7

On the implementation of product model interfaces in CIME

T. Sørensen
Department of Control- and Engineering Design,
Building 421, Technical University of Denmark
DK-2800 Lyngby
Phone: +45 45 25 45 32
Fax: +45 45 88 40 24
E-mail: torben.soerensen@iks.dtu.dk

Abstract

This paper address how neutral product model interfaces can be identified, specified, and implemented to provide intelligent and flexible means for information management in manufacturing of discrete mechanical products.

The use of advanced computer based systems, such as CAD, CAE, CNC, and robotics, offers a potential for significant cost-savings and quality improvements in manufacturing of discrete mechanical products.

However, these systems are introduced into production as 'islands of automation' or 'islands of information', and to benefit from the said potential, the systems must be integrated into an integrated manufacturing unit. Such units are known as Computer Integrated Manufacturing and Engineering (CIME) systems.

The basic concept in CIME is to share and reuse information between the different computer based subsystems. Consequently, for the integration purposes, the CIME systems are highly dependent on reliable product model interfaces to bridge the information islands together.

The neutral product model interfaces addressed in this paper are used to facilitate an integration strategy know as 'the open systems CIME architecture'. This architecture is basically based upon three different domains; the CA(X) systems are placed in two different domains for design and planning, respectively. A third domain within the CIME architecture comprises the automated equipment on the shop floor.

Computer Applications in Production and Engineering. F. Plonka and G. Olling (Eds.)

Keywords
Neutral product model interface, STEP, CIME, information modeling, function modeling, schema-driven processors.

1 INTRODUCTION

The success of a manufacturing process is determined by the degree to which the manufacturing system masters the essential constituents of a product: Matter, energy, and *information*, (Schlechtendahl, 1989).

For the manufacturing of discrete mechanical products three different types of information are relevant. 1) *Product definition data*, that defines product properties such as geometric and kinetic features, and 2) *process definition data*, that define the process parameters such as welding speed, -current, and -voltage, and 3) *production definition data*, such as bill of materials, logistics and production management data.

The history of the manufacturing of discrete products is characterized by a continuous expansion of emphasis from matter via energy towards information, leading to the domain of *information technology*. One main reason for this is the fact that information technology and manufacturing automation have proven to be important instruments to reduce costs, increase product quality and to accelerate the process within an enterprise, especially within product development and preparation for manufacturing. Today, manufacturing of e.g., automobiles, airplanes, and ships is no longer cost-effective and competitive without modern information technology and automation.

The information technology domain has two important sub-areas: Information *processing*, and information *communication*, (Schlechtendahl, 1989).

Computer aided information processing systems, CA(X), such as CAD for design, CAR for robotics, CAQ for quality assurance etc., are prerequisites for the automated manufacturing of discrete products based upon programmable, general purpose, automatic shop-floor machines, such as robotic systems and NC machines. However, these information processing systems are inherently stand-alone systems and act inevitable as 'islands of information', or 'islands of automation' when they are introduced into production. What is needed is the other aspect of information technology, namely free communication of information for integrating or 'bridging' these 'islands' together by information exchange.

This development led to new manufacturing concepts such as Computer Integrated Manufacturing and Engineering (CIME). The main objective of a CIME system is to provide a reliable and rational manufacture of products. A CIME system integrates the information flow of product data that is communicated between the various CA(X) systems, or CIME subsystems, that feature the functional 'building blocks' of a computer based manufacturing system. The CIME area encompasses the whole range of computer integrated activities with CA(X) systems.

The basic concept in CIME systems is to *share* and *reuse* information between the different subsystems. In modern manufacturing technology a huge amount of product information must be created, identified, stored and transferred. The CIME concept therefore needs intelligent communication of product data back and forth between the sub-systems, inside the CIME system in question, as well as between sub-systems of other CIME systems, e.g., those at a sub-contractors site.

Often the individual subsystems are purchased from different vendors and have been developed over many years. The systems tend to use non-standard ways to represent product definition data which leads to integration problems due to the lack of inter-system communication and data exchange possibilities. The growing number of diverse sub-systems that appear on the CIME market place further amplifies this interconnection problem.

