

Model Sharing among Agents in a Concurrent Product Development Team

Masanori Ozawa*

Systems & Software Research Laboratories
Research and Development Center
Toshiba Corporation
70, Yanagi-cho, Saiwai-ku, Kawasaki-shi, 210 Japan
ozawa@ssel.toshiba.co.jp

Mark R. Cutkosky

Brian J. Howley
Center for Design Research
Stanford University

560 Panama Street Stanford. CA 94305-2232
cutkosky, bhowley@cdr.stanford.edu

Abstract

We describe a project involving the use of software agents for concurrent engineering of electromechanical products. The agents are a mix of autonomous programs and software interfaces for human specialists in an engineering team. Our goal is to accelerate the design process by reducing delays associated with the exchange of design information, including models, constraints, objectives, and dependencies associated with each agent. Part of this effort involves methods for representing and exchanging design information in a machine-readable form.

In this paper we focus on the sharing of information in models, such as the lumped-parameter models of electromechanical systems that engineers often use in the early stages of design. We first review the design information-sharing problem and examine the different levels of abstraction and detail at which such information is represented and shared. We then introduce a representation language and information-sharing infrastructure that we have employed for an agent-based concurrent engineering system. We illustrate some of the issues involved in our approach with a short scenario involving the redesign of an optical pick-up head for DVD (digital versatile disc) players. We then discuss our plans for extending the system to allow more general exchanges of engineering models among agents.

1. Introduction

One of the major challenges associated with the design of modern consumer products is the need to integrate mechanical, electrical, optical and software components in a compact, low-cost system. The design of such devices as DVD and CDROM players typically involves a team of specialists who exchange models, preferences, decisions and constraints as the design progresses. Delays are introduced when such information is not transferred efficiently among human specialists and among the engineering programs they use.

A way to address this problem is to represent the engineering team as a set of agents that adhere to a common communication language and protocol. The agents can be autonomous programs that perform a specific function or software interfaces through which human specialists exchange information between their own engineering tools and those of the rest of the team. Our work on agent-based engineering teams has involved two efforts: the development of software for coordinating the actions of agents by dynamically tracking the dependencies among their decisions, tasks and goals, and the development of methods for capturing and exchanging the information that agents need to share in a machine-readable form. The focus of this paper is the second effort; the agent coordination problem is discussed in other publications [Petrie, Webster and Cutkosky, 1995].

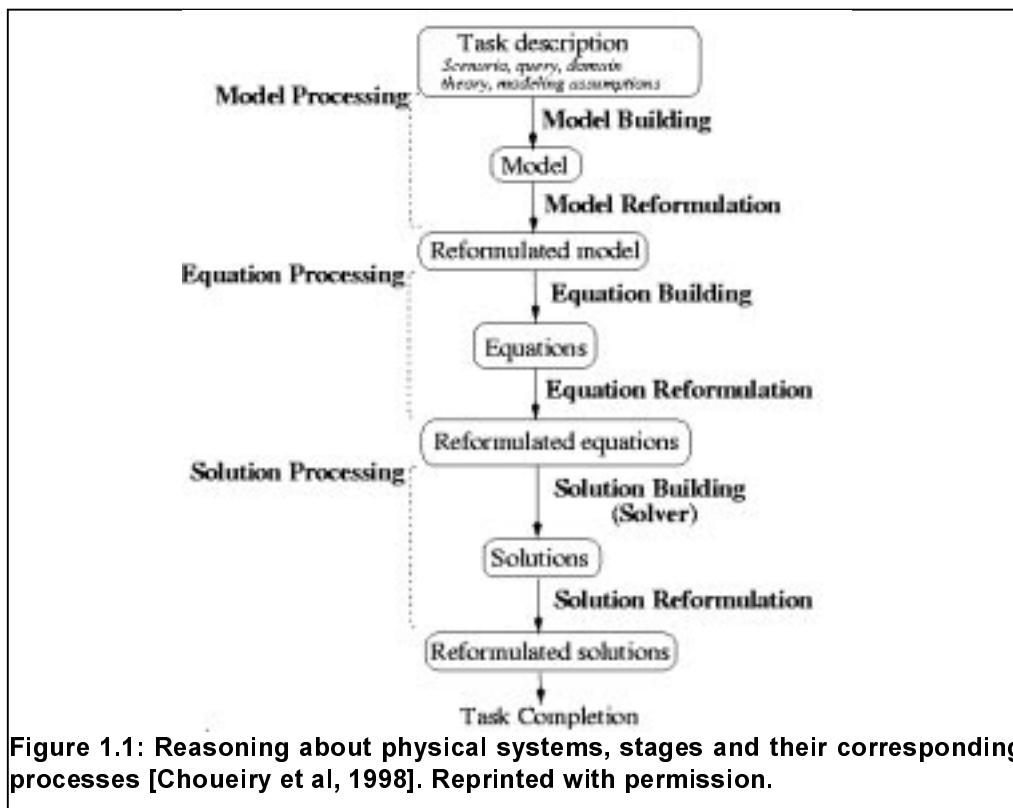
Our approach to representing and sharing design information among agents is a specialized subset of the Knowledge Sharing effort in the Artificial Intelligence community [Olsen et al, 1995, Iwasaki et al 1997, Falkenhainer et al, 1994]. The

* Visiting scholar, Center for Design Research, Stanford University

sharing of information among agents is supported by a common *ontology* [Gruber, T. R., 1993] (a set of terms and definitions in the product domain) and a compositional modeling language (CML) in which models can be created such that conventions, constraints, applicability conditions and assumptions are defined formally and ultimately grounded in logic.

1.1 Levels of information sharing in design

The sharing of design information can take place at a variety of levels, ranging from detailed numerical representations, such as FEM mesh for a component, to abstract principles (e.g., of energy conservation in a circuit). Choueiry et al [1998] have proposed a "theory of model reformation for reasoning about physical systems" which provides a useful framework for examining the different levels at which engineering information is used and exchanged. They describe the process of composing and analyzing models as a sequence of steps, as shown in Figure 1.1. The process begins with a *task description*, including an underlying *domain theory* (representation of the relevant physical laws, properties of components, etc.), *modeling assumptions* (for example, that the temperature will remain within some range) and a *scenario* that captures the particular conditions that hold during the task.



The steps shown in Figure 1 can also be applied to engineering analyses. Given a task description, the first step is to compose models appropriate for the task. A model at this point often consists of knowledge of the physical structure (components and their topology, for example) as well as knowledge of the relevant physical phenomena, including the conditions under which they are active. The models may also be transformed, or reformulated, to improve their suitability for addressing the task. For example, a continuous model may be discretized. The next step is to generate equations from the models. The equations may also be reformulated or transformed for ease of analysis and solution. For example, the equations may be transformed from cartesian to polar coordinates or from time domain to frequency domain. The final step is solution processing, which may involve analytic or numerical solution. The solution may also be reformulated the better to answer particular questions.

