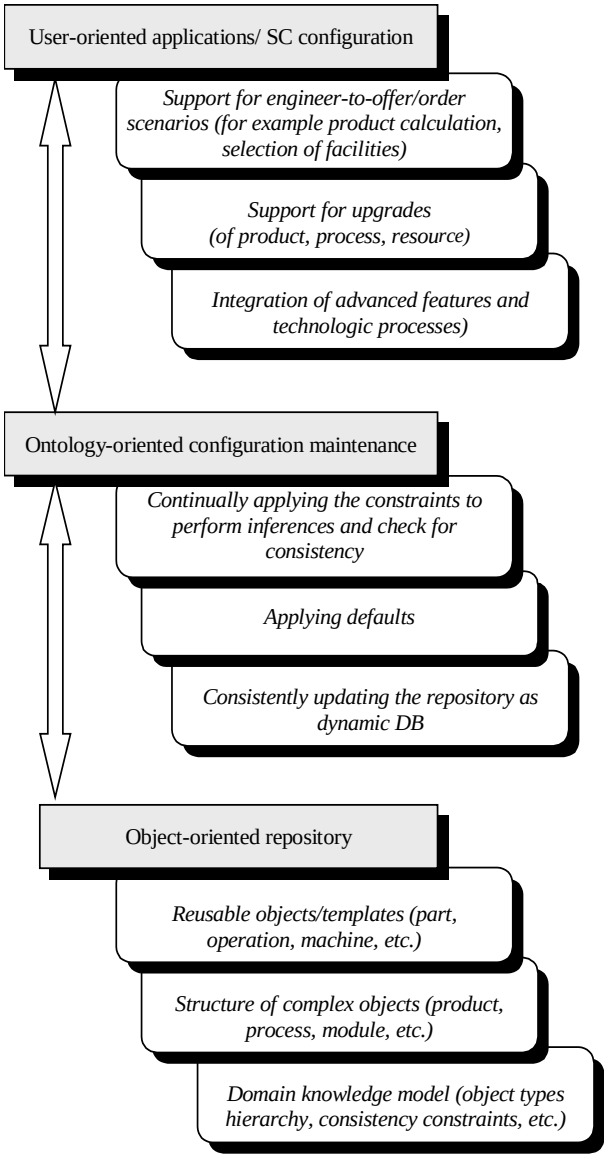

Figure 4. DESO architecture

For example:

Class = Machine, Type = Extruding Machine.

Attributes, objects and values are shown in Table I.

Table I. DESO: Object Properties

Attributes: Objects	Max Diameter Mm	Precision	Price, USD
M1415	1500	4	55,800
M1335	3500	3	63,400

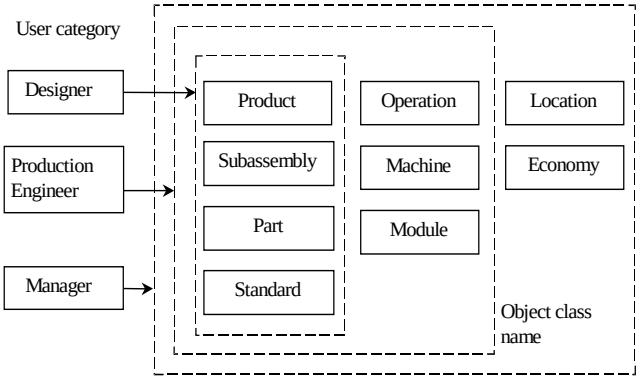

Figure 5. Sharing ontology example for the "product-process-resource" model

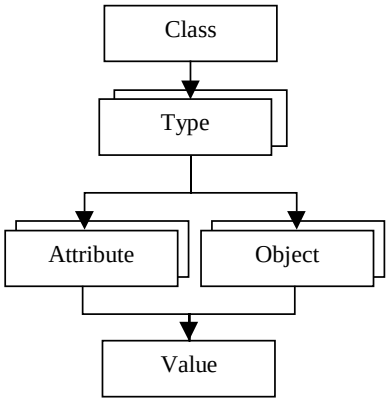

Figure 6. DESO: database structure

The database structure allows setting relations between classes and types of *objects*, e.g., machines of "Extruding_type_1" and "Extruding_type_2" can perform the extrusion operation. At the level of *types*, complex constraints for values of attributes can be set, e.g., the length and the diameter of a part should not exceed maximum values allowed for turning machines. An example depicted in figure 7 is described as follows:

- Operation "123T – A" can process only part "O – 123T", and
- Operation "123T – B" can process both part "O – 123T" and part "I – 123T".

Constraint for parts «Yarn» and operations «Extrude, Draw, Wind»:

- Part.L1 ≤ Operation.L1, and
- Part.L Ref ≤ Operation.LREF

Properties for parts are given in Tables II, and III.

Table II. Properties for parts «Yarn»

Attribute	O – 123T	I – 123T
L1	1300	1500
L Ref	1200	1450

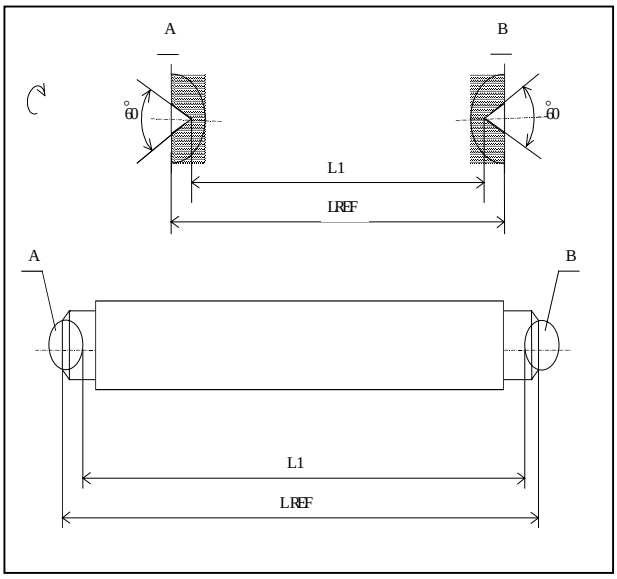

Figure 7. DESO: The use of constraints (upper figure is operation, lower one is part)

Table III. Properties for operations «Extrude, Draw, Wind»

Attribute	123T – A	123T – B
L1	1300	1500
L REF	1400	1600

The hierarchy editor is a tool for creating, editing and managing hierarchical relations between objects, e.g., "part > technological process > operation > machine". These relations may show structures of objects, sequence of operations for a part production, and possible alternatives of accommodation etc. The hierarchy editor supports inheriting subordinate objects, that allows creating complex hierarchical systems of objects by some stages and using templates automating the user's work.

The user could evaluate systems costs for every unit, such as time, efficiency, fixed and variable production costs. Every cost centre could include several machine tools.

Conclusions

The universality of the knowledge representation by object-oriented dynamic constraint networks for all kinds of SC models makes it feasible to provide powerful interactive tool for DB & KB maintenance. Knowledge management environment comprises means for SC project consistency control and allows multiple aspect / ontology-oriented collaborative engineering. Ontology-based DESO architecture supports co-operation among agents, thereby reducing time and increasing the quality of SC configuration process. Ontology-based knowledge management technology is an innovative technology in the domain of SC. Using this technology enables fewer people

to make faster and better quality decisions on SC configurations from SC unit templates, under constraints networks with reduced variance. Implementation of this technology will enable realise increased quality, reduced cost, reduced errors, decreased personnel requirements, and better SC configuration solutions.

Future plans for research on this topic are to develop and experiment generic SC network configurations based on, (a) divergent problem solving strategies, (b) functional or operational policies of Members and / or Group, and (c) levels of co-operation among members of the SC.