One problem in CIME is therefore to define an *integration strategy* sufficient to solve these inter-communication and data exchange problems between the various CA(X) systems.

Another problem is that the CIME architecture must provide an *infrastructure* for building individually tailored CIME sub-systems. This way, the CIME sub-systems can later be updated, expanded, added or excluded in accordance to the users needs and opportunities, in an evolutionary and progressive way, without ruining the overall architecture, (Trostmann, 1987).

A promising concept, used to solve these problems, is known as the *open systems'* architecture. Open systems CIME architecture can appropriately be implemented based upon *neutral interfaces.* In this context the term 'neutral' denotes that the data and program formats have been universal accepted as a carrier of data and programs, irrespective of native data and program formats used by the sending or receiving CIME sub-system.

2. THE OPEN SYSTEMS' CIME ARCHITECTURE

An open system CIME architecture is conceptually based upon three different domains as shown in Table 1; the CA(X) subsystems are placed in two different domains for design and planning, respectively. A third domain within the CIME architecture comprises the automated equipment for real-time execution of the manufacturing tasks on the shop floor. Neutral interfaces are defined between each domain to assure the free exchange of product definition data and programs, both between CIME-subsystems in different domains and between CIME-subsystems in the same domain. The product model interfaces are implemented as physical files, as databases, or as application programming interfaces by use of processor programs.

Table 1 An open CIME system kernel (Kroszynski, 1991)

Function	*CIME sub-system*	*Neutral Interface*
Domain 1: Product design, analysis and evaluation.	CAD / CAE	
		Product model description
Domain 2: Manufacturing and process planning, simulation, and programming.	CAM / CAP / CAR	
		EMC description (data & programs)
Domain 3: Explicit Machine Control (EMC) of automated manufacturing equipment, for real-time shop-floor manufacturing.	NC / CNC / DNC Robots/FMS/AGV	

The first neutral interface in Table 1 refers to the product model description interface, which should include all the relevant characteristics of the object to be produced i.e., the product-, the process-, and the production definition data mentioned above.

CAD/CAE systems usually generate a description of the product definition data, i.e. geometrical characteristics such as shape, dimensions, and topology, of each single component, subassembly, and entire machine part.

CAD/CAE/CAP application modules, or dedicated software packages and systems supplement the product definition data with, e.g., the kinematics that governs the moving parts of the product, as well as process-, and production definition data. The first neutral interface is associated to the acronyms IGES (IGES, 1991), SET (SET, 1989), VDA-FS (VDA-FS, 1987), CAD*I (CAD*I, 1989), and the international standard STEP (ISO 10303-1, 1993). STEP is the working title for 'ISO10303-STandard for the Exchange of Product model data' for communication and storage of information models represented as *neutral product definition data*, and *neutral program* formats.

The second neutral interface in Table 1 refers to the actual programs that drive the computer controlled manufacturing equipment on the shop floor. These programs, generated at the CAM/CAP/CAR level of the CIME system, can range from instructions to perform a drilling cycle to produce a pattern of holes, to a

complex program for the assembly of a mechanism by a robot system. Explicit Machine Control (EMC) descriptions for numerically controlled machine tools are realized with the standardized programming language 'NC-processor' (ISO 4342), which is based on the well-known APT language. The NC-processor languages produce two standardized output formats 'CL-data and structure' (ISO 3592), and 'CL-data for post-processing' (ISO 4343). For robotics, no robot programming languages have matured enough, politically and/or technically, to become an international accepted standard yet. Today, robot programming languages are at best, national standards such as the two German DIN standards IRDATA, (IRDATA, 1986), and IRL, (IRL, 1992).

Having shortly discussed the open systems' architecture, we now turn our attention to the development of the product model interface, which is addressed above as the 'first' neutral interface to implement this architecture.

2 DEVELOPMENT OF NEUTRAL PRODUCT MODEL INTERFACES

The development of a product description interface for exchange of advanced, discrete product models introduces some new and interesting challenges to be met.