In an engineering team, there will be multiple specialists or agents going through steps like those in Figure 1.1, and each stage will typically involve sharing information. Table 1 is an attempt to distinguish among the main levels of detail and their associated uses. At the highest level are representations used primarily in the conceptual design stages for communication among human experts. These models are often difficult to encode in a machine-understandable form. At the next level are

ontologies and information such as assumptions and constraints associated with the model-building and model-reformation stages of analysis. Generation of machine-interpretable models and ontologies is challenging and time consuming, but can be accomplished using research tools like CML [Falkenhainer et al, 1994, Bobrow et al, 1996], to be discussed in the next section. The next level in the table involves equations. Automatic generation and manipulation of equations is possible with commercial tools like Mathematica™¹, which is used as the "equation processor" in the CML environment. The most common levels of information sharing for integrated engineering tools are levels (d) and (e), exchanging model parameters and numerical data sets.

Table 1: Taxonomy of knowledge representation levels in design

Knowledge representation level	Tool, Means	Processing stage
a: Conceptual Model- Market needs, Specifications	QFD, TRIZ	Task description
b: Ontologies, Model Fragments, and Constraint set	CML Models	Task description /Model Processing
c: Mathematical Equations = Intermediate, sharable model	Mathematica, Design Sheet, CDME(1)	Equation Processing
d: Parameters = Low level, sharable model	MATLAB, 3D-CAD (I-DEAS), CDME(2)	Task Description
e: Product Model - 3D-CAE models or FEM models	I-DEAS, Pro-E, STEP etc.	Task Description

Although levels (d), (c) and (b) are increasingly abstract and difficult to exchange among automated agents, they are also increasingly flexible and adaptable for early design stages when the form of a design may be changing dramatically. The hypothesis guiding our work is that by exchanging models, starting at the earliest design stages, it will be possible to execute more design tasks in parallel, as shown in Figure 1.2, minimizing delays and backtracking caused by conflicts and misunderstandings. We call this early-stage evaluation "Performance Sizing" [Ozawa, Iwasaki and Cutkosky 1998].

In the next sections we describe briefly the approach we have taken for sharing such information among agents in a concurrent engineering system. We start with a brief description of CML and how it is used.

2. Compositional Modeling Language

In this section we briefly describe CML and its characteristics that make it suitable for developing engineering models for early-stage design. For a more detailed discussion see [Ozawa, Iwasaki and Cutkosky 1998].

Compositional modeling is a paradigm for formulating a behavior model of a physical system by composing descriptions of symbolic and mathematical properties of individual system components. CML is a general-purpose declarative modeling language for representing physical knowledge required for compositional modeling. CML is intended to facilitate model sharing between research groups, many of which have long been using similar languages. These languages are based primarily on the language originally defined by Qualitative Process Theory [Forbus 1984] and include the languages used for the Qualitative Physics Compiler [Farquhar

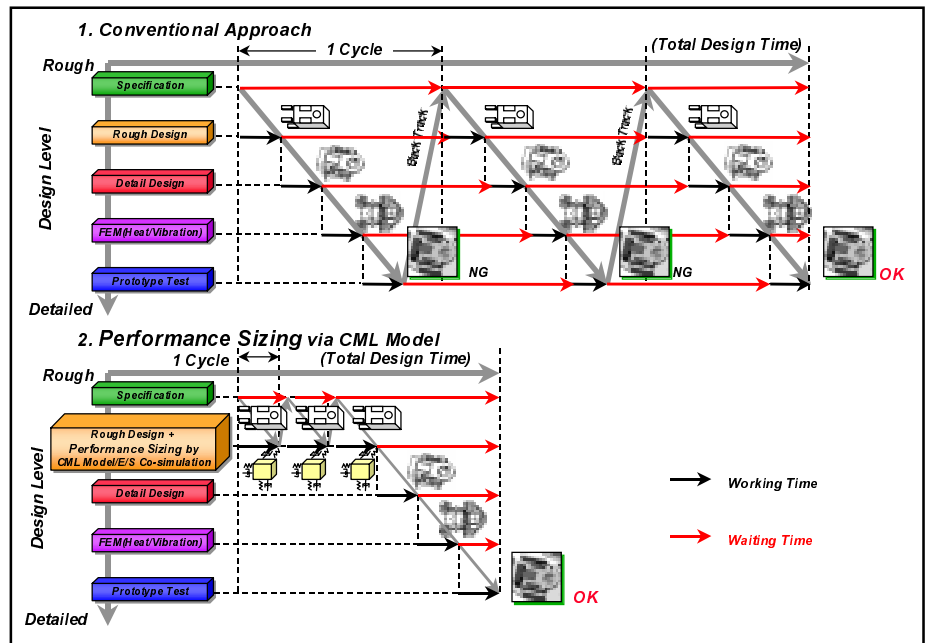


Figure 1.2: Reducing delays in the early ("Performance Sizing") stages of design through better sharing of information among agents.

¹ <http://www.wri.com>

1994], compositional model formulation [Falkenhainer 1991], and the Device Modeling Environment [Low and Iwasaki 1993]. CML is fully translatable to the knowledge interchange format (KIF)[Genesereth and Fikes 1992].

A domain theory in CML is a finite set of the following top-level forms:

- **defEntity** for defining properties of persistent objects (e.g., resistors, containers).
- **defModelFragment** for describing the behavior of modeled entities under explicitly specified conditions. Model fragments are used to describe phenomena that arise out of the interactions of a composite set of objects (e.g., collisions or flows), or the behavior of a single object (e.g., a resistor, pump, or valve).
- **defScenario** for defining initial value problems consisting of a set of objects, their configuration, and initial values for the quantities that describe them.

To predict the behavior of a physical system in some domain, knowledge about the physics of the domain is captured in a general purpose *domain theory* that describes classes of relevant objects, phenomena and systems. A domain theory in CML consists of a set of definitions, called *model fragments* and *entities*, each of which describes a piece of the domain's physics or objects. Once the domain theory has been constructed, it can be used to model different physical devices under a variety of conditions. The description of a specific system or situation being modeled is called a *scenario*. The scenario includes an initial configuration of the device, the initial values of some of the parameters that are relevant to modeling it, and perhaps conditions that further characterize the system. The CML implementation automatically identifies those model fragments that are applicable to the scenario. These model fragments are composed into a single model that comprises both a symbolic description and a set of governing equations. The equations may be solved or simulated to produce a behavioral description. Because the conditions under which the model fragments hold are stated explicitly in the domain theory, the system is able to assemble new models that describe the device as it moves into new operating regions. The full specification of the languages can be found in [Falkenhainer et al. 1994].