Acknowledgements

Some components of the above approach have been developed for the "Demand Activated Manufacturing Architecture project for the United States Integrated Textile Complex", supported by a grant to the second author from the United States Department of Energy; and the project "Affordable Cost Structure", supported by a grant from Ford Motor Company to the first author.

References

Bond, A. H., and Gasser, L., eds. 1998. *Readings in Distributed Artificial Intelligence*. San Mateo, California: Morgan Kaufmann.

Chandra, C. 1997. Enterprise architectural framework for supply-chain integration. In Proceedings of the Sixth Annual Industrial Engineering Research Conference, 873-878. Norcross, Georgia: Institute of Industrial Engineers.

Delphigroup. 1998 www.delphigroup.com

Donkin, R. 1998. Disciplines complete for a newcomer. *Financial Times* October 28:14.

Gasser, L. 1991. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence* 47:107-138.

Giachetti, R. E., Young, R. E., Roggatz, A., Eversheim, W., and Perrone, G. 1997. A methodology for the reduction of imprecision in the engineering process. *European Journal of Operational Research* 100:277–292.

Hillier, F. S., and Lieberman, G. J. 1990. *Introduction to Operations Research*. New York, New York: McGraw-Hill Publishing Company.

Hirsch, B. 1995. Information System Concept for the Management of Distributed Production. *Computers in Industry* 26:229–241.

ISO TC 184/SC 5/WG 1. 1997. Requirements for enterprise reference architectures and methodologies, http://www.mel.nist.gov/sc5wg1/gera-std/ger-anxs.html

Lee, H. L., and Billington, C. 1993. Material management in decentralised supply chains. *Operations Research* 41(5): 835-847.

Moulin, B. and Chaib-Draa, B. 1996. *An overview of distributed artificial intelligence*. O'Hare, G. M. P., and Jennings, N. R. eds. New York, New York: John Wiley & Sons. 3-55.

Nadler, G. 1970. *Work Design: A Systems Concept*. Homewood, Illinois: Richard D. Irwin.

Neches, R.; Fikes, R. E.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; and Swartout, W. R. 1991. Enabling Technology for Knowledge Sharing. *AI Magazine* 12(3):16-36.

Patil, R. S.; Fikes, R. E.; Patel-Schneider, P. F.; MacKay, D.; Finin, T.; Gruber, T.; and Neches, R. 1992. The DARPA Knowledge Sharing Effort: Progress Report, In Proceedings of the Annual International Conference on Knowledge Representation. Cambridge, MA.

Poirier, C. C. 1999. *Advanced Supply Chain Management: How to build a sustained competitive advantage*. San Francisco, California: Barrett-Koehler Publishers, Inc.

Royer, J. 1992. A new set interpretation of the inheritance relation and its checking. *OOPS Messager* 3(3):22-40.

Sheremetov, L. B., and Smirnov, A. V. 1997. A Model of Distributed Constraint Satisfaction Problem and an Algorithm for Configuration Design. *Computación y Sistema* 2(1):91-100.

Smirnov, A. V. 1994. Conceptual Design for Manufacture in Concurrent Engineering. In Proceedings of the Concurrent Engineering: Research and Applications Conference, 461-466. Pittsburgh, Pennsylvania.

Smirnov, A. V., and Sheremetov, L. B.. 1998. Agent & Object-based Manufacturing Systems Re-engineering: a Case Study. In Proceedings of the Conference on Integration in Manufacturing, 369-378. Göteborg, Sweden: IOS Press.

Taha, H. A. *Operations Research: An Introduction*. 1987. New York, New York: Macmillan Publishing Company.

Tsang, J. P. 1991. Constraint Propagation Issues in Automated Design, In: *Expert Systems in Engineering: Principles and Applications*, Gettlob, G., and Nejdl, W., eds. 462:135-151. Berlin: Springer-Verlag.

Tzafestas, S., and Kapsiotis, G. 1994. Co-ordinated control of manufacturing/supply chains using multi-level techniques. *Computer Integrated Manufacturing Systems* 7(3):206-212.