Development Of A Message Model To Support Integrated Design And Manufacturing

Venkat Allada[1], Shaw Feng[2] and Steve Ray[2]
[1]Engineering Management Department
University of Missouri-Rolla
Rolla, MO 65409
[2]National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

Mere sharing of information between engineering design systems and manufacturing systems does not represent an ideal integrated system. While information sharing represents an important aspect of an integrated design and manufacturing environment, an equally critical aspect is the "interaction" capability of the two systems. This interaction could be in the form of feedback and request messages between the design and manufacturing systems. The goal of this study is to investigate the issues involved in the development of a conceptual message model that will facilitate an "upstream" and a "downstream" communication between the design activities and process planning activities. The development of a message model is an attempt to capture the design/process planner dialogue in a computer representable and interpretable form.

Keywords

Message model, Collaborative design and process planning

Introduction

Not too long ago, the sequential approach to product design and development was widely prevalent. In this approach, the designer would design a part and throw it "over the wall" for the manufacturing personnel to figure out how to manufacture it. This approach resulted in a vast number of design iterations before the artifact was finally ready for production release. Faced with intense global competition, manufacturing companies realized that they could no longer afford long design-to-market lead time and expensive product development costs. This situation has forced many manufacturing companies to embrace the design for manufacture (DFM) philosophy. The basic idea behind DFM is to address the manufacturability issues at the design stages.

The need to integrate the design and manufacturing domains has been recognized and addressed by many researchers in the past [1-4]. However, most integrated design and manufacturing systems cited in literature concentrate solely on the "data sharing" aspects of integration. While data sharing represents an important aspect of an integrated design and manufacturing environment, an equally critical aspect is the capability of the two domains to interact with each other. This interaction could be in the form of feedback and request messages to facilitate a "two-way" communication between the design activities and process planning activities. One of the earliest works along these lines was reported by Frost and Cutkosky [5]. They described a DFM approach (machining and shape-deposition processes) that has been implemented using agents, written in the Java language, which exchange feature-based capability models. The basic building block of intelligent interaction is knowledge sharing that includes (1) mutual understood knowledge and (2) the communication of that knowledge [6].

Message Model Roadmap

The roadmap for the development of a message model is depicted in Figure 1. The research issues that need to be addressed in order to reach the conceptual message model milestone include: (1) characterization of manufacturability evaluation, (2) development of a unified product-process ontology, and (3) development of a domain-specific message taxonomy.

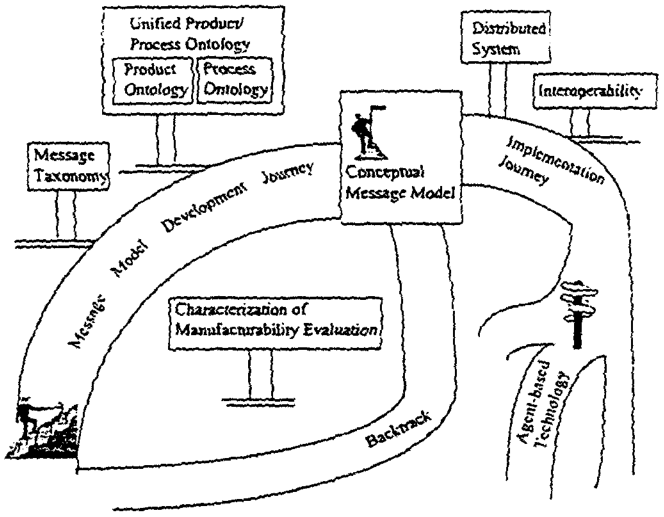

Fig. 1: Message model roadmap

This study attempts to disassociate these conceptual message model research issues from the actual implementation issues. We feel that once the foundation of a conceptual message model is laid, then one can concentrate on the implementation issues such as interoperability, distributed and collaborative environment, etc. We also feel that an agent-based approach is more suitable for the messaging application. Preliminary discussion on the suitability of the agent-based approach is presented at the end of this paper.

Characterization of Manufacturability Evaluation

It may be a worthwhile task to investigate what exactly is meant by *manufacturability* evaluation [7]. The characterization of manufacturability evaluation will serve as a basis to develop a unified product-process ontology for the design/process planning messaging application domain. The manufacturability evaluation coverage is depicted in Figure 2. The figure illustrates that at different levels of product abstraction (product-level, part-level, and feature-level), different manufacturability evaluation tasks can be conducted. In addition, it has to be noted that the coverage of manufacturability evaluation is dependent on the stage of a product engineering life-cycle (ranging from conceptual design to detailed process plan). For example, the tool path planning task cannot be conducted at the concept design stage.