CDME is a web-based interface through which users can compose and interact with models in CML [Iwasaki et al, 1997]. If users define the domain theory in CML, and initial conditions or simulation conditions, etc. in a scenario, CDME will automatically interpret those definitions, convert them into an internal logical model, and prepare a procedure for numerical calculation. Currently, "Mathematica™" is the solver used in CDME. In addition, CDME can extract and generate equations from the CML model for use with an external solver or simulator.

Interaction between CML models and scenarios and external agents is accomplished using the Open Knowledge Base Connectivity (OKBC) protocol from the Stanford Knowledge Systems Laboratory [Chaudhri et al, 1997]. OKBC provides access to the classes, individuals, slots, facets, etc., of the CML library. OKBC is based on the Generic Frame Protocol (GFP) [Karp, Myers, Gruber, 1995].

3. Logical Description of Virtual PUH Model in CML

To investigate the sharing of models among engineering agents, we used CML to build models of the pick-up head (PUH) of a DVD optical disk device. The models capture several different domains of physical behavior, and the interactions between them. A CAD drawing of the PUH and its major components is shown in Figure 3.1. Figure 3.2 shows some of the CML models for the PUH, including rigid body and heat transfer dynamics. The dynamics include nonlinear changes in magnetic field strength as a function of actuator displacement and changes in ambient temperature.

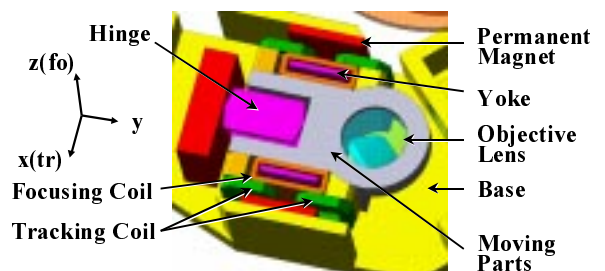


Figure 3.1: Structure of PUH

3.1 6-DOF Rigid Body Dynamics

The moving part of the PUH can be approximated as a rigid body with six degrees of freedom, suspended by a combination of linear and rotational stiffness and damping elements. Voice coil motors actuate the body in two directions to achieve focus and maintain tracking. In Figure 3.1, the focus direction is along the z axis and the tracking direction is along the x axis.

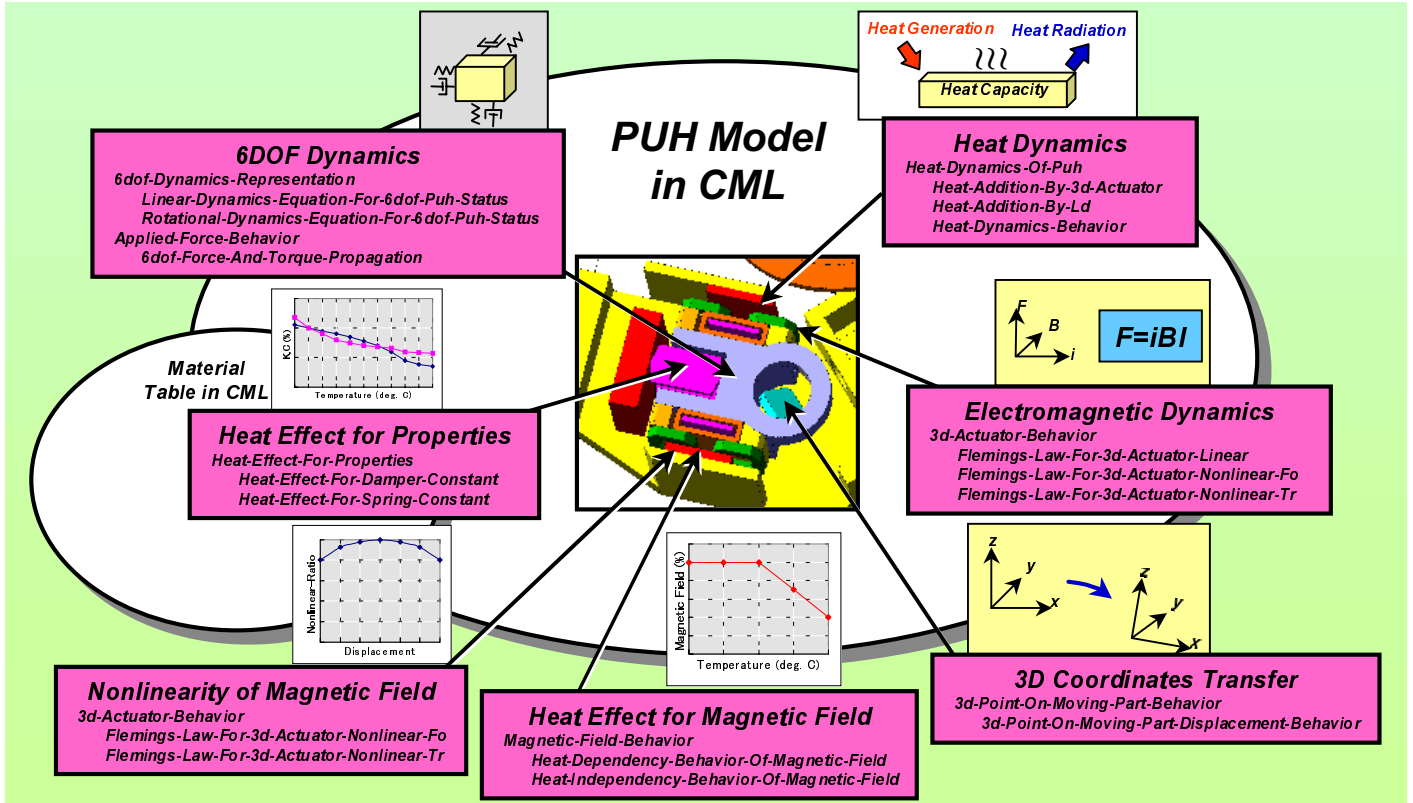


Figure 3.2: PUH Model in CML

The actuation force, F , is governed by Fleming's law ($F=iB_{j_{mt}}r_{md}l$), where l is the effective length of the coil in the magnetic field. The strength of the magnetic field varies as a function of temperature and actuator displacement. These dependencies are captured by the parameters r_{mt} and r_{md} , which are further described in Sections 3.3 and 3.4. A detailed description of the CML model fragments is given in [Ozawa, Iwaasaki and Cutkosky 1998].