Some of these challenges are to find appropriate *interface design-* and *implementation* methodologies, as well as *tools* for increasing the inter-system communication capabilities, and accordingly contribute with solutions to the above-mentioned integration problem in an intelligent fashion.

A proper solution of these interrelated challenges is a prerequisite for raising the integration level and thereby increasing the degree of *interoperability* in CIME. Interoperability in CIME reflects the capability of diverse CIME sub-systems to work together for the benefit of the total CIME system to achieve a desired goal.

The design phase, in Table 1 Domain 1, is distributed over 'traditional' computer Aided Design (CAD) and Computer Aided Engineering (CAE) systems such as Computer Aided Control Systems Design (CACSD) systems. This distributed design phase gives raise to *interoperability problems* because two different designs are often transferred concurrently as a compound model in order to fulfill primarily specific functional needs of the receiving CAM/CAP/CAR system in Table 1, Domain 2.

Therefore, the generic, enhanced product description interface has to be designed with the said open architecture and by use of proper methodologies to provide mechanisms to administer and integrate these distributed designs, reliably and efficiently.

Three 'classic' phases of work are used for interface development, i.e. the identification phase, the specification phase, and the implementation phase. Each of these phases has its own associated methodologies; the identification phase utilizes function modeling, the specification phase is based upon information models, and the implementation phase is based upon a computer programming language.

The choices of methodologies for interface development affects the scope (or capabilities) of the interface and consequently the application of the interface. It is therefore important to choose the best possible framework for the interface development.

It is, of course possible to develop dedicated processors by use of arbitrary processor development principles and home-made interface formats, but the scope and application of such processors and interfaces are quite limited.

A more appropriate approach is to base the open systems' CIME architecture upon neutral interfaces and corresponding neutral processor programs as mentioned above.

Today, the international STEP initiative (ISO-10303-1, 1993) provides a promising infrastructure to cope with function modeling and information modeling of product definition data for the development of neutral product model description interfaces, and it provides a common, universally accepted data and program format for information transfer.

The STEP methodology allows (among other) us to use the IDEF0, (IDEF0, 1981) function modeling technique for identification, and EXPRESS/EXPRESS-G (ISO-10303-11, 1994), information modeling techniques for specification as shown in Figure 1. For the interface implementation purposes one is free to use whatever programming language preferred (e.g., C++ is quite suitable for use with STEP).

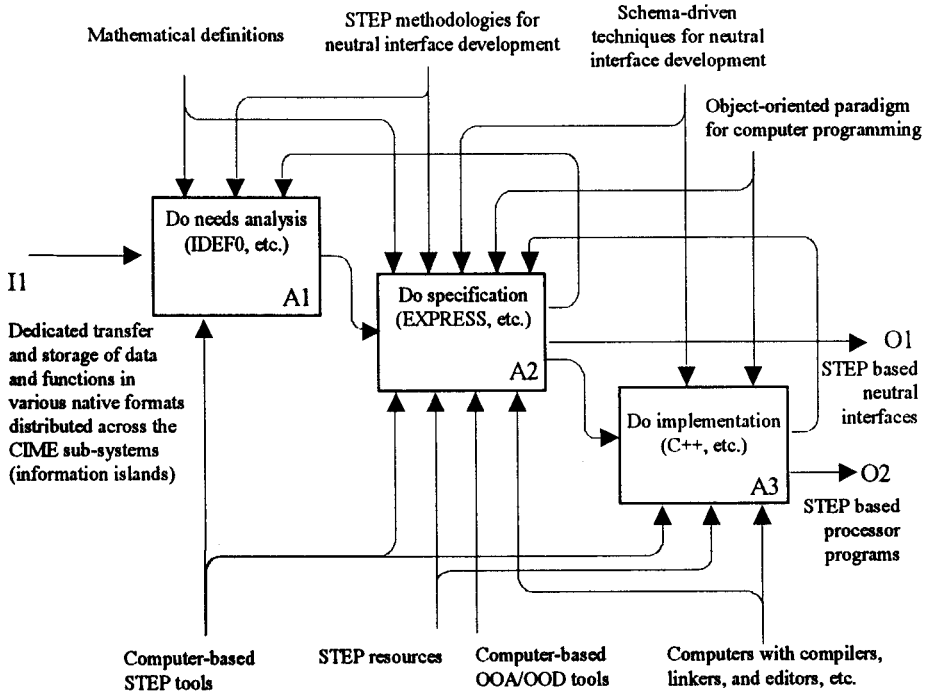

Figure 1 IDEF0 diagram showing the different phases of the iterative interface developments by use of the ISO-STEP methodology (Sørensen, 1996).