Development of a Unified Product-Process Ontology

An ontology serves several purposes: (1) provides a shared understanding that several applications can jointly understand and use, (2) defines the semantics in a precise and unambiguous fashion, and (3) implements the semantics in a set of axioms that will facilitate automatic deduction to many "common sense" queries [8]. An ontology is an explicit specification of a conceptualization [9]. When the domain knowledge is represented using a declarative formalism, the set of represented objects is called the universe of discourse. This set of objects, and the relations among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge [9]. Gruber [10] discusses the design of ontologies for the purpose of enabling knowledge sharing and reuse. In that context, an ontology is a specification used for making ontological commitments. An ontological commitment is an agreement to use a vocabulary to ask queries and make assertions in a way that is consistent with respect to the theory specified by an ontology. Uschold and Gruninger [11] presented a skeletal methodology for building new ontologies.

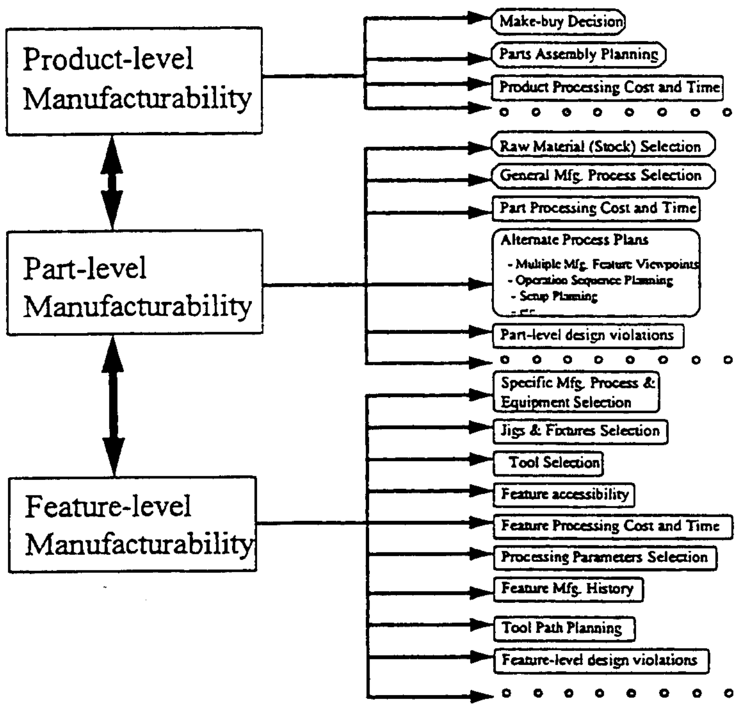
Product-level Manufacturability
Part-level Manufacturability
Feature-level Manufacturability
Make-buy Decision
Parts Assembly Planning
Product Processing Cost and Time
Raw Material (Stock) Selection
General Mfg. Process Selection
Part Processing Cost and Time
Alternate Process Plans
- Multiple Mfg. Feature Viewpoints
- Operation Sequence Planning
- Setup Planning
Part-level design violations
Specific Mfg. Process & Equipment Selection
Jigs & Fixtures Selection
Tool Selection
Feature accessibility
Feature Processing Cost and Time
Processing Parameters Selection
Feature Mfg. History
Tool Path Planning
Feature-level design violations

Fig. 2: Manufacturability evaluation tasks at various levels of product granularity

The product/process ontology should define the vocabulary with which queries and assertions are exchanged between the designer and the process planner. A product ontology should be able to describe the characteristics of the artifact as it is typically described by the designer such as: design features, dimensions, tolerances, design rationale, etc. Re-use of existing literature (such as the Standard for the Exchange of Product Model Data (STEP) documentation, existing ontologies on the Ontolingua server[1,2] , etc.) will greatly help in the development of a product ontology. A process ontology should describe the characteristics of the process employed for manufacturing the artifact as it is typically described by the process planner such as: manufacturing features, processing time and cost, process capability, etc. The existing literature (process reasoning study by Ray [12], manufacturing resource modeling study by Jurrens et al. [13], machining process planning activity model by Feng [14], etc.) should be consulted for building a process ontology. Intense research is necessary to develop a mapping mechanism for the unification of the product-process ontology.

Communication Protocols

Communication consists of three major components [6]: (1) interaction protocol; (2) transport protocol; and (3) communication language. Finin et al. [6] further describe these three components. The interaction protocols can range from complex negotiation schemes and game theory protocols to much simpler protocols between the software agents. The transport protocol is the actual transport mechanism for the

[1] Ontolingua is a set of tools, written in Common Lisp, for analyzing and translating ontologies. Further details can be obtained on the web page http://www-ksl.stanford.edu/kst/ontolingua.html

[2] For an introduction to the ideas underlying ontolingua, refer to an article by T. R. Gruber., A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2):199-220, 1993.

communication (e.g., Transmission Control Protocol (TCP), e-mail standards such as Simple Mail Transfer Protocol (SMTP), Hyper Text Protocol (HTTP), etc.). The communication language is a mode of expressing the *attitudes* regarding the content of the communication exchange [6].