3.2 Temperature Dependency of Material

High-polymer materials are often used in the flexure (hinges) of the PUH to provide controlled motion of the lens with respect to the base. The stiffness and damping parameters of these materials are often temperature-dependent. The changes in stiffness and damping must be accounted for when designing the servo system. The heat-dependent stiffness (k) and damping (c) are expressed by equations (3.3.1 – 3.3.3).

$$k(t_{PUH}) = k_{20} \cdot e^{\{-c_k(t_{PUH}-20)\}} \quad (3.2.1)$$

$$c(t_{PUH}) \propto \frac{\sqrt{m \cdot k(t_{PUH})} \cdot \tan \delta}{\sqrt{1 + \tan^2 \delta}} \quad (3.2.2)$$

$$\tan \delta = c_{c1} \cdot e^{(c_{c2} t_{PUH})} \quad (3.2.3)$$

K_{20} ≡ spring constant at 20 (c°)

c_k, c_{c1}, c_{c2} ≡ constants that are peculiar to each high-polymer material

3.3 Temperature Dependency of Magnetic Field

Magnetic fields of permanent magnets are also affected by temperature. Although the sensitivity is not high at normal room temperatures, it may be a concern in automotive applications where the ambient temperatures around the PUH exceed

80° C and the surface temperatures of actuator coils can exceed 120° C.

An approximate expression of the temperature dependency is given by equation (3.3.1).

$$\begin{cases} r_{mt} = 1.0 & (t_{PUH} \leq t_{cr}) \\ r_{mt} = 1.0 - c_{mt} \cdot (t_{PUH} - t_{cr}) & (t_{PUH} > t_{cr}) \end{cases} \quad (3.3.1)$$

In this equation, c_{mt} and t_{cr} are constants that are peculiar to each magnetic material, c_{mt} is the slope of the curve shown in Figure 3.2 and t_{cr} is the critical temperature at which the magnetic field starts to be weakened by heat.

For certain materials, a more accurate model may be available in the form of a look-up table or a function. Such models can be incorporated using the "Blackbox Function" of CML. Explicit conditions can be imposed on the applicability of such blackbox models.

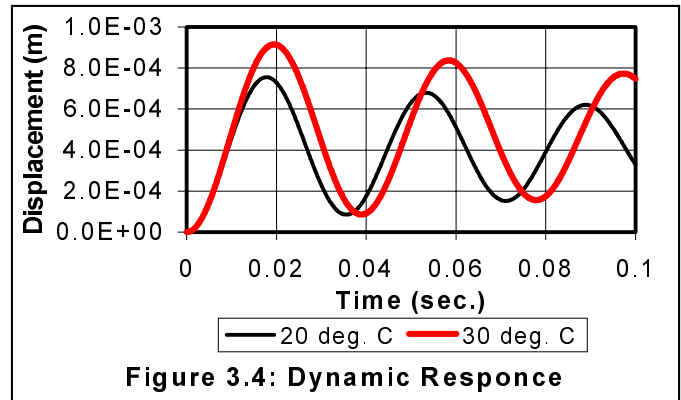
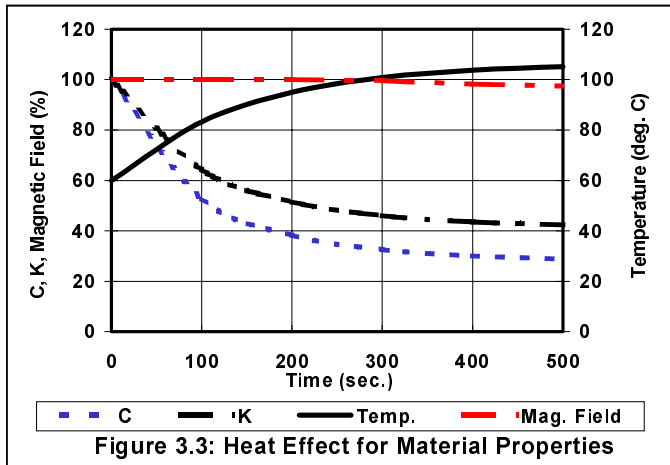
3.4 Non-Linearity of Magnetic Field

Ideally, the field produced by the permanent magnet should be uniform over the range of motion of the actuator coils to obtain maximum servoing accuracy. However, the field usually weakens somewhat near the edges. This effect is captured by the ratio, $r_{m_{ds}}$, where the full strength of the field corresponds to a value of $r_{m_{ds}}=1.0$ at the center of the field.

A typical curve of $r_{m_{ds}}$ is illustrated in the lower left corner of Figure 3.2. This information is entered into CML in the form of a table.

3.5 Simulation Results

Co-simulation results of PUH heat and rigid body dynamics are shown in Figures 3.3 and 3.4. The results were obtained using CDME. Figure 3.3 shows the change in material and magnetic properties over a change in PUH temperature. Figure 3.4 compares PUH dynamic response at two different temperatures. The simulation captures physical interactions that are normally characterized by separate design disciplines.



4. CML/MATLAB/I-DEAS Integration

Although cross-disciplinary simulations can be performed entirely within CML/CDME, we do not anticipate that this will be the usual approach. As discussed in Section 2, CML is an object-oriented, declarative language that emphasizes expressiveness and re-use of models and model fragments. It is not optimized for numerical efficiency. Moreover, we believe that most engineers will prefer to use their own specialized tools for computationally intensive analyses and simulations. The role of CML/CDME is to provide models that these engineers can interact with (publish, view, query, and refine).

4.1 Agent Infrastructure

To support interaction with CML, we have developed an agent-based framework as shown in Figure 4.1. This approach is based on the agent-interaction technology described in Cutkosky *et al* [1993]. The engineers and their tools interact through agents using the emerging standard communications language, KQML (see <http://www.cs.umbc.edu/kqml/> for the current KQML standard). The communications are built on open Internet standards, TCP/IP, SMTP, and FTP.

To facilitate the wrapping of commercial engineering tools we have developed the Java Agent Template (JAT). A description of JAT is beyond the scope of this paper, but details on the beta-release of JAT can be found at <http://java.stanford.edu>. Briefly, JAT is a package of programs written in the Java language that allow users to quickly create new software "agents" that communicate over the Internet. JAT facilitates especially construction of agents that send and receive messages in KQML and provides services including name registration, queuing and buffering of messages, connect/disconnect and security. Agents can be stand-alone programs or applets, downloaded through a standard web browser. Interfaces have been created for programs written in C++ and Lisp as well as Java.

