Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2005), *19*, 1–18. Printed in the USA.

DOI: 10.1017/S0890060405050018

Ontologies for supporting engineering analysis models

IAN R. GROSSE,[1] JOHN M. MILTON–BENOIT,[2] AND JACK C. WILEDEN[3]

[1]Department of Mechanical and Industrial Engineering, 160 Governor's Drive, University of Massachusetts, Amherst, Massachusetts 01003-2210, USA
[2]United Technologies Research Center, East Hartford, Connecticut, USA
[3]Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003-2210, USA

(RECEIVED May 28, 2003; ACCEPTED August 5, 2004)

Abstract

In this paper we lay the foundations for exchanging, adapting, and interoperating engineering analysis models (EAMs). Our primary foundation is based upon the concept that engineering analysis models are knowledge-based abstractions of physical systems, and therefore knowledge sharing is the key to exchanging, adapting, and interoperating EAMs within or across organizations. To enable robust knowledge sharing, we propose a formal set of ontologies for classifying analysis modeling knowledge. To this end, the fundamental concepts that form the basis of all engineering analysis models are identified, described, and typed for implementation into a computational environment. This generic engineering analysis modeling ontology is extended to include distinct analysis subclasses. We discuss extension of the generic engineering analysis modeling class for two common analysis subclasses: continuum-based finite element models and lumped parameter or discrete analysis models. To illustrate how formal ontologies of engineering analysis modeling knowledge might facilitate knowledge exchange and improve reuse, adaptability, and interoperability of analysis models, we have developed a prototype engineering analysis modeling knowledge base, called ON-TEAM, based on our proposed ontologies. An industrial application is used to instantiate the ON-TEAM knowledge base and illustrate how such a system might improve the ability of organizations to efficiently exchange, adapt, and interoperate analysis models within a computer-based engineering environment. We have chosen Java as our implementation language for ON-TEAM so that we can fully exploit object-oriented technology, such as object inspection and the use of metaclasses and metaobjects, to operate on the knowledge base to perform a variety of tasks, such as knowledge inspection, editing, maintenance, model diagnosis, customized report generation of analysis models, model selection, automated customization of the knowledge interface based on the user expertise level, and interoperability assessment of distinct analysis models.

Keywords: Engineering Analysis Models; Interoperability; Knowledge Base; Ontologies

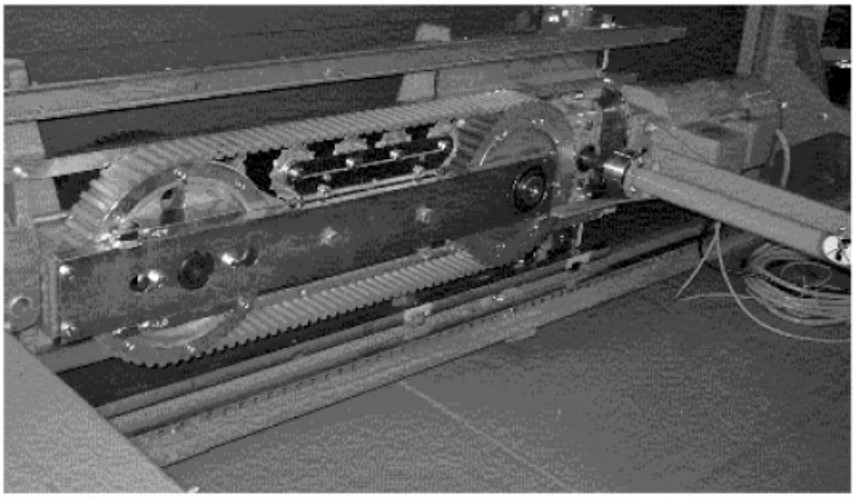
1. INTRODUCTION

The design of virtually all engineered products requires the services of engineering analyses to predict the behavior of the product and/or its manufacturing processes for evaluation and optimization of the product design in terms of its intended function, reliability, quality, manufacturability, and so forth. In the course of development of these analysis models of physical systems, engineers make numerous modeling idealizations or assumptions. Indeed, modeling idealizations and their associated justification are at the heart of all engineering analysis models. Which modeling idealizations are invoked depends on a variety of factors, such as the goals of the engineering analysis, the specific analysis objectives, analysis time constraints, the risk preferences of the analysis engineer to uncertainty in the analysis results, estimates of analysis cost, accuracy, and resolution, and observations regarding the behavior of the physical system (Doraiswamy et al., 1999). Further, this modeling knowledge is dynamic in nature. Analysis goals and objectives, time and cost constraints, and other related modeling information change as the product development cycle proceeds. As a result, analysis models that are appropriate or even optimal at one point in the product development cycle are inadequate at other phases of the product development cycle. For example, early in the product development cycle engineering analysis may consist of quick back-of-the-envelope analysis, yielding only order of magnitude estimates, while

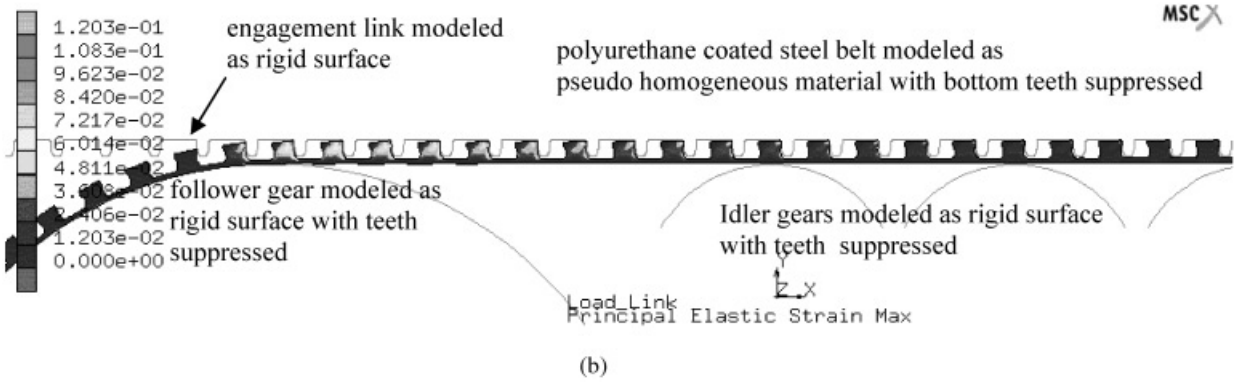
Reprint requests to: Ian R. Grosse, Department of Mechanical and Industrial Engineering, 160 Governor's Drive, University of Massachusetts, Amherst, MA 01003-2210, USA. E-mail: grosse@ecs.umass.edu

in the later stages of the development cycle detailed finite element analysis (FEA) may be required to obtain sufficiently accurate analysis results.

When engineers develop analysis models, they apply a mixture of domain modeling knowledge and general modeling principles to create the analysis model abstraction. Consider the NextStep™ (registered trademark of United Technologies Corporation, Hartford, CT) escalator drive unit module shown in Figure 1a and a corresponding FEA model in Figure 1b developed by an engineer at United Technologies Research Center. This analysis model is a 2-dimensional (2-D) plain–strain idealization of the system with a number of other important modeling assumptions. For example, all the metal components interacting with the flexible drive belt (drive pulley, follower pulley, idler rollers, and engagement link) are modeled as rigid surfaces; the bottom teeth on the belt, as well as on the drive pulley and idler rollers, are suppressed, and the belt itself is modeled as a pseudohomogeneous single material part, although in reality the belt consists of a number of steel cords embedded in polyurethane (PU). This pseudohomogeneous belt has a modulus of elasticity and belt thickness such that its bending and longitudinal stiffnesses are equivalent in a macroscopic sense to that of the actual belt. Despite these idealizations, this model simulates with sufficient accuracy the engagement of the top teeth of the cogged belt with the metal link (which connects to the escalator steps and drives the escalator) and the resulting strains and stresses induced in the upper teeth of the belt. Obviously, a host of modeling idealizations has been made in the process of creating this engineering analysis model. Further, underlying this modeling knowledge is mathematical and physical knowledge related to this physical phenomenon (i.e., the continuum–mechanics boundary value problem that models this physical system behavior) and numerical knowledge involved in the solution of this boundary value problem (i.e., finite element model). If we consider the wealth of modeling knowledge that is involved in developing analysis models that are used to predict combustion and fluid flow in a jet aircraft engine, the aerodynamic performance of entire planes, defor-



(a)



(b)

Fig. 1. (a) The NextStep™ escalator drive unit module and (b) 2-D finite element analysis model of the NextStep™ escalator model showing the distribution of the maximum principal strains in the drive belt.

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

mation of automobiles in crashes, noise and vibration in helicopters and planes, and so forth, it is not surprising that organizations find that merely exchanging model data (i.e., input and output files) does little to ensure reusability, adaptability, or interoperability of EAMs.

It has been estimated that the cost of imperfect computer-aided design (CAD) interoperability is at least $1 billion per year for members of the US automotive supply chain (Brunnermeier & Martin, 1999). Poor reusability, adaptability, and interoperability of engineering analysis models as a result of imperfect knowledge exchange could have similar, if not larger, cost impacts.