The Common Object Request Broker Architecture[3] (CORBA) introduced by Object Management Group (OMG) attempts to address the issue of interoperability among the rapidly growing number of hardware and software products. The OMGs Object Request Broker (ORB) is the middleware that establishes the client-server relation between objects. A CORBA-based distributed object software system was developed to support Sandia's Agile Manufacturing Testbed [4](SAMT). Each of the heterogeneous physical objects such as the lathe, milling machine, robot arm, etc., has a corresponding object that supports a common Interface Definition Language (IDL) interface called an IDevice. This interface provides operations for material processing, material movement, status monitoring, and other administrative tasks. CORBA objects allow for the encapsulation of a machine tool, its controller, and the network interface to the controller. The SAMT architecture supports flow of information into manufacturing devices (e.g., control and Numerical Control (NC) code), and extraction of data (e.g., sensor data or inspection reports) back out of the machine and into the broader information processing environment.

The External Interfaces Working Group was formed as a part of the Defense Advanced Research Projects Agency (DARPA) Knowledge Sharing Effort, with an objective to develop protocols for the exchange of knowledge among autonomous systems [6]. The main result of this effort has been the development of the Knowledge Query and Manipulation Language (KQML).[5] KQML is both a message format and a message-handling protocol that uses a set of performatives (messages) that agents can use while communicating with each other. KQML-based experimental prototype systems have been applied in applications such as concurrent engineering, design and intelligent planning and scheduling. KQML has about two dozen reserved performatives that fall into seven basic categories as shown in Table 1 [15]. Readers are referred to Finin et al. [6], Mayfield et al. [15] for further details on the performatives listed in Table 1.

Table 1. KQML performatives

Category	**Performative Name**
Basic query	evaluate, ask-if, ask-about, ask-one, ask-all
Multi-response query	stream-about, stream-all, eos
Response	reply, sorry
Generic informational	tell, achieve, cancel, untell, unachieve
Generator	standby, ready, next, rest, discard, generator
Capability-definition	advertise, subscribe, monitor, import, export
Networking	register, unregister, forward, broadcast, route

Domain-specific Performatives

The KQML performatives can be viewed as a base set of performatives for the design/process planning messaging model. Apart from these base set KQML performatives, some domain-specific performatives must be devised in order to support the domain-specific communication. We are currently investigating the viability of a preliminary list of domain-specific performatives shown in Table 2.

Let us consider a scenario where a designer wants to know from the process planner the manufacturability of a feature "F." Depending on the nature of the problem, the process planner may make a number of alternative suggestions. These suggestions could vary in the degree of specificity depending on the how much product information is available to the process planner (assuming that as the product moves from the concept design stage to the final stage, more product information is available). At the

[3] Refer to web page http://www.omg.org

[4] Refer to http://nittany.ca.sandia.gov:8001/DistObjsMan.html

[5] Refer to http://www.cs.umbc.edu for additional details on KQML

concept stages, the process planner may prefer to use the performative "modify" to indicate the manipulation of feature "F." However, as more product information becomes available, the process planner may feel more confident to assert a specific feature modification performative such as "increase" or "decrease." It is to be noted that the domain-specific ontology (product and process ontology) will have to carry the concepts and terminology used by the domain-specific performatives. For example, a feature "hole" may be specified in the ontology having one of its attributes as say "length." The process planner will use both the domain-specific performatives and ontology to pass on the message to the designer. A sample message (KQML style) from the designer to the process planner inquiring about the manufacturing cost of an artifact can be as follows:

```
(calculate    :sender       Shaw
              :receiver     Steve
              :language     Prolog
              :ontology     product/process ontology
              :content      (Maufacturing_cost Artifact_1 ?mfgcost))
```

In this message, the performative is "calculate", the ontology assumed by the query is identified by the token "product/process ontology", the message is sent by an agent "Shaw" to an agent "Steve", and query is written in a language called "Prolog."