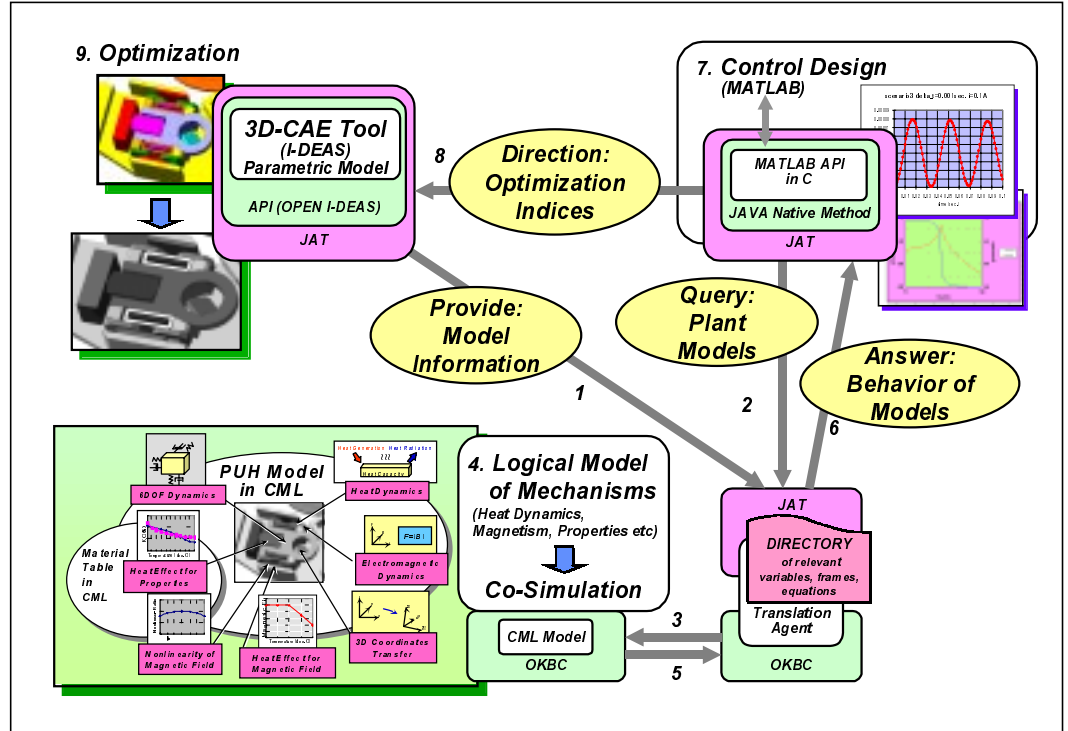


Figure 4.1 Agent Implementation and Model Sharing Example

4.2 Agent Interaction Example

Figure 4.1 shows an example exchange that we have implemented with three agents: a Geometry Optimization agent, which is an interface for a human designer using the I-DEAS^{TM2} CAD package, a Control Design agent, which is an interface to a servo designer using the Matlab^{TM3} controls toolbox, and a Logical Model agent, which maintains the CML models of the PUH domain and design scenario. The Logical Model agent uses OKBC and a Java translation agent to communicate with other agents. The Translation Agent converts between the (verbose) internal CML variable names and shorter task-specific names used by the other agents in the scenario. Table 2 is a partial listing of these terms.

The interaction illustrated in Figure 4.1 has the following sequence:

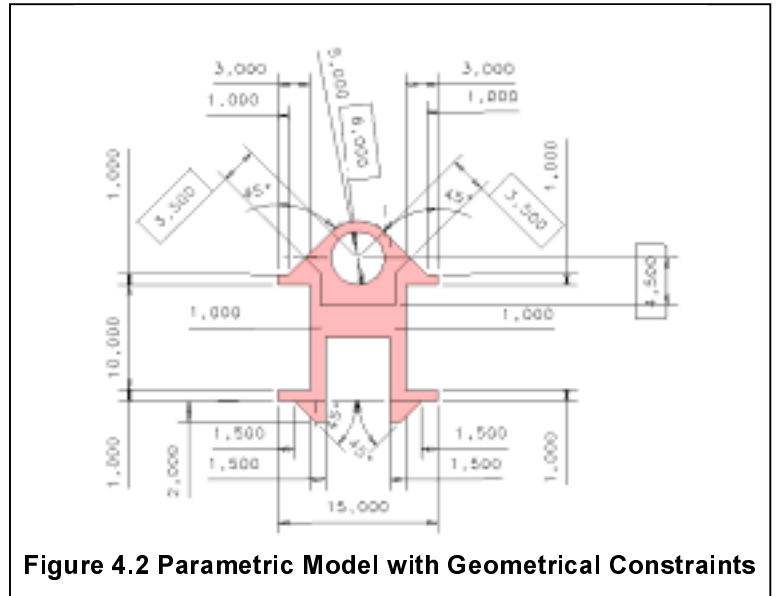


Figure 4.2 Parametric Model with Geometrical Constraints

² I-DEASTM, Open I-DEASTM: <http://www.sdrc.com>

³ <http://www.mathworks.com>

```

MrMatlab query to MechaAgent:
(ask-one
 :sender      MrMatlab
 :receiver    MechaAgent
 :language    KIF
 :content     (get-initial-values :values Mx))

MechaAgent response:
(reply
 :sender      MechaAgent
 :receiver    MrMatlab
 :language    KIF
 :content     (initial-values Mx 0.002))

```

Figure 4.3: Example message exchange between control design (“MrMatlab”) and CML (“MechaAgent”) agents.

- 1) The Geometry Optimization agent uses the CAD tool to create an initial PUH design as a collection of parametric models. (Figure 4.2 shows a parametric model of the PUH "bobbin" in more detail). Parameter values describing the properties of the PUH (e.g., overall mass, moments of inertia, stiffness) are transmitted to the CML model via the Translation Agent.
- 2) The Control Design agent requests the parameters of a PUH dynamic model, valid over a range of temperatures. A sample request is shown in Figure 4.3.
- 3) The requests are translated and submitted as a set of queries to CML
- 4) CML composes a model of the PUH dynamics and computes changes to the spring constant, damping, and magnetic field strength parameters as a function of temperature,
- 5) CML returns the parameters,
- 6) which are translated and passed back to the Control Design agent as KQML messages (see Figure 4.3)
- 7) The servo designer analyzes the plant response and determines the need to reduce PUH mass.
- 8) The Control Design agent sends a message to the Geometry Optimization agent requesting a reduction in the moving mass.