The lack of standards in this area has only compounded the problem. Although voluminous standards are beginning to emerge for the representation of analysis modeling data, such as the ISO 10303-104 (1994) standard for representation of finite element modeling data in the domain of structural analysis, there are no current standards for modeling knowledge. It is important to distinguish between *modeling knowledge* and *modeling data*. ISO 10303-104 (1994) supports the representation of entities such as coordinate systems, nodes, elements and element types, degrees of freedom, shape functions, element matrices, integration schemes, analysis control options, analysis results, and so forth. It does not support any information regarding modeling idealizations, limitations of the analysis model, analysis purpose and objectives, and the justification that supports each modeling idealization. For example, if a 2-D finite element model is electronically transmitted via ISO 10303-104 (1994) to someone else inside or outside the organization for reuse, adaptation, or interoperability purposes, answers to such basic questions such as what is the overall goal of the analysis, what 3-D geometric features of the system were approximated as having 2.5-D geometry or simply ignored and why, what changes in the original physical system would make this modeling simplification no longer appropriate, what loading and boundary condition idealizations were idealized as 2-D and why, and what are the constraints on all the various 2-D modeling idealizations, cannot be found or even inferred from ISO 10303-104 (1994) data.

2. RELATED WORK

This section outlines the various approaches taken by researchers in the past and present in two key areas: adaptation and interoperability and knowledge-based systems and ontologies for engineering analysis. The subsection on knowledge-based systems and ontologies focuses primarily work in the domain of FEA.

2.1. Reusability, adaptation, and interoperation

We define reusability as the ability to reuse an analysis model for the same application (typically with minor changes to one or more input parameters) by someone other than the model developer. Adaptation is defined as the ability to easily reuse or modify an analysis model developed for one application for another similar or related application. Interoperability is defined as the ability of independently developed models or analyses to be easily applied or combined for use in a hybrid application that involves multiple distinct analyses. This section deals with the existing approaches to reusability, adaptation, and interoperability problems. Most of these solutions are *ad hoc* and unsatisfactory, in a large measure because of the lack of suitable computer science foundations for addressing the underlying fundamental issues from which the problems arise.

Reusing or adapting an analysis model is a knowledge-intensive process, and to date, manufacturing enterprises have had very limited success in capturing, archiving, and reusing knowledge in a computational environment. Organizations resort to offline documentation of the design and analysis assumptions. Apart from offline documentation, the next most prevalent approach for overcoming this problem is automated transformation of data or code, such as by a macroprocessor or similar tools, for example, GenVoca (Batory et al., 1994). These, however, tend to be programming language or platform specific and unable to assure uniform, complete, or consistent transformations. Moreover, these existing approaches lack two critical features required of adaptation in engineering applications: the ability to capture both the nature and intent of adaptation, and the ability to combine or sequence adaptations and to recognize (in)compatibilities among them.

One of the most common concerns among manufacturing enterprises is the persistent lack of data portability and interoperability (Brunnermeier & Martin, 1999). Major CAD vendors that provide a complete package of tools [product data management (PDM), CAD, computer-aided manufacturing, FEA, etc.] have a distinct advantage in the market simply because interoperability concerns are lessened, even though one or more of the individual software components may be inferior to others in the market. Hence, this means that an interoperable solution these days requires organizations to lock into a single monolithic computational environment in lieu of a suite of application tools, each one being optimally suited for the specific task at hand.

The problem of interoperability has been addressed in areas apart from FEA. The most notable contribution has been in the area of PDM. Szykman et al. (2000) have proposed a foundation for interoperability in the next generation product development systems. They have made efforts to integrate data exchange standards such as ISO 10303-104 (1994), known as STEP (standard for the exchange of product model data) into the existing generation of software tools. This system addresses the lack of formal product representations such as function, behavior, and structure, which are the shortcomings in the existing PDM systems. These existing systems concentrate mainly on database-related issues and do not place any emphasis on information models for artifact representation.

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

Earlier work by deKleer and Brown (1983), Iwasaki and Chandrasekaran (1992), Alberts and Dikker (1992), Chandrasekaran et al. (1993), Henson et al. (1994), Goel et al. (1996*a*, 1996*b*), Qian and Gero (1996), Ranta et al. (1996), and Umeda et al. (1996) on product information representation has helped in the high-level division of artifact information into form, function, and behavior. Szykman et al. (2000) have developed on this high-level division to obtain a core-level representation consisting of objects and relationships with flow, function, form, geometry, material, behavior, specification, and artifact as the various classes representing them. Utilizing this type of a product representation would provide a rich source of knowledge that could be utilized in the development of the product. Although not in the area of engineering analysis, the fundamental concept of a formal information structure that, when instantiated, represents knowledge about a particular domain that can then be operated upon by computers and humans alike is a key component to how we propose to solve problems involving reusability, adaptation, and interoperability of engineering analysis models.

2.2. Knowledge-based systems and ontologies in engineering analysis

In this section we focus on research that has been done to develop knowledge-based systems for engineering analysis, with particular focus on FEA, and on the development of ontologies for supporting knowledge base systems in engineering. Surprisingly, relatively little work has been done in this area, although it has been over 20 years since the development of the rule-based expert system, MYCIN, in the field of medical diagnostics (van Melle, 1978; Clancey, 1983).

One of the early applications of the MYCIN environment to nonmedical domains was a structural-analysis assistant that addressed some of the numerical modeling aspects of the problem using the heuristic classification problem-solving paradigm (Clancey, 1985). By incorporating a large number of heuristics, these systems can recommend problem specifications, analysis strategies, and program options, starting with a description of problem features. Under most circumstances, a domain expert and a knowledge engineer are locked away for months or years and return with a model and computer program, which are comparable in performance to human specialists (Grower, 1982). This method has produced several remarkable programs such as SACON (Bennett et al., 1978), PROSPECTOR (Duda et al., 1978), R1 (McDermott, 1980), and INTERNIST (Miller et al., 1982). Yet, maintenance of these rule-based systems has been difficult, as seemingly insignificant changes in the rule base can produce dramatic and unpredictable results (Musen, 2000).

Over a period of time a number of frameworks have been developed to support engineering analysis, such as the frameworks proposed by Shephard et al. (1990) and the knowledge-based assistance for finite element modeling proposed by Turkiyyah and Fenves (1996). These frameworks continue to have the problems of domain-specific knowledge tightly coupled with procedures and rules, making the systems difficult to maintain and extend. Object-oriented frameworks address this issue. Extensive work by Tomiyama et al. (1989), Zimmerman et al. (1992), Mackie (1992), and Dubois–Pelerin and Zimmermann (1993) have shown a complete representation of FEA models in object-oriented frameworks. This has led to the development of a number of semiautonomous systems, such as an automated system for the modeling and analysis of multichip modules by Sheehy and Grosse (1997) and Holzhauer and Grosse (1999). These researchers developed a hierarchical object-oriented database to represent the physical system at various levels of abstractions, such as user-defined physical system level and computer-generated physical and finite element model levels. Even though these systems have been restricted to certain application domains such as thermal or structural analysis of multichip modules, they demonstrated the improved efficiency and ease of applying blackboard architecture to integrate the various numerical and symbolic knowledge sources in an object-oriented framework. Peak (2000) proposed the X-Analysis Integration Technology system, which includes capturing design knowledge, transferring the same into reusable libraries, and deployment of these template analyses for the creation of a new analysis type. However, in this system, as well as the previous systems discussed above, there is no formal representational scheme for modeling knowledge. Therefore, these systems cannot be easily extended by methods that would operate on this knowledge, nor can modeling knowledge be easily explicated and exchanged with people or software agents.

The use of Java-based frameworks to support engineering analyses has become a recent trend (1997). This work includes Onyx (Read & Afjeh, 1998), MOB-FEM (Shanbhag, 2002; Shanbhag et al., 2002), and FIPER (Rohl et al., 2000; Wujek et al., 2002). Onyx is a Java-based object-oriented application framework for aerospace propulsion system simulation that is capable of integrating advanced multidisciplinary and multifidelity analysis methods, dynamically constructing arbitrary simulation models, and distributing computationally complex tasks. This system provides a robust framework for interoperability between the different disciplines (structural, fluid, thermal, etc., referred to as phenomenon in this proposal) along with fidelity (0-D, 1-D, 2-D, and 3-D referred to as geometry in this proposal). However, the system requires a representation of these models to preexist in the database of components, and is restricted to aerospace propulsion systems. This system assumes the same kind of boundary, initial, and load condition when analyzing a particular component of a model. Hence, reusability, adaptability, and interoperability exists only in the context of a specific multidiscliplinary analysis problem. Onyx exploits the Java reflection capability in developing the analysis models but has no capability of intercession, unlike OpenJava (Tatsubori, 1999). Shanbhag et al. (Shanbhag, 2002;

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

Shanbhag et al., 2002) developed an OpenJava application called MOB-FEM for automatically reusing finite element modeling knowledge. A separate Visual Basic tool, called TEK-FEM, with a graphical user interface was developed to extract modeling knowledge from a domain expert. This knowledge is then stored in a Microsoft Access backend database. Although the modeling knowledge extracted from the expert is based on a taxonomy similar to the one proposed by Finn et al. (1992), the knowledge was not represented or stored using a formal information structure. As a result, the researchers were forced to hardwire much of the functionality of the coupled TEK-FEM/MOB-FEM system for the specific problem used to demonstrate the methodology: thermal cooling FEA and subsequent mold ejection elastostatic FEA of a plastic part. FIPER, the result of a $21.5 million project effort funded primarily by NIST's Advanced Technology Program and a number of high technology companies, is a component-based Web-centric framework with a common protocol that to enable companies to "wrap" their engineering tools, data, and processes into the FIPER environment. This enables across the Web data exchange and access to various engineering tools that support product development. The architecture is based on Java and Sun's Jini software system. However, there is no across the Web knowledge exchange.