Table 2. Sample Design/Process Planning Domain Performatives

Category	Performative Name
Entity manipulation	modify, increase, decrease, move, translate, rotate, invert, disintegrate, merge, substitute, convert, reverse, relocate, mirror
Entity creation	create, apply
Entity deletion	delete
Evaluation analysis	effect-of, simulate, display, check, sort, calculate, compare
Entity History	rationale
Resources	generate-list, select-one, select-all
Documentation	help-on

Future Implementation Plans

While it is too early to decide the specifics of the implementation strategy, we feel that the agent-based technology holds good promise for the messaging application. A vast majority of engineering software applications offer "point solutions." This poses major integration problems. The concepts of modularity and interoperability of software programs have long been the ideals of the software engineering community (Khedro et al. 1994)[6] . However, the current software technology is still unable to support the goal of full automated interoperability. Recent research efforts on interoperable software agents have shown promise to significantly advance the software engineering technology [16]. Frost and Cutkosky [5] point out that the primary difference between the agents and distributed objects lies in the message structure employed for communication. They argue that agents provide a level of abstraction above distributed objects. While distributed objects communicate using method invocations, software agents interoperate with other software agents using the Agent Communication Language (ACL) such as KQML described earlier.

Conclusions

We feel that data sharing ability merely emphasizes the structured and syntactic aspects of information flowing between disparate domains; it is the "interaction" between disparate domains (such as design and process planning) that captures the true spirit of integration. A foundation for the development of a conceptual message model has been laid in this study. We hope that further investigation along these lines

[6] Reference is Khedro, T., Genesereth, M. R., and Teicholz, P. M., Concurrent Engineering through Interoperable Software Agents, http://www-leland.stanford.edu/group/CIFE/FCDA/CERA94.html

will eventually lead to a system that will facilitate near real-time "*dialogue*" between the designer and the process planner.

Acknowledgments

The first author wishes to acknowledge the financial support extended by the National Institute of Standards and Technology (NIST) to conduct this study. We wish to thank Prof. Tim Finin, Dr. Ram Sriram and Mr. Craig Schlenoff for their helpful suggestions/comments on KQML, design process and ontology respectively.

References

[1] Allada, V., and Anand, S., "Feature-based Modeling Approaches for Integrated Manufacturing: The State-of-the Art Survey and Future Research Directions." *International Journal of Computer Integrated Manufacturing,* Vol. 8, 1995, pp. 411-440.

[2] Allada, V., and Anand, S., "Machine Understanding of Manufacturing Features." *International Journal of Production Research,* Vol. 34, 1996, pp. 411-440.

[3] Chang, T. C., "Expert Process Planning for Manufacturing." Addison-Wesley Co. Inc., 1990.

[4] Ishii, K., "Life-cycle Engineering Design." *DFM Track: ASME, DE-Vol. 81,* 1995, pp. 39-45.

[5] Frost, H. R., and Cutkosky, M. R., "Design for Manufacturability via Agent Interaction." *ASME Design for Manufacturing Conference,* Irvine, CA, August 18-22, 1996.

[6] Finin, T., Labrou, Y., and Mayfield, J., "KQML as an Agent Communication Language." Book Chapter, *Software Agents,* Bradshaw, J., (Ed.), MIT Press, Cambridge, 1995.

[7] Gupta, S. K., Regli, W., Nau, D. S., "Integrating DFM with CAD through Design Critiquing." *Concurrent Engineering: Research and Applications,* Vol. 2, No. 2, 1996.

[8] Gruninger, M., Schlenoff, C., Knutilla, A., and Ray, S., "Using Process Requirements as the basis for the Creation and Evaluation of Process Ontologies for Enterprise Modeling." *ACM SIGGROUP Bulletin,* Vol. 18, No. 3, August 1997, pp. 52-55.

[9] Gruber, T. R., "A Translation Approach to Portable Ontologies." *Knowledge Acquisition,* Vol. 5, No. 2, pp. 199-220, 1993a.

[10] Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, 1993b.

[11] Uschold, M., Gruninger, M., "Ontologies: Principles, Methods, and Applications." *Knowledge Engineering Review,* Vol. 11, No. 2, June 1996.

[12] Ray, S. R., "Process Reasoning." *IFIPWG 5.7: Information Flow in Automated Manufacturing Systems. Computers in Industry,* Vol. 9, 1987, pp. 329-325.

[13] Jurrens, K. K., Fowler, J. E., and Algeo, M. E. A., "Modeling of Manufacturing Resource Information: Requirements Specification." NISTIR 5707, NIST, Gaithersburg, MD, 1995.

[14] Feng, S. C., "A Machining Process Planning Activity Model for Systems Integration." *NISTIR 5808,* National Institute of Standards and Technology, Gaithersburg, MD 20899.

[15] Mayfield, J., Labrou, Y., and Finin, T., "Evaluation of KQML as an Agent Communication Language." Intelligent Agents Vol. II - *Procc. of the 1995 Workshop on Agent Theories, Architectures, and Languages,* Wooldridge, M., Muller, J. P., and Tambe, M., (Eds.), Springer-Verlag, 1996.

[16] Genesereth, M. R., "An Agent-based Framework for Software Interoperability." *Procc. of DARPA Software Technology Conference,* 1992, pp. 359-366.