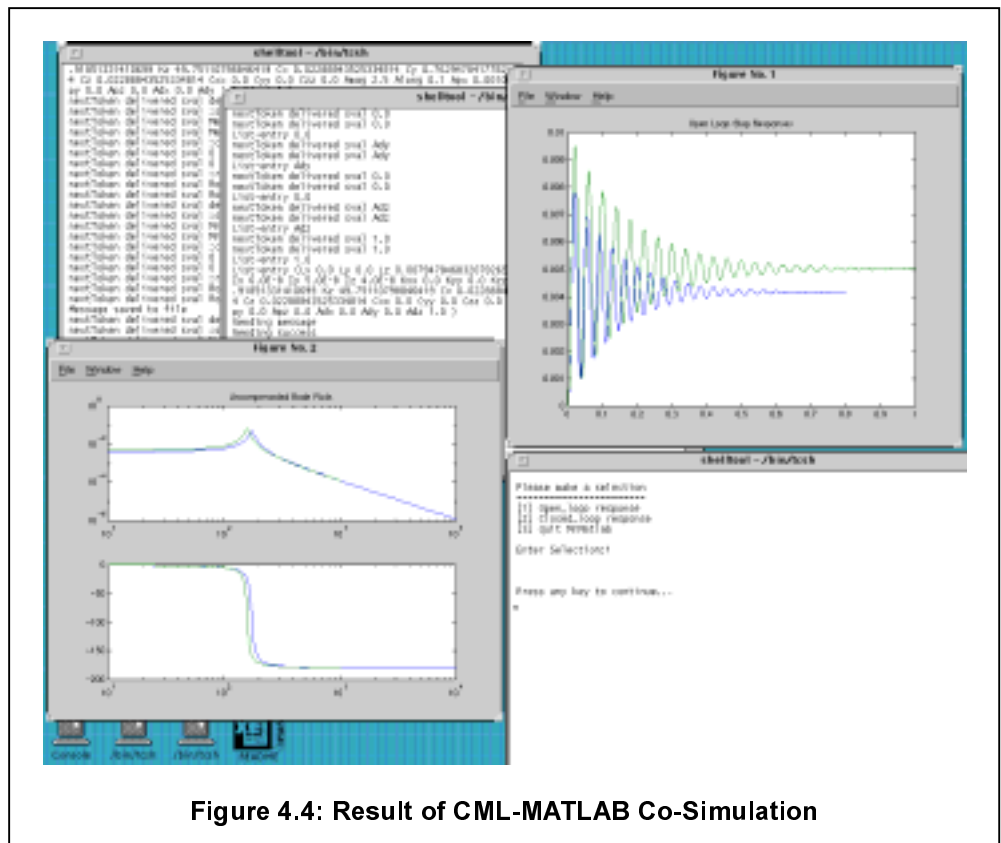


Figure 4.4: Result of CML-MATLAB Co-Simulation

A sample KQML message exchange between Control Design agent ("MrMatlab") and the CML

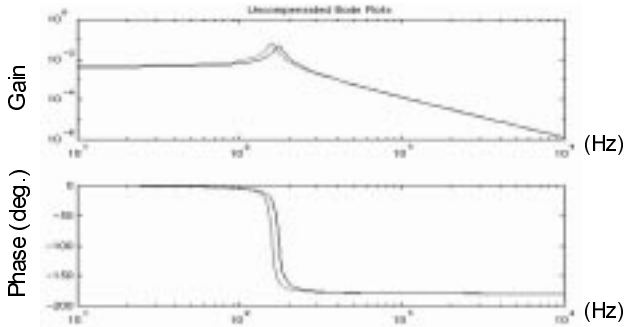


Figure 4.5: Open Loop Response

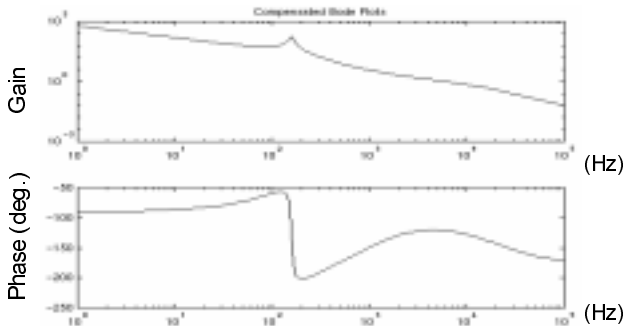


Figure 4.6: Closed Loop Response with Servo Agent

agent ("MechaAgent") is shown in Figure 4.3. The query contains the performative "ask one" and specifies the intended recipient, and the contents of the question

The "get-initial-values" and the model parameter "Mx" are part of a shared ontology between the control design and CML software agents.

The content expression is written in Knowledge Interchange Format, (KIF). The translation agent converts the expression "Mx" to the internal CML label "M-X", "The-3D-Mass," using table 2.

Figure 4.4 is the screen shot of the CML/MATLAB Integration System.

Figures 4.5 and 4.6 show the open and closed-loop Bode plots for the PUH. Recalling the temperature dependence of the magnetic field, the two different curves in Figure 4.5 show the effects of two different temperatures (20 c° and 30 c°, respectively).

Figure 4.6 shows that with the reduced moving mass, despite the variation in open-loop response, the closed loop response is insensitive to temperature change (the two curves are indistinguishable in Figure 4.6).

5. Discussion

In this section we discuss some of the issues that arose during our development of the system used for the example exchange in Figure 4.1.

5.1 Levels of the Information Sharing

It is useful to reconsider Table 1 in light of the simple example described in the last section. The example involved sharing of information at the parameter level (d) in the table. The sharing just of parameters could have been accomplished with agents that did not use CML models but instead used a common domain model and table of terms, such as in Table 2. In addition, sharing of models at the level of numerical data could be accomplished using commercial technology. Indeed, it is common to exchange 3D geometry models between CAD systems such as I-DEAS and FEM solvers such as Nastran™ and Abacus™

In fact, our view is that the main point of creating models in CML is to publish them so that others, outside of one's own area of expertise, can assimilate them. As an example, consider what it takes for a control engineer to incorporate models of laser optics and actuators into the controller of a DVD device. Before she can use the models, she needs to understand the terminology associated with models (what parameters, physical units, reference frames, etc. are involved) and what underlying domain theory the models are built upon. In addition, there may be limits to the applicability of the models, or additional constraints affecting some of the model parameters, which must be understood before the models can be used correctly. The more explicitly and formally these terms, conventions and limitations are defined, the more likely that non-experts will be able to use the models and the less chance there is for accidental misuse.