Generally speaking, the purpose of such initiatives as STEP is to enable *sharing of product model information between different computer applications*. For information to be shared between two systems, an agreement on the contents and the semantics must be made. In STEP this is accomplished by defining information models in EXPRESS, a formal information modeling language (based upon data identified by use of e.g. IDEF0 as shown in Figure 1). EXPRESS has been developed specifically to describe product information for exchange between engineering applications, and the semantics of an information model is determined partly by the EXPRESS text (schema), and partly through natural language annotations.

The role of a product information model for the definition of interfaces is shown in Figure 2.

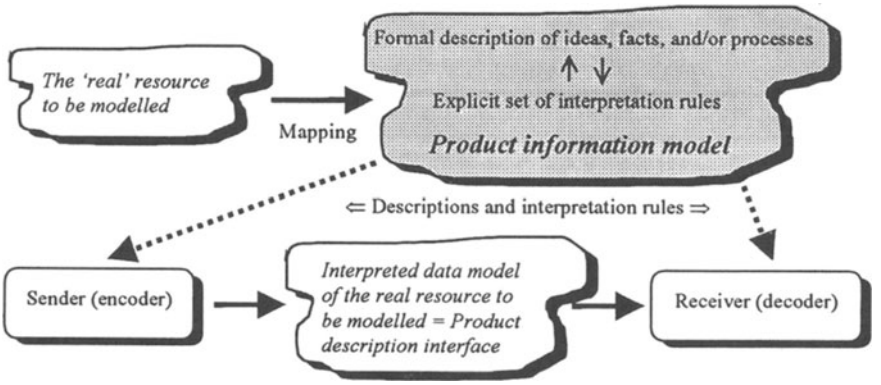

Figure 2 The role of an information model for interface definition for transfer and storage of information (Sørensen, 1996).

An information model represents the structure and semantics of information within the modeled system or subject area.

Information models for interface definition (Figure 2) 'maps' ideas, facts, and processes, related to the 'real' resources to be modeled, into formal interface descriptions defining the scope and constraints of the information resources. Moreover, information models utilize explicit sets of rules defining how data, which are subject to transfer via the interface, should be interpreted.

An information model can be defined as (Schenck, 1994):

An information model is a formal description of types of ideas, facts, and processes which together form a model of a portion of interest of the real world, and which provides an explicit set of interpretation rules. (If an information model

is written in EXPRESS or any other computer sensible representation, it has the additional quality of being computer processible).

An information model requires three interconnected concepts: *scope*, *perspective* and *purpose*. Scope means what is included and what is excluded from the model; in other words it defines the boundaries of the model. Perspective represents the viewpoints that are incorporated into the model. The model will be affected by the objectives of the modeller's perspectives, and thus the purpose of the model is also important; what is the model intended for?

3 IMPLEMENTATION OF PRODUCT MODEL INTERFACES

The last stage in the iterative development process of a neutral product model interface, addressed above, and shown in Figure 2, is the *physical implementation* of the interface. The neutral interface is implemented by use of *processors*. Processors, or translators, are computer programs that facilitate transfer, access, and storage of data within a neutral interface defined by the information model.

The implementation of the STEP based product model interface implies the selection of proper processor development techniques, implementation paradigm, CASE tools, and exchange scenario for processor implementation and tests.

A product model interface implementation has been done at the Technical University of Denmark (DTU) facilitating the exchange of dynamic control models between general purpose Computer Aided Control Systems Design (CACSD) Systems and Computer Aided Robotics (CAR) systems for dynamics simulation of robot control systems. This test set-up is using the so-called schema-driven processor development technique, the object-oriented implementation paradigm, and a fourth generation STEP based CASE tool.