Parallel to the development of these frameworks to support engineering analysis activities, researchers in the field of artificial intelligence began to uncover serious shortcomings with procedural rule-based expert systems. They discovered that subtle changes in the way the rules were organized produced dramatic changes in the expert system results (Buchanan & Shortliffe, 1984). It was clear that relationships existed among the rules that were difficult to determine by direct inspection of the knowledge base, and therefore, large production rule systems were difficult to maintain and problematic (Bachant, 1988). A way was needed to separate out operational knowledge, that is, problem-solving methods, from the domain knowledge upon which the problem-solving methods operate.

One of the most promising ways to separate and represent domain knowledge from operational or problem-solving knowledge is through the use of ontologies. A popular definition of ontology is the one offered by Gruber in 1993: *ontology is an explicit specification of a conceptualization of a domain*. It is a formal specification of the terms in the domain and the relations among them. Noy and McGuinness (2001) elaborate to define an ontology as a formal explicit description of domain concepts, often called classes, properties of each class, called slots, which describe the various features and attributes of the class, and restrictions on the properties or slots, called facets. The classes may be arranged hierarchically with subclasses inheriting properties from their superclasses and adding further specification with additional properties and facets. Ontologies represented or modeled in this manner are well suited for implementation using object-oriented languages.

When an ontology is populated with information that fills the properties of the classes, that is, instances of the class objects, it represents a knowledge base for that particular domain. One important distinction between an ontology-based knowledge tool and a pure object-oriented architecture is that, in the latter, problem-solving methods are interleaved with the knowledge structure, that is, class and instance objects. This has advantages in terms of permitting a good bit of flexibility regarding the semantics of encoded objects and making it straightforward for developers to include new functionality into the code. Yet, it is impossible to consider control–flow relationships separately from the data structures that comprise the object hierarchy or to view the data or knowledge model without being distracted by associated program code (Musen, 2000).

The development of ontologies for a wide variety of domains has been an active area of research in recent years. Extensive ontology libraries can be found at the Stanford's Knowledge Systems Laboratory (www.ontolingua.stanford.edu) and at DARPA's DAML ontology library (www.daml.org). Unfortunately, little work has taken place to date in developing formal ontologies for engineering and for engineering analysis modeling knowledge. However, there are a few notable contributions in this area. Dym and Levitt (1991) appear to be the first to propose taxonomies for engineering analysis modeling knowledge. The authors present a topology of engineering knowledge that includes fundamental or first principle knowledge, phenomenological knowledge, analytical model knowledge, numerical model knowledge, and even meta knowledge, such as goals, objectives, and intentions of the analysis model. They implement their knowledge topology in a LISP-based Life Safety Code Advisor, a prototype knowledge-based expert system for reviewing architecture designs for conformance with NFPA safety code. In a similar vein, Finn et al. (1992) proposes an intelligent modeling assistant based on a taxonomy of modeling idealizations for continuum-based analysis models, and subsequently proposes a system (CoBRA) for the development of a model of a physical system (called a physical model by the authors) adopting both case-based and model-based reasoning (Finn & Cunningham, 1994). They describe the generation of a model as the application of idealizations and simplifications of spatial, phenomenological, and temporal aspects. They have broken down the modeling simplifications into two categories: geometric and phenomenon. The four types of geometrical simplifications discussed by Finn et al. (1992) are dimensional reduction, geometric symmetry, feature removal, and domain alterations. Similarly, the phenomenon simplifications are reduced or removed. Phenomenon removal is the omission of an entire phenomenon from an analysis, while phenomenon reduction is the removal of a component of a phenomenon such as ignoring radiation in a heat transfer analysis.

Borst et al. (1994, 1995) have developed a set of formal engineering ontologies called PhysSys for dynamic physical systems. The set includes three ontologies (component,

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

process, and engineering mathematics) with ontological projections that map components to processes and processes to engineering mathematics. PhysSys basically defines components as carriers of physical processes that can be represented mathematically with physical quantities and mathematical relations using Gruber's and Olsen's EngMath ontology (1994). The authors discuss the need to include metalevel information as attributes of engineering analysis models, and suggest validation information and modeling assumptions as examples of such knowledge. Their system captures this knowledge as text, but the authors acknowledge that a more formal structure is needed so that a support system for engineering modeling could actually reason about it.

3. RESEARCH APPROACH

We argue that formal taxonomies or ontologies for the representation of engineering analysis modeling knowledge are needed. Efficient methods and tools are needed for extracting analysis modeling knowledge from engineers and incorporating this knowledge into a computational environment. Finally, we need methods and associated tools that can exploit the existence of such knowledge in a computational environment to improve interoperability, reusability, and adaptability of analysis models.

3.1. Ontologies for representing engineering analysis modeling knowledge

Noy and McGuinness (2001) provide a step by step guide for developing an ontology. Based on their methodology, we have begun to develop an ontology that would be applicable to all engineering analysis models. This ontology draws upon some of the analysis modeling taxonomies and concepts presented by Tomiyama et al. (1989), Finn et al. (1992), Dym and Levitt (1991), Finn and Cunningham (1994), Sinha et al. (2001), and Paredis et al. (2002). Figure 2 shows our hierarchical categorization of types of analysis models. We first distinguish between physics-based and non-physics-based models. Physics-based models are predictive models of the physical behavior of a physical system, such as an engineered product or manufacturing process. Example of a non-physics-based model is a market analysis model that predicts the demand for a product. In this research we are focused on physics-based models, which we further subtype as empirical, lumped parameter or discrete, and continuum based, and our proposed ontologies are applicable to all physics-based engineering analysis models. Empirical models are models directly derived from experimental observations, such as response surface models or material behavior models derived from material testing. Lumped parameter or discrete models are mathematic models that consist of one or more interconnected model components with individual model component parameters lacking spatial dependency. Continuum-based or distributed models are mathematic models with one or more variables or parameters that are dependent upon one or more independent spatial variables. Continuum-based models are classified as analytical or numerical, depending on whether or not numerical techniques are needed to obtain the model solution. The class of numerical continuum-based models has a number of subclasses according to specific numerical techniques, such as boundary element method, finite element method, finite difference method, wavelet methods, and hybrid methods involving one or more of the previous methods. As new methods emerge, additional subclasses would be included here.

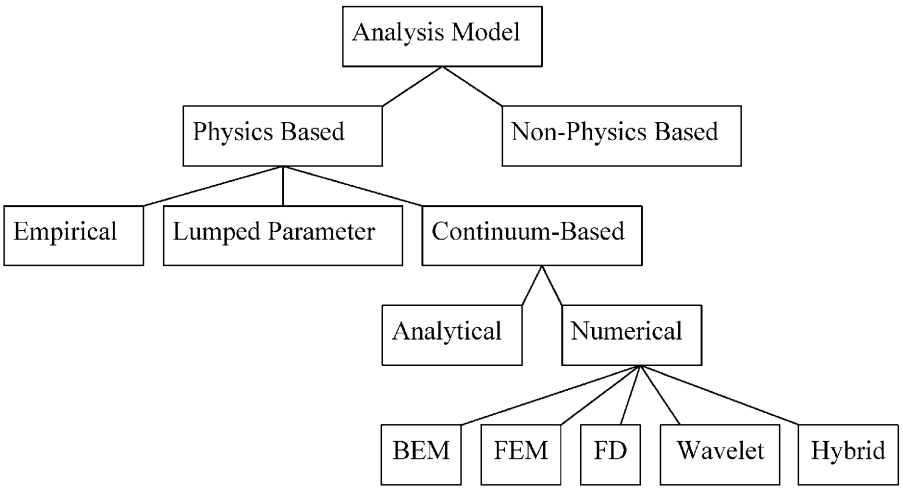


Fig. 2. The class hierarchy of some engineering analysis model types. The proposed ontologies are applicable to all physics-based engineering analysis models.