The extreme case of writing models for non-experts is to develop models that can be assimilated by software agents, defined in this context as programs that automate some elements of an engineering task. An example might be a controller design agent that specifies the parameters of a linear controller in response to stated objectives and a description of the system to be controlled. In this case it is necessary to develop machine-interpretable representations of the following:

- Definitions of all terms used in the model, especially those related to inputs and outputs - for example, whether the torque output of a motor is in Newton meters or gram centimeters and whether the position of a servomechanism is assumed to be a continuous function or discretized.

- The underlying knowledge in which a model is grounded - for example, how a model of robot dynamics is grounded in rigid body dynamics, kinematics, coordinate frames, vectors, etc.
- Conditions under which the model(s) are valid and conditions under which to change from one model to another - for example, that the field strength of a pair of magnets is constant below 100° C, but varies as an inverse function of temperature above 100° C.

These requirements are what motivates our work with CML.

Returning to Figure 1.1, we also note that equation *reformation* may be needed between agents. For example, the dynamics models as represented in CML are in a general ordinary differential equation format that is not particularly suited for efficient solution, nor in the usual state-variable form ($\dot{x} = Ax + Bu$) used in controls.

5.2 Models for sharing versus models for internal use

The 3D-CAD agent has the task of modifying the geometry of the parts to meet functional and physical requirements. The modification of part geometry requires expertise and is best done by a human engineer using a parametric CAD system. The parametric CAD model (see Figure 4.2) involves constraint equations that relate the various geometric parameters. While it would certainly be possible to express these constraints in CML, and accomplish parametric model updating via communication between the CAD agent and CML, this would be a cumbersome approach. The point here is that since geometry modification requires human expertise, and since the model used for accomplishing the modifications is encoded efficiently within the 3D CAD system, we should not treat it as a model to be published and possibly assimilated by other agents. On the other hand, complicated dependencies or trade-off problems involving multiple agents should probably be modeled in CML and solved either using the CDME environment, or exported as equations to be used with a solver such as Design Sheet [Rebby, Fertig and Smith, 1998] that is specialized for trade-off studies.

6. Conclusions and future work

In our current implementation, parameters are shared among CML, MATLAB, and I-DEAS (level d in Table 1). Model parameters are partially listed in Table 2. Under the model assumptions shared by these agents the parameters in Table 2 provide a complete description of the PUH dynamics. But without a set of shared assumptions, parametric information is not sufficient. For example parametric values alone do not guarantee a consistent set of units, nor do they define a reference coordinate system. Without making model assumptions explicit, values for inertia moments could be reasonably interpreted to be either about the PUH mass center or flexure.

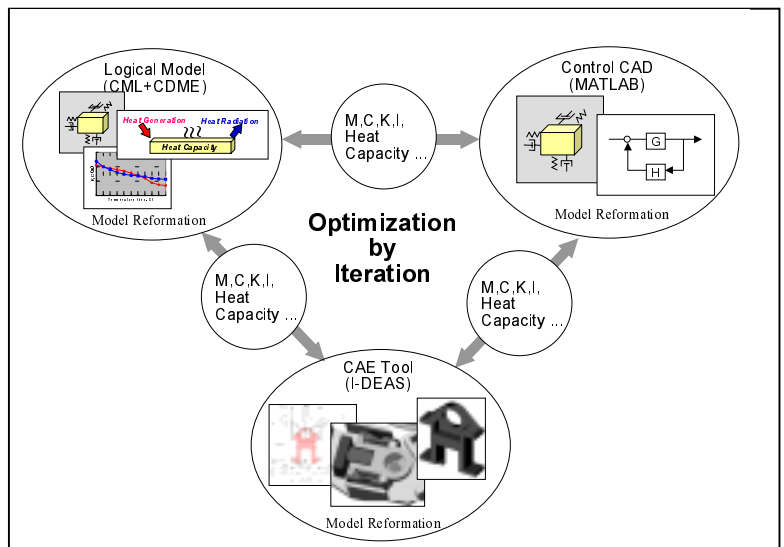


Figure 5.1: Model Sharing at the Parameter Level

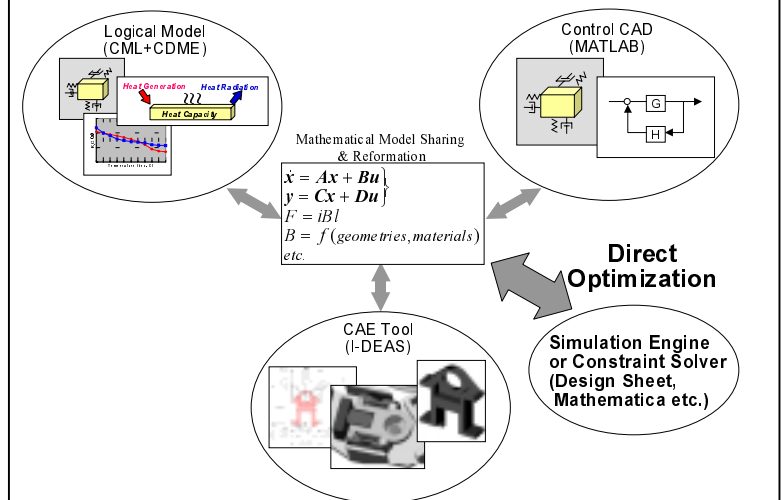


Figure 5.2: Model Sharing at the Mathematical Model Level

Table 2: Partial Listing of Terminology Translation Table

Matlab Agent name	MechaAgent name (CML cross reference)	Description
Mx	"M-X", "The-3D-Mass"	Effective PUH mass in X direction
Kx	"K-X", "The-K-Matrix"	Linear spring constant along the X direction
Cx	"C-X", "The-C-Matrix"	Linear damping coefficient along the X direction
Ix	"I-Xx", "The-3d-Inertia-Tensor"	PUH X axis inertia through mass center
Krx	"Kr-X", "The-K-Matrix"	Torsional spring constant about the X axis
Crx	"Cr-X", "The-C-Matrix"	Torsional damping coefficient about the X axis
l_track	"Effective-Length", "The-3d-Actuator-Tr"	Tracking actuator coil length
b_track	"Magnetic-Induction", "The-3d-Actuator-Tr"	Tracking actuator air gap flux density
s_track	"Directional-Sine-Against-Magnetic-Field", "The-3d-Actuator-Tr"	Tracking actuator coil winding perpendicularity
dx_track	"Direction-Cosine-For-X", "(Direction-Vector-For-3d-Applied-Force(Generated-Force The-3d-Actuator-Tr))"	X axis component of tracking actuator's direction of applied force
ax_track	"X-Coordinate", "(Location-For-3d-Applied-Force(Generated-Force The-3d-Actuator-Tr))"	X component of the tracking actuator's point of application
cgx	"X-Coordinate", "The-Center-Of-Gravity"	X component of the PUH center of mass
Lx	"X-Coordinate", "The-Focal-Point"	X component of the PUH center of focus

In future work, we intend to support information sharing at the mathematical model level (see Figure 5.2). Information sharing at the mathematical model level reduces the model reformulation task, and increases flexibility because models are no longer constrained to a pre-defined format. For instance, the temperature dependency of the damping coefficient can be readily expressed at the equation level. Additionally, model information will be augmented with applicability conditions to make modeling assumptions explicit. For example, the spring-mass-damper model of the PUH dynamics is only valid over a range of frequencies, beyond which PUH flexible modes become significant. Applicability conditions will help ensure models are used correctly and reduce design errors.