Such STEP based processor development is an iterative modeling and programming process with functional modeling in IDEF0, information modeling in EXPRESS-G and EXPRESS (or NIAM or IDEF1X), and implementation specification with e.g. OOA/OOD, and implementation with OOP as show in Figure 3. The chronological modeling and development order goes from IDEF0 modeling to EXPRESS-G and EXPRESS, and further on to OOA/OOD followed by OOP.

One advantage with the interface implementation chosen at DTU is that the link between the EXPRESS schema and the data structure of the processor program is facilitated by a compiler (CASE tool) that translates the EXPRESS language into an intermediate data structure of the implementation language (typically C++). This feature promotes interoperability between the CIME sub-systems by allowing easy adoptions of new information models and easy modifications of the existing information models and thereby allowing distributed sub-designs to act together in the interface as a virtual complex design.

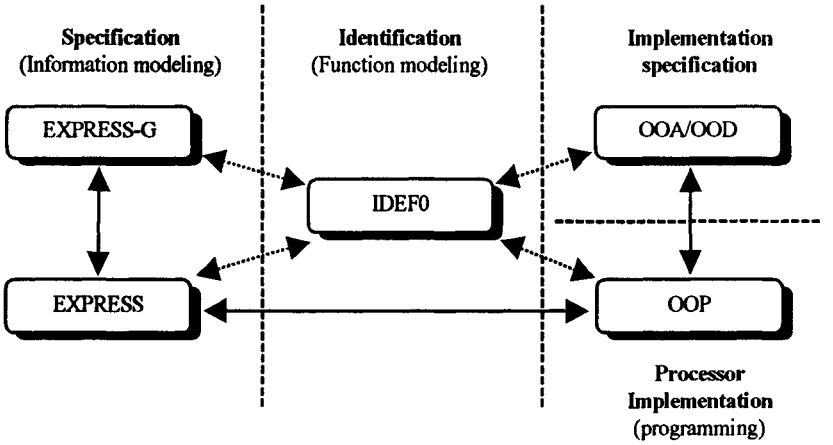

Figure 3: The iterative modeling process between lexical and graphical representations. Dotted arrows indicate manual model data transfer and full arrows indicate automatic conversion between data representation formats (Sørensen, 1996).

Note that for 'non-schema-driven' processors the EXPRESS schema is interpreted and translated *manually* into the intermediate data structure. This also implies that, for every change in the EXPRESS schema the intermediate data structure has to be exposed to a slow, error-prone, and tedious manual updating process.

Another advantage with the chosen approach is that it uses a 'modular' architecture for developing interfaces that reduces the overheads in creating neutral interfaces, such as those based upon STEP. Only the so-called pre-processor 'front-end', and the post processor 'back-end' are specific to a given CA(X) system, and the intermediate data structure is derived directly from the information model (represented here by the EXPRESS schema). In addition to this, the use of the modular architecture together with computer aided software engineering (CASE) tools, allow for the data to be manipulated and checked as a part of the translation process.

The creation of pre- and post processor programs for another system therefore requires development of only the 'front-end' and the 'back-end' elements. Advanced architectures for data exchange (here STEP based exchange) include not only the ability to create and maintain the internal data structures directly from the information model (here the EXPRESS schema), but also the creation and mappings between different (EXPRESS) data models.

4 CONCLUDING REMARKS

In this paper we have seen how neutral product model interfaces can be identified, specified, and implemented to provide intelligent and flexible means for information management in manufacturing of discrete mechanical products.

The international STEP initiative provides an infrastructure for the identification, specification, and implementation of neutral interfaces based upon the open systems' CIME architecture.

The generality and flexibility of the open systems' architecture, based upon neutral interfaces and implemented by processor programs, is strongly affected by the choice of the processor development techniques, the implementation paradigm, the CASE tools, and the exchange scenario for processor implementation and tests.

A product model interface implementation has been done at The Technical University of Denmark (DTU) facilitating the exchange of dynamic control models between general purpose CACSD Systems and CAR systems for dynamics simulation of robot control systems. This test set-up is successfully using the schema-driven processor development technique, the object-oriented implementation paradigm, and a fourth generation STEP based CASE tool. The set-up is very flexible and general in the sense that it is easy to update with new product model features directly based upon EXPRESS specifications, modular, and relatively easy to maintain.

5 REFERENCES

(CAD*I, 1989) 'CAD Data Transfer for Solid Models', E. G. Schlechtendahl (ed.), ESPRIT Research Reports, Vol. 3, CAD Interfaces (CAD*I), Springer Verlag, Heidelberg, D, 1989.

(ISO 10303-11, 1994) 'ISO IS 10303-11: Product Data Representation and Exchange: Description Methods: The EXPRESS Language Manual', ISO IS, 19 August 1994.

(IDEF0, 1981) 'ICAM Architecture, Part II, Volume IV - Function Modeling Manual (IDEF0),' Report number AFAWL-TR-81-4023 (U.S. Air Force Wright Aeronautical Laboratories, Wright-Patterson Air Force Base, OH, June 1981).

(IGES, 1991) 'The Initial Graphics Exchange Specification (IGES) Version 5.1' Iges/Pdes Organization, NIST, Gaithersburg. MD 20899, USA.

(IRDATA, 1986) 'IRDATA, VDI 2863, Blatt 1, Entwurf', VDI - Verlag GmbH, Dusseldorf, 1986.

(IRL, 1992) 'Industrial Robot Language (IRL)', DIN 66312 Teil 1, Deutsches Institut für Normung, Beuth Verlag, Berlin, 1992.

(ISO 10303-1, 1993) 'Part 1: Overview and Fundamental Principles', Industrial Automation Systems - Product Data Representation and Exchange, ISO/TC 184/SC4, 1993.

(Kroszynski, 1991) 'Driving Robots via Neutral Interfaces', U. Kroszynski, T. Sørensen, T. G. Clausen and E. Trostmann, Proceedings of the Annual ESPRIT Conference, Brussels, 25-29 November 1991, pp. 646-660.

(Schenck, 1994) 'Information modeling the EXPRESS way', Schenck, D., Wilson, P., Oxford University Press, New York, Oxford, 1994.

(Schlechtendahl, 1989) 'Intelligent communication of product definition data', E. G. Schlechtendahl, 11th World Computer Congress, IFIP '89, San Francisco. Pre-print pp. 1-5.

(Sørensen, 1996) 'Interoperability of CAD Standards and Robotics in CIME', T. Sørensen, Ph.D. Dissertation. Department of Control and Engineering Design, Technical University of Denmark. June 1996. ISBN 87-90130-08-1.

(Trostmann, 1987) 'Cad Data Interfaces for Robot Control', E. Trostmann, Paper for the 2. Duisburger Kolloquium, Automation und Robotik, 15 - 17 July 1987, Universität Duisburg.

(VDA-FS, 1987) 'VDA Surface Interface Version 2.0', Verband der Automobil-industrie e.V (VDA), W-6000 Frankfurt am Main, Westendstrasse 61, Germany, 1987.

6 BIOGRAPHY

Assistant professor, Ph.D. Torben Sørensen
received his M.Sc. degree in electrical engineering from the Technical University of Denmark (DTU) in 1986, and his Ph.D. degree in Mechanical Engineering from DTU in 1996. In 1986 he joined the Control Engineering Institute of DTU where he was appointed as a research fellow, and later as a senior engineer in the CAD and Robotics areas. Since 1996 he has been an assistant professor at the Department of Control and Engineering Design of DTU.

Dr. Sørensen has been supervisor, co-supervisor, and external reviewer for several M.Sc. students in the CAD and Robotics areas during the last seven-eight years. Currently, he teaches CAD, Robotics, and basic Control Engineering at Masters' level.

He has participated as Senior Researcher and Group Leader in large international projects, and is currently engaged with research in the areas of robot control and CAD.

He has published several papers and co-authored two books in the above fields of interest.