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

Table 1. *Properties of engineering analysis models*

EAM Term	Type	Cardinality	Notes
Model name	String	Required single	Unique identifier
Description	String	Required single	Short and concise
Version no.	Float	Required single	Recommend x.x format
Documentation	String	Multiple	References to technical report(s)/presentations
Creator or developer	String	Required multiple	
Creation date	Date	Single	<Current date
Modifiers	String	Multiple	
Modification dates	String	Multiple	<Current date
Development time	Float	Single	In hours
Purpose	String	Required multiple	
Primary objective	String	Required single	Engineering quantity predicted
Secondary objectives	String	Multiple	
Physical system basis	String	Single	Product or process
Graphic of model	Instance	Multiple	Image files of model
Accuracy expectation	String	Single	Qualitative multiple choice
Resolution expectation	String	Single	Qualitative multiple choice
Causality expectation	String	Single	Qualitative multiple choice
Software requirements	String	Multiple	
Hardware requirements	String	Multiple	
Robustness	String	Single	Qualitative multiple choice
Model input	String	Required multiple	List of input parameters
Model output	String	Required multiple	List of output parameters
Model limitations	Instance	Required multiple	List of model limitations
Model idealization	Instance	Required multiple	List of model idealizations
Model components	Instance	Multiple	List of model components
Model development steps	String	Multiple	List of major model development steps
Intended user/user group	String	Required multiple	Employees, customers, etc.
Training hours required	Float	Required single	0 = no training
Validation plan	String	Single	
Certification level	Integer	Single	

In Table 1 we identify basic properties of all physics-based analysis models. The first column is the term employed to represent this modeling knowledge concept, the second column is the type of information contained in this concept, the third term is the cardinality of this information, and the fourth column lists constraints or other properties of this concept. The information is typed to facilitate computational representation and manipulation. An instance type means that this information is an instance or object of another class with separate properties that define that class. Cardinality is the number of values an instance of the term or property may take on. Note that a cardinality of multiple means this term supports multiple values, but it can be empty. A term with required multiple cardinality means that the term must contain at least one value. A term with required single cardinality means the term must and can only have one value.

The class of engineering analysis models has the basic properties of name (a string), creator (a string), creation date (a date), modifier (a string), modification date (a date), model version number (a floating point number), and model documentation (a string that provides links to technical reports, presentations, etc.). This information is similar to standard software engineering documentation. We assert that additional more abstract, or meta, knowledge needs to be formalized into an ontology to support engineering analysis modeling, as shown in Table 1. Specifically, all engineering analysis models have a purpose that describes what the overall goal of the model is in terms of strategic objectives of the engineering department. An engineering analysis should have a primary objective in terms of specific analysis results that are sought (i.e., specific engineering quantities of interest, such as maximum temperature, stress, deflection, response time, natural frequency, acoustic amplitude, power, energy, force, etc.), and possibly secondary objectives. Engineering analysis models are derived from physical systems, such as products, subassemblies, components, or manufacturing processes, and therefore, this information should be captured.

As Hazelrigg (1996) notes, analysis models have three fundamental properties: accuracy, resolution, and causality. Resolution is the ability of the model to differentiate between, or resolve, results for different values of input parameters, and causality is the ability of the model to help establish causal relationships between output parameters and input parameters. Therefore, in our ontology we include as fundamental properties of the analysis model class slots for expected accuracy, expected resolution, and expected cau-

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

sality. At this point it is not clear how to type these slots, that is, as numbers or perhaps as strings for qualitative information.

We also include model robustness in our ontology. We define robustness as a measure of the stability of the engineering analysis model to variations in input parameters. It should be noted that if the model seeks to simulate an inherently unstable phenomenon, such as a flame, then instability in the results may be indicative of the physical system being simulated and not due to a lack of model robustness. Instability in computer-based analysis is often due to finite precision in digital computers, coupled with large differences in scales of physical quantities represented in the analysis model. Included in our definition of robustness is the degree to which the analysis tool (or macro that executes the analysis tool) has error trapping to detect if supplied input parameters are out of bounds. Analysis models may demand certain resource requirements in terms of software and hardware, and therefore, the analysis class could include these slots to specify this information as well.

All analysis models have a set of limitations that describe the applicability of the analysis model to a class of problems. Each limitation may be the result of one or more modeling idealizations and/or due to limitations of the analysis tool used to solve the analysis model. Each modeling idealization has justification knowledge to support the specific modeling assumption. Figure 3 shows the relationship between the Analysis Model, Limitation, Idealization, and Justification classes, as well as fundamental properties of each concept or class. Each concept has slots for specifying the name and description of an instance of the concept. For the sake of brevity, the property list for the Analysis Model class is abbreviated in the figure to include only the additional slot for storing limitations property. The complete list of properties of the Analysis Model class is given in Table 1. In addition to a Justification property, the Idealization class includes properties for specifying the effect the idealization has on accuracy, resolution, causality, and reducing the analysis modeling cost. Justification knowledge includes three properties for justifying the idealization based on heuristics, theory, or first-principle, and/or empirical observations.

Analysis models are often comprised of components that may or may not map directly to physical components. Thus, we have included in our ontology a slot for specifying a list of model components that comprise the engineering analysis model. As in the model itself, model components may have specific limitations associated with the component. The component limitations are associated with instances of the idealization class with justification knowledge for each idealization.

All analysis models have inputs, either parameters or files containing input data, and outputs, either parameters or files containing output data. Input parameters and files have associated suppliers, whereas output parameters and files have associated recipients. Suppliers and recipients may be people, databases, or other software tools. Input parameters have constraints that restrict their values. Our ontology struc-

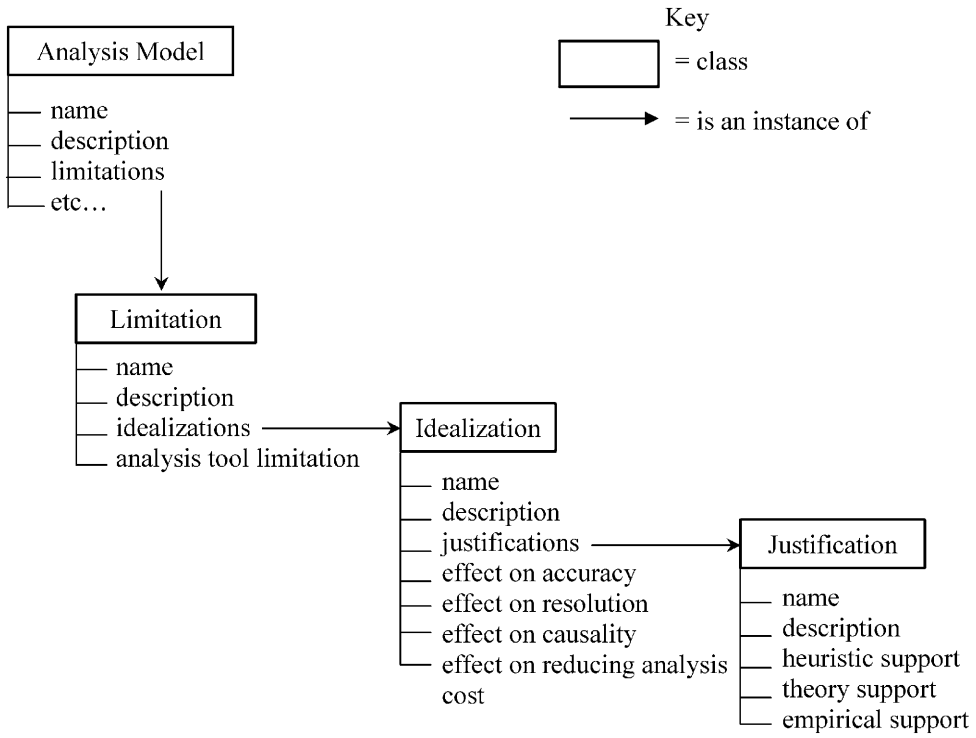


Fig. 3. The properties and relationships of the analysis model, limitations, idealization, and justification classes.

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

ture would support explicit representation of all of this information by having the model input parameter slot value be an instance of the input parameter class.

We can think of other modeling knowledge that may be important, especially to organizations with quality control processes imposed on the model development process. For example, the model may have a validation or critical risk reduction plan that seeks to address uncertainty with regard to validity of model idealizations or a certification level indicating the overall robustness and accuracy of the model. The ontology could capture the model development process itself, either informally by specification of model development steps as a property of the analysis model class using a text list or more formally by an instance of a process ontology. A process ontology has fundamental process properties, such as task, inputs for a task, outputs for a task, preceding task, following task, task resources required, and so forth.

3.2. Domain-specific EAM ontology research and development

In Section 3.1 we have broadly discussed the type of knowledge that might support all EAMs. Within this overall EAM class, there are subclasses of analysis models, as shown in Figure 2. Each of these subclasses of EAMs may have specific modeling knowledge that needs to be specified in a formal information structure. For example, continuum-based analysis models have the concept of geometry as defined by one or more spatial coordinates and have an underlying theory based on continuum mechanics, and therefore have a unique class of modeling idealizations related to spatial coordinates. For example, geometry itself can be idealized as 2-D or 1-D, geometric features can be ignored or modified, forces and constraints can be idealized as point loads, and so forth. Further, all but the simplest continuum-based models require numerical techniques, such as FEA, to solve. This introduces additional modeling knowledge related to discretization techniques and the specification of the proper numerical solution techniques. For example, in the finite element model analysis subclass discretization knowledge could include element types and meshing methodology.

Lumped parameter models have the concept of flow, such as energy or current, between components that are defined with specific input–output relationships. Therefore, lumped parameter models have idealizations regarding input–output black box behavior of interconnected functional components of the model. Underlying theory, such as control theory, supports these models. Empirical models have the concept of observations and measurement error. Statistical theory supports empirical models.

4. AN EXAMPLE IMPLEMENTATION OF EAM ONTOLOGIES

We have implemented our EAM ontologies into a computational framework and instantiate the object classes to form a prototype engineering analysis modeling knowledge base called ON-TEAM. Implementation has been facilitated by using Protégé-2000, a noncommercial open-source Java tool developed at Stanford's Medical Informatics Lab that provides an ontology and knowledge-base development environment (www.protege.stanford.edu; Grosso et al., 1999). Using Protégé-2000 we were able to quickly implement our engineering analysis domain ontology, design knowledge acquisition forms, and enter engineering analysis modeling domain knowledge. Protégé-2000 supports plugins and application program interfaces (APIs). Thus, ON-TEAM can be extended with object-oriented methods that operate on the knowledge base to perform various tasks.