We are also eager to extend our concurrent system to address other engineering functions. Namely, we are trying to integrate the control logic model for the tracking, focusing and seeking controllers with our PUH model in the example. In this phase, we expect hybrid (continuous/discrete) modeling and simulation to be an important key technology to overcome the differences of granularity and behaviors of each simulation model.

Acknowledgments

The authors would like to thank the past and the present members of the Super Design Technology: Toshiba-Stanford project and Sheila McIlraith for their advice and assistance in this work. We gratefully acknowledge the contributions of Yumi Iwasaki, Tony Looser, James Rice, Robert Engelmores, and Richard Fikes from the Stanford Knowledge Systems Lab for their help with CML, CDME and OKBC. We also acknowledge the help of HeeCheol Jeon, Neeraj Shodhan and Liang Chu with the Java Agent Template/OKBC interface and Matlab API integration.

The work at CDR has been supported by Toshiba Corporation and DARPA under Navy contract N00014096-1-0679.

References

Ozawa, M., Iwasaki, Y., Cutkosky, M., R., "Multi Disciplinary Early Performance Evaluation via Logical Description of Mechanisms: DVD Pick Up Head Example" Proc. of ASME Design Engineering Technical Conference and Computers in Engineering Conference, 1998, Atlanta, GA.

Berthe Y. Choueiry, Sheila McIlraith, Yumi Iwasaki, Tony Looser, Todd Neller, Robert S. Engelmores and Richard Fikes, "Thoughts Towards a Practical Theory of Reformulation for Reasoning About Physical Systems", Working notes of the Symposium on Abstraction, Reformulation, and Approximation (SARA'98), 1998, Pacific Grove, CA, USA, pp. 25-36.

C.J. Petrie, T. Webster and M.R. Cutkosky, "Using Pareto Optimality to Coordinate Distributed Agents," AIEDAM Vol. 9, 1995, pp. 269-281

Antonio Diaz-Calderon, Christiaan J. J. Paredis, Pradeep Khosla, "A Modular Composable Software Architecture for the Simulation of the Mechatronic Systems", Proc. of ASME Design Engineering Technical Conference and Computers in

Engineering Conference, 1998, Atlanta, GA.

Francis Phang, Seokhoon Bae, David Wallace, "A Web-based Collaborative Design Modeling Environment", IEEE 7th Intl. Workshops on Enabling Technologies; Infrastructure for Collaborative Enterprise, 1998 (WET ICE '98: <http://www.cerc.wvu.edu/WETICE/>), Stanford, CA.

Plamen I. Bliznakov, Jami J. Shah, Susan D. Urban, "Integration Infrastructure to Support Concurrency and Collaboration in Engineering Design" Proc. of ASME Design Engineering Technical Conference and Computers in Engineering Conference, 1996, Irvine CA

Forbus, K. D., (1984). "Qualitative Process Theory." Artificial Intelligence, 1984, vol. 24, pp. 85-168.

Kiriyama, T., Tomiyama, T., and Yoshikawa, H., "Building a physical feature database for integrated modeling in design" In The Sixth International Workshop on Qualitative Reasoning about Physical Systems, 1992, pp. 124-138.

Bobrow, D., Falkenhainer, B., Farquhar, A., Fikes, R., Forbus, K., Gruber, T., Iwasaki, Y., Kuipers, B., "A Compositional Modeling Language", 96 AAAI Technical Report WS-96-01, 1996, Fallen Leaf Lake, CA.

Cutkosky, M. R., Englemore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Mark, W. S., Tenenbaum, J. M. and Weber, J. C. "PACT: An Experiment in Integrating Concurrent Engineering Systems" IEEE Computer, 1993, Vol. January pp. 28-38.

Reddy, S. Y., Fertig, K. W., Smith, D. E. "Constraint Management Methodology for Conceptual Design Tradeoff Studies" Proc. of ASME Design Engineering Technical Conference and Computers in Engineering Conference, 1998, Sacramento, CA.

Falkenhainer, B., Farquhar, A., Bobrow, D., Fikes, R., Forbus, K., Gruber, T., Iwasaki, Y., Kuipers, B., "CML: A Compositional Modeling Language", Technical report KSL-94-16, 1994, Knowledge Systems Laboratory, Stanford University.

Farquhar, A., "A Qualitative Physics Compiler", Proc., Twelfth National Conference on Artificial Intelligence, 1994, Seattle, Washington, The AAAI Press/The MIT Press.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., and Rice, J. P., "Open Knowledge Base Connectivity 2.0", Artificial Intelligence Center, SRI International, 21 July 1997.

Karp, P. D., Myers, K. L., Gruber, T., "The Generic Frame Protocol", Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, 1995, Montreal, pp768-774.

Genesereth, M. R., Fikes, R. E., "Knowledge Interchange Format, Version 3.0 Reference Manual", Technical report Logic-92-1, Stanford University Logic Group, 1992.

Iwasaki, Y., Farquhar, A., Fikes, R., and Rice, J., "A web-based compositional modeling systems for sharing of physical knowledge", Proc. of the 15th International Joint Conference on Artificial Intelligence, AAAI Press/The MIT Press, August 1997.

Olsen, G., R., Cutkosky, M., Tenenbaum, J. M., and Gruber, T., R., "Collaborative Engineering Based on Knowledge Sharing Agreement", CONCURRENT ENGINEERING, June 1995, Vol. 3, No. 2, pp. 145 – 159.

Gruber, T. R. (1993). "Toward principles for the design of ontologies used for knowledge sharing", International Journal of Human-Computer Studies, special issue on Formal Ontology in Conceptual Analysis and Knowledge Representation, August 1993 (Available as technical report KSL-93-04, Knowledge Systems Laboratory, Stanford University).

Tom R. Gruber and Pierre O. Gautier, Machinegenerated Explanations of Engineering Models: a Compositional Modeling Approach. In Proc. of the 13th IJCAI, pages 1502-1508, Chamb'ery, France, 1993.