Figure 4 shows the basic object class ontology implemented in ON-TEAM. In the window pane on the left the various classes and class hierarchy for the Analysis Modeling Knowledge class are shown. The Analysis Modeling Knowledge class is an abstract class with no properties that serves as an organizational container for subclasses Analysis Models, Component, Development_Process, User, Limitation, Idealization, Idealization_Justification, Analysis_Tool_Limitation, and Parameter each with their own properties. The pane on the right shows the basic properties of the Analysis_Model subclass that were discussed in Section 3.1. Note that some slot or property information is specified as a text string such as the purpose of model, primary, and secondary objectives, and model documentation (URL to technical reports), as a multiple choice pick from a list (type is symbol) such as accuracy expectation, resolution expectation, causality expectation, robustness, and so forth, as a number such as model version number, as a class such as intended user group, or as an instance of a class such as model inputs, model outputs, output parameters, limitations, model idealizations, model components, graphic of the model, and intended users.

5. AN APPLICATION EXAMPLE

In this section we present a real world engineering analysis modeling application. Engineers at United Technologies Research Center and Otis Elevator Company, both units of United Technologies Corporation, develop sophisticated analysis models to validate new technology that are used to improve products. One new technology that has recently been introduced in the NextStep™ escalator product is a drive unit that employs a thermoplastic PU (TPU)-coated, cogged steel belt for transferring forces from the escalator drive motor to the steel links that directly drive the escalator steps. This TPU-coated steel belt technology replaces a conventional steel chain drive, improving noise and vibration characteristics while reducing the maintenance costs of the escalator. Numerous analysis models were developed during the course of developing and validating this technology. Figure 5 is a screen capture of part of the completed modeling knowledge acquisition form for a 2-D FEA model

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

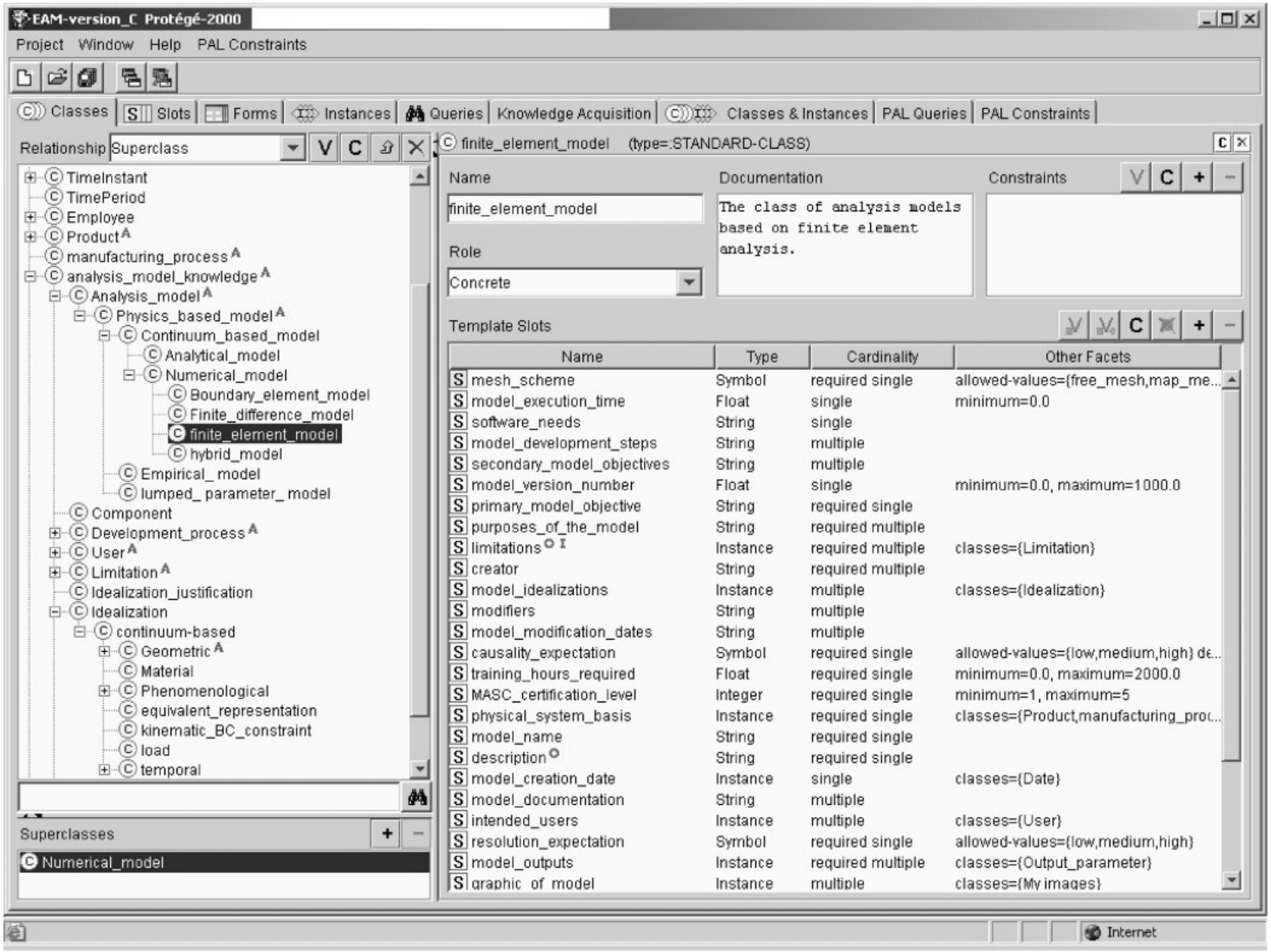


Fig. 4. Implementation of an ontology for representing analysis modeling knowledge.

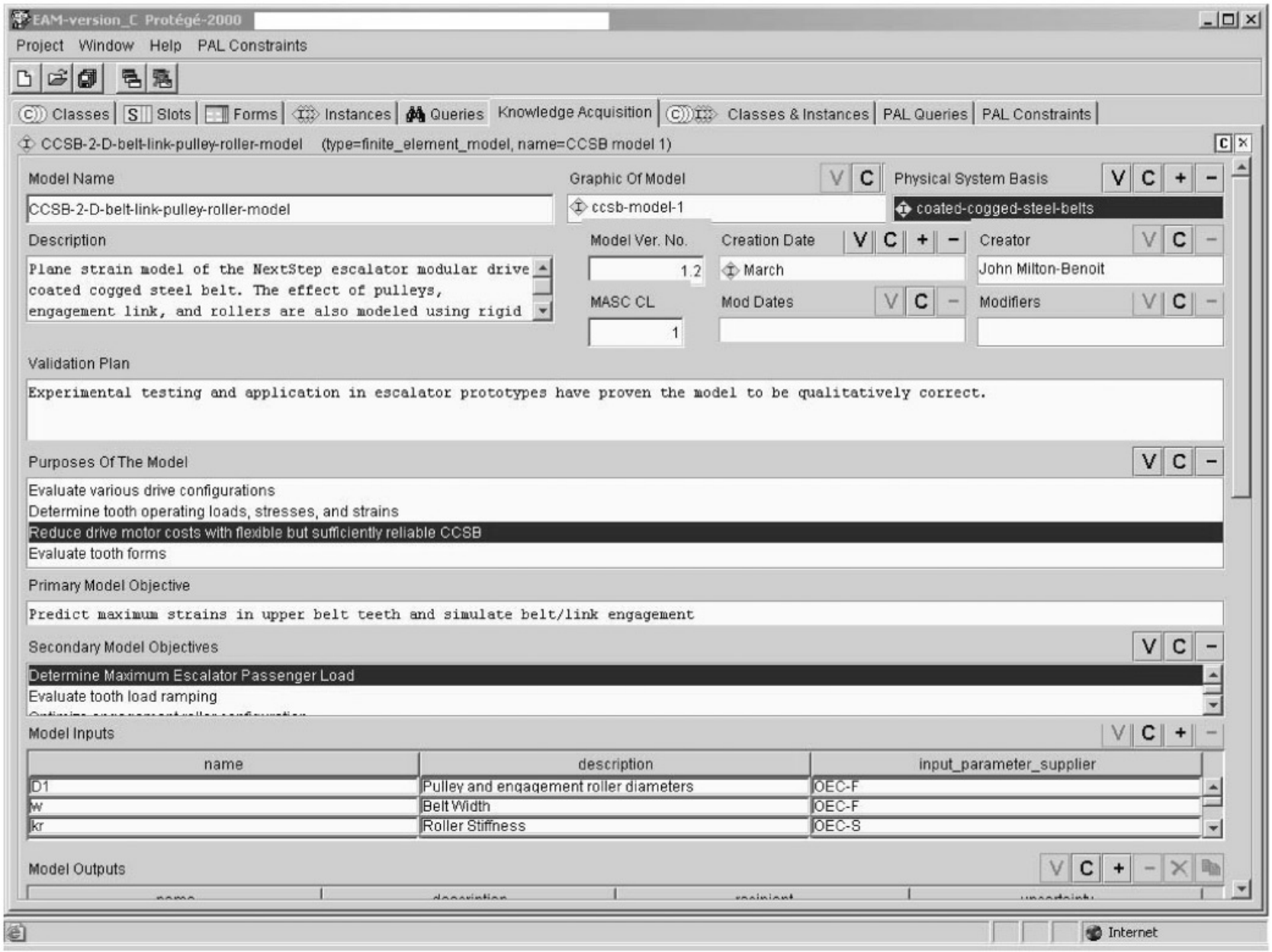


Fig. 5. A screen capture of the completed knowledge acquisition form for the NextStep™ escalator modular drive.

of the drive unit modular. In this interface the "V" buttons stand for view, "C" for create, "+" for add from a list of existing objects or instances of objects, and "−" for delete. If the user clicks on the V button of the graphic of model widget, the window shown in Figure 6 pops up. This is a maximum principal strain contour plot showing the strain distribution in the cogged belt as the belt is engaged by the link, which is represented by a rigid surface. The knowledge engineer, that is, the original developer of this analysis model, provided this analysis result figure and specified to the ON-TEAM system the location of the image file to be displayed by the graphic of model property. Note from Table 1 or Figure 4 that the graphic of model property of the analysis model class is, in turn, an instance of the "My Images" class. Thus, the information shown in the various parts of Figure 6, such as the image name, the image itself, uses, filename, and so forth, is specified as properties of this generic image class. Thus, we can use the My Images class to support storage of graphical information that might be instances of many different types of classes. For example, the Physical System Basis property of the analysis model class shown in Figure 5, as well as in Table 1 and Figure 4, is an instance of either the product or manufacturing process class. The product (or manufacturing process) class include the property "graphic of product" (or "graphic of manufacturing process"), which is an instance of the My Image class. Thus, by clicking on the view physical system basis button shown in Figure 5 and then the subsequent view graphic of product button, the result is shown in Figure 7. Thus, the analysis model class, the product class, and the manufacturing process class each have a graphic property that is an instance of the My Image class. This is an excellent example of how object-oriented technology facilitates modularity and object reuse.

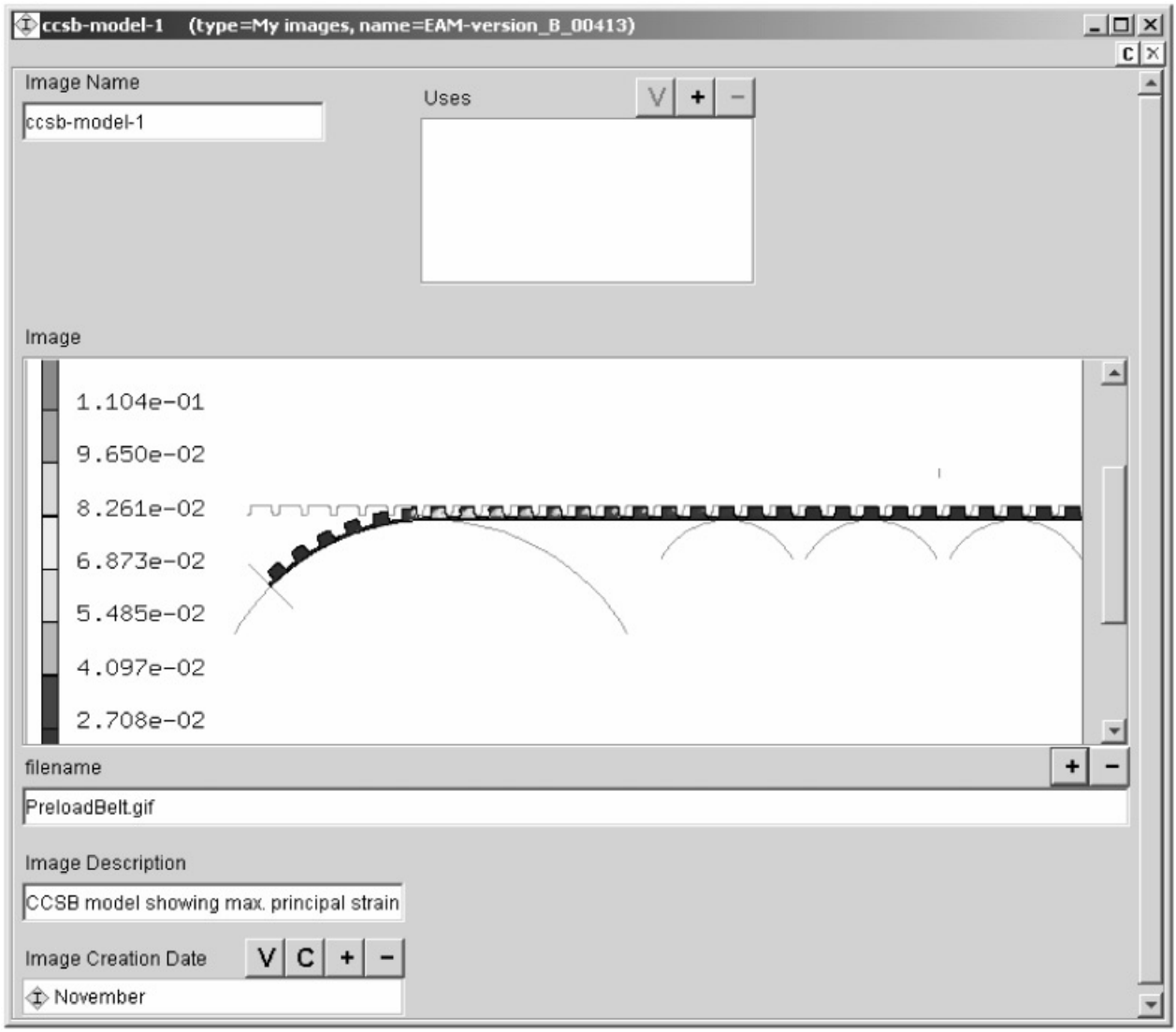


Fig. 6. A graphic of the analysis model results for the NextStep™ escalator drive unit module.

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

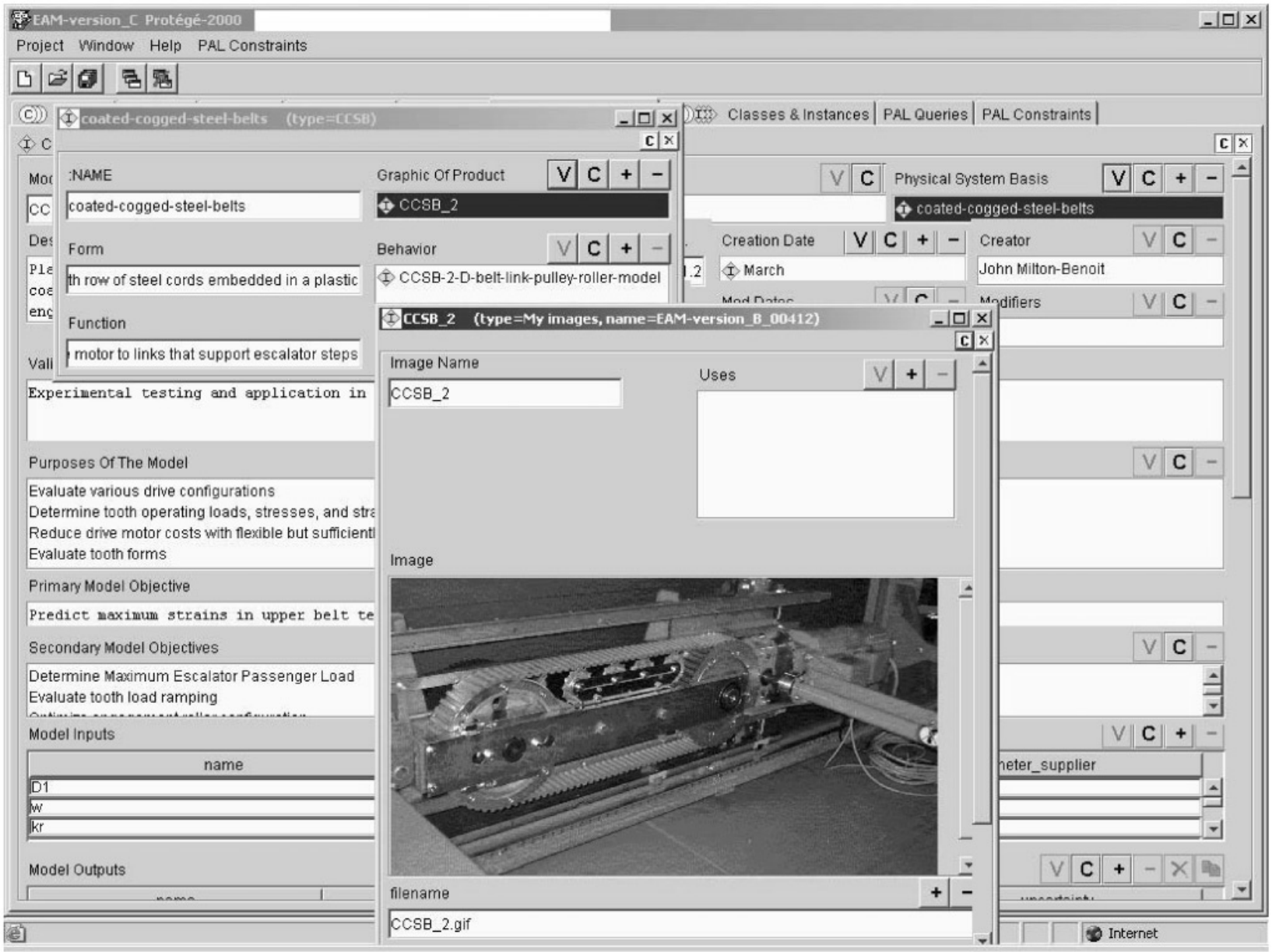


Fig. 7. A graphic of the physical system basis of the NextStep™ escalator drive unit module.

By comparing the graphic of the analysis model results (which shows the underlying model) shown in Figure 6 and the graphic of the physical system basis (i.e., the picture of the NextStep™ escalator drive unit module), it is clear that the analysis model is a significant abstraction, or idealization, of the underlying physical system. In ON-TEAM analysis models have a property called idealizations, which are instances of the idealization class. Idealizations can be directly inspected by clicking on an idealization shown in the window or by inspecting first the limitation property. Limitations are instances of the limitation class, and one of the properties of this class include idealizations that contribute to the limitation. Figure 8 shows a portion of the knowledge acquisition form. The figure shows inspection of the specific analysis model limitation that stresses are uniform through the depth of the belt in this analysis model. Figure 8 also shows that the associated idealization for this limitation of the analysis model is called steel_corded_belt_2D_dimensional_reduction and the properties of this idealization. This idealization essentially involves abstracting the round steel cords contained within the belt as a single rectangular section steel cord that extends through the entire belt depth, so that the belt may be modeled with 2-D geometry. The justification knowledge for this idealization, called steel_cord_equivalency, is an instance of the justification knowledge class whose properties are shown in Figure 3. The values of the properties of steel_cord_equivalency justification knowledge provides the equivalency calculation in which the section and material properties of the rectangular steel cord section are determined such that it is equivalent to the set of round steel cords in terms of uniaxial and bending stiffnesses.

The knowledge acquisition forms shown in the Figures 5–8 are the default (but customizable) forms offered by the development environment to enable the domain expert to input modeling knowledge for a specific application into ON-TEAM. However, different organizations may prefer different types of user interfaces for specifying knowledge. For example, some organizations may prefer a question and answer format, or a specific checklist sequence with accompanying pull-down menus. Because Protégé-2000 is a free open-source Java development environment, organizations can customize how knowledge is acquired to suit their preferences. As currently implemented, the knowledge provider interacts with the knowledge acquisition form, proceeding top down through the properties of the analysis model class. If an analysis model property is an instance of another class, such as the analysis model limitations slot values are instances of the limitation class, then the properties or slot values of the linked class instances must also be provided. However, a more bottom-up approach may prove more intuitive for the knowledge provider. Referring to Figure 3, it may be more natural for the knowledge provider to first create instances of all the modeling idealizations of the analysis model by defining properties of instances of the limitation class and supporting these properties with instances of justification knowledge. Then the knowledge provider would associate these idealizations with resulting instances of limitations of the analysis model by linking the model idealizations to the idealizations slot of the limitation class. The limitation instances are then linked to the analysis model class via the limitations slot of the analysis model class.

Another issue is how to best present this knowledge to end users of the system. One Web-enabled way that knowledge can be presented to end users is via a server hosting html tables containing a library of the analysis modeling knowledge. Through standard features and plug-ins developed for Protégé-2000, ON-TEAM has the ability to output its knowledge base in various formats, including html, xml, rdf schema, and, through Java database connectors, to MS Excel, or various common relational databases. Table 2 below shows a portion of the html table generated automatically by ON-TEAM for the application example. All underlined items in the table are hyperlinked to additional html tables with property values that provide the appropriate knowledge.

6. CONCLUSIONS AND FUTURE WORK

In the world of rapidly evolving technology and highly competitive markets, manufacturing enterprises rely heavily on engineering analyses to provide the information needed to inform design decisions as quickly and as cost effectively as possible. Considerable modeling knowledge is invested in the development of these analysis models, and yet no formal schemes exist to represent and operationalize this knowledge in ways that will facilitate successful exchange, reuse, adaptation, or interoperation of analysis models. In this research we propose such a scheme based on identifying and classifying fundamental analysis modeling knowledge into a set of formal ontologies. We then presented an implementation of these ontologies into a Java-based object-oriented information structure, which was instantiated with a real-world industrial analysis model to form the basis for an engineering analysis modeling knowledge base system called ON-TEAM.

It should be noted that our research in developing ontologies to support engineering analysis modeling knowledge and implementing them in ON-TEAM is a work in progress. To validate and refine the effectiveness of the proposed ontologies and our implementation tool, evaluation is needed in the context of real-world product development activities. To this end we will draw upon the *Center for e-Design and Realization of Engineered Products and Systems*, a new NSF-sponsored Industry/University Cooperative Research Center currently involving the University of Massachusetts, University of Pittsburgh, University of Central Florida, and approximately 15 companies (see www.e-designcenter.info for more information). The Center's industry members will serve as an evaluation test bed for the proposed ontologies and our ON-TEAM implementation in the context of practical engineering analysis interoperability problems.

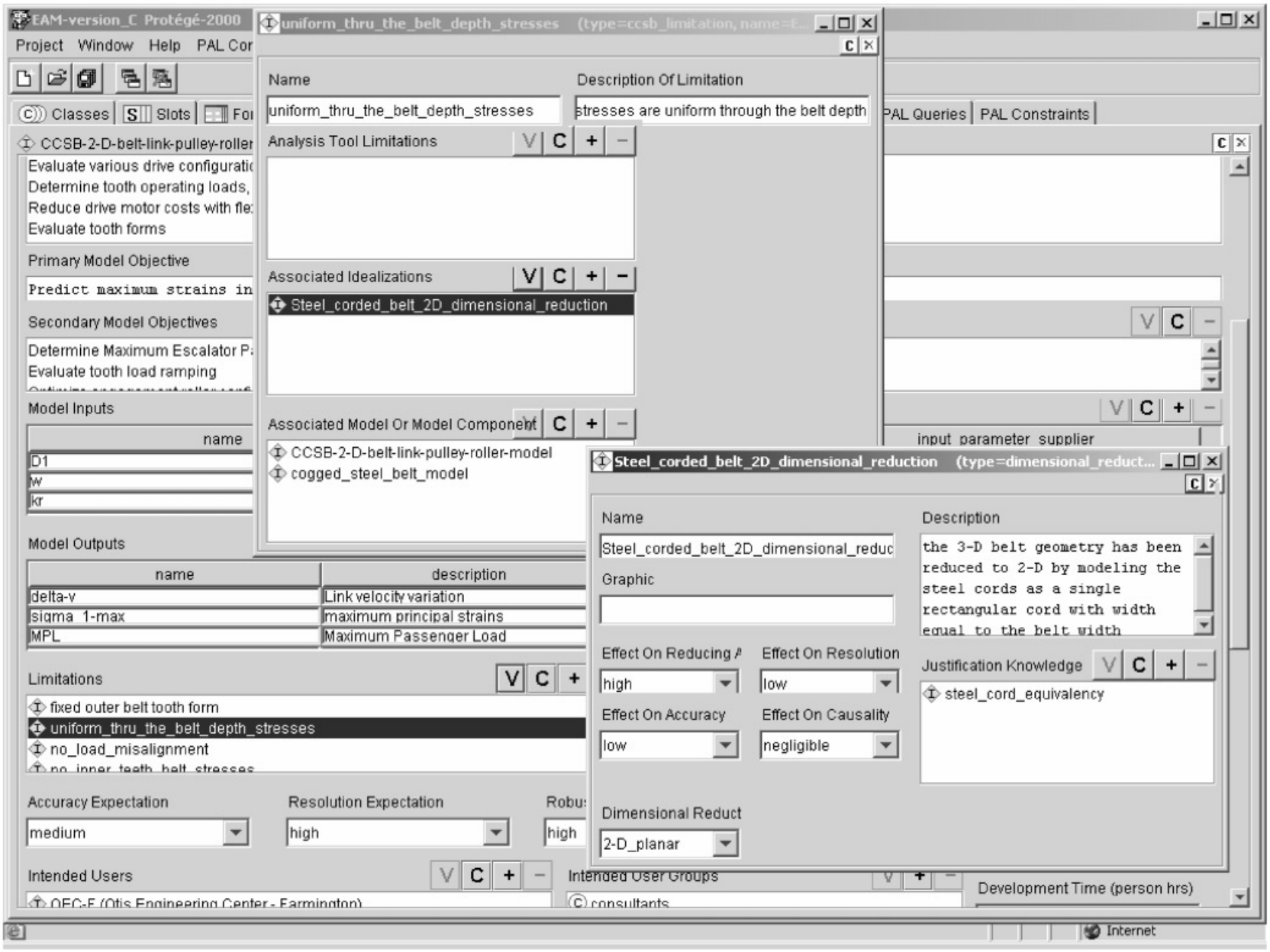


Fig. 8. Part of the knowledge acquisition form showing inspection of a specific model limitation.

Table 2. *Partially hyperlinked html table of application-example EAM knowledge for CCSB-2-D-belt-link-pulley-roller-model and finite_element_model class*

Own Slots		
Slot Name	Value	Type
Model_name	CCSB-2-D-belt-link-pulley-roller-model	String
Description	Plane strain model of the NextStep™ escalator modular drive, coated cogged steel belt. The effect of pulleys, engagement links, and rollers are also modeled using rigid surfaces and spring elements.	String
Model_version_number	1.2	Float
Intended_users	1. OEC-F (Otis Engineering Center–Farmington) (of class customers) 2. John Milton_Benoit (of class employee_users)	Instance
Physical_system_basis	coated-cogged-steel-belts (of class CCSB)	Instance
Graphic_of_model	ccsb-model-1 (of class My images)	Instance
Primary_model_objective	Predict maximum strains in upper belt teeth and simulate belt/link engagement	String
Secondary_model_objectives	1. Determine maximum escalator passenger load 2. Evaluate tooth load ramping 3. Optimize engagement roller configuration	String
Accuracy_expectation	Medium	Symbol
Robustness	High	Symbol
Limitations	1. fixed outer belt tooth form (of class ccsb_limitation) 2. uniform_thru_the_belt_depth_stresses (of class ccsb_limitation) 3. no_load_misalignment (of class ccsb_limitation) 4. no_inner_teeth_belt_stresses (of class ccsb_limitation) 5. quasi-static analysis (of class ccsb_limitation) 6. constant_belt_velocity (of class ccsb_limitation)	Instance
Model_idealizations	1. steel_corded_belt_2D_dimensional_reduction (of class dimensional_reduction) 2. feature_suppression (of class Feature_class_suppression) 3. rigid_link_tooth_surface (of class Material) 4. rigid_idler_pulley (of class Material) 5. rigid_drive_pulley (of class Material) 6. constant_velocity (of class suppress_time_behavior)	Instance
Model_outputs	1. delta-v (of class output_numeric_parameter) 2. sigma_1-max (of class output_parameter) 3. MPL (of class output_numeric_parameter) 4. tooth_load (of class force)	Instance
Model_creation_date	March (of class Date)	Instance
Secondary_model_objectives	1. Determine maximum escalator passenger load 2. Evaluate tooth load ramping 3. Optimize engagement roller configuration	String
Training_hours_required	10.0	Float
Model_inputs	1. D1 (of class float_input_parameter) 2. w (of class float_input_parameter) 3. kr (of class float_input_parameter) 4. C1 (of class float_input_parameter) 5. PL (of class float_input_parameter) 6. material models (of class material_behavioral_model of class input_parameter) 7. tooth form (of class geometric_forms of class input_parameter)	Instance
Validation_plan	Experimental testing and application in escalator prototypes have proven the model to be qualitatively correct.	String

Because our formal ontology for EAM concepts has been modeled using object-oriented concepts and implemented using an open-source Java-based tool that provides an API, object-oriented methods can be implemented to operate on this knowledge to perform a variety of important tasks. These tasks include the ability to automatically inspect the modeling knowledge for consistency, editing, maintaining, and extending the knowledge base for model reuse and adaptation. In addition, methods can be developed to assess the suitability of an analysis model in terms of its robustness,

https://doi.org/10.1017/S0890060405050018 Published online by Cambridge University Press

costs, value, uncertainty, and so forth, in meeting a particular analysis goal, to select the most appropriate analysis model class for a given analysis problem, to enable interoperability of multiple distinct analyses, and to customize interaction with our environment based on properties of the user.

Although capturing EAM knowledge in a well-designed ontology structure is expected to facilitate reusing, adapting, and exchanging analysis models, the captured knowledge is of very limited utility unless it can be easily brought to bear on real engineering analysis problems. The kinds of methods that are needed include methods for assessing the properties of models, for determining the applicability of specific models to specific analysis problems or categories of problems, for figuring out whether and how two or more models can interoperate, and for customizing user interfaces to support novice engineers, students, or users with different learning styles. The former methods should be based on characteristics of the captured EAM knowledge associated with models using the ontology structure, while the latter methods are based on characteristics of different user classes. To support this, a user ontology structure could be implemented as part of our system. In this scheme users would be classified based on analysis experience level, underrepresentation, and ethnic group. This information will be requested as voluntary information at the start of an interactive session with the ON-TEAM environment.

With a basic user ontology included as part of our system and tracking of user interactivity, one can study how ON-TEAM is used by different classes of users, such as novice or student, intermediate, advanced, underrepresented, and ethnic group classes. Refinements based on this study will enable us to maximize the effectiveness of our EAM knowledge base environment for a diverse mix of users.

Finally, the foundations laid by an ontological representation of engineering analysis modeling knowledge can serve as basis for the development of an international standard to supplement existing analysis data and CAD standards. With such a schema we believe engineering analysis models can be more easily exchanged, reused, and adapted by people within and external to engineering organizations.

ACKNOWLEDGMENTS

The authors acknowledge the support of United Technologies Research Center (UTRC) in East Hartford, CT, and NSF under their I/UCRC program (Grant EEC0332508). In addition, the first author is particularly grateful to Dr. Orisamolu and Dr. Bonilha of UTRC for the sabbatical opportunity and guidance and support of this research.

REFERENCES

Alberts, L.K., & Dikker, F. (1992). Integrating standards and synthesis knowledge using the YMIR ontology. In *Artificial Intelligence in Design* (Gero, J.S., & Sudweeks, F., Eds.), pp. 517–534. Boston: Kluwer Academic.

Bachant, J. (1988). RIME: preliminary work toward a knowledge acquisition tool. In *Automatic Knowledge for Acquisition for Expert System* (Marcus, S., Ed.), pp. 201–224. Boston: Kluwer Academic.

Batory, D., Singhal, V., Thomas, J., Dasari, S., & Sirkin, M. (1994). The GenVoca model of software-system generation. *IEEE Software 11(5)*, 89–94.

Bennett, J., Cleary, L., Englemore, R., & Melosh, R. (1978). *SACON: A Knowledge-Based Consultant for Structural Analysis*. Report No. STAN-CS-699. Stanford, CA: Stanford University, Department of Computer Science.

Borst, P., Pos, A., Top, J.L., & Akkermans, J.M. (1994). Physical systems ontology. In *Working Papers of the European Conf. Artificial Intelligence ECAI'94 Workshop on Implemented Ontologies* (Mars, N.J.I., Ed.), pp. 47–80. Amsterdam: ECCAI.

Borst, P., Akkermans, J.M., Pos, A., & Top, J.L. (1995). The PhysSys ontology for physical systems. In *Working Papers of the Ninth Int. Workshop on Qualitative Reasoning QR'95* (Bredeweg, B., Ed.), pp. 11–21. University of Amsterdam.

Brunnermeier, S.B., & Martin, S.A. (1999). *Interoperability Cost Analysis of the US Automotive Supply Chain, Final Technical Report To NIST*. RTI Project No. 7007-03. Research Triangle Park, NC: Research Triangle Institute.

Buchanan, B.G., & Shortliffe, E.H. (1984). Uncertainty and evidential support. In *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristics Programming Project*. Boston: Addison–Wesley.

Chandrasekaran, B., Goel, A., & Iwasaki, Y. (1993). Functional representation as design rationale. *IEEE Computer January*, 48–56.

Clancey, W.J. (1983). The epistemology of rule-based expert system: a framework for explanation. *Artificial Intelligence 20*, 215–251.

Clancey, W.J. (1985). Heuristic classification. *Artificial Intelligence 27(3)*, December.

de Kleer, J., & Brown, J.S. (1983). Assumptions and ambiguities in mechanistic mental models. In *Mental models* (Genter, D., & Stevens, E.L., Eds.), pp. 155–190. Hillsdale, NJ: Erlbaum.

Doraiswamy, S., Krishnamurty, S., & Grosse, I. (1999). Decision making in finite element analysis. *Proc. 1999 Design Technical Conf., DETC99/CIE-9058*, September. Las Vegas, NV: ASME.

Dubois–Perlerin, Y., & Zimmermann, T. (1993). Object-oriented finite element programming: III. An efficient implementation in C++. *Computer Methods in Applied Mechanics and Engineering, 108*, 165–183.

Duda, R.O., Gasching, J., Hart, E., Konolige, K., Reboh, R., Barrett, P., & Slocum, J. (1978). Development of the PROSPECTOR consultation system for mineral exploration, final report. In *SRI Projects 5821 and 6415*. Menlo Park, CA: SRI International.

Dym, C., & Levitt, R. (1991). Toward the integration of knowledge for engineering modeling and computation. *Engineering with Computers 7*, 209–224.

Finn, D., & Cunningham, P. (1994). Physical model generation in PDE analysis using model-based case-based reasoning. *QR '94: 8th Int. Workshop on Qualitative Reasoning About Physical Systems*, pp. 90–97, Nara, Japan, June.

Finn, D., Grimson, J.B., & Harty, N.M. (1992). An intelligent mathematical modeling assistant for analysis of physical systems. *Proc. ASME 1992 Computers in Engineering Conf. Exposition*, Vol. 2, pp. 69–74. San Diego, CA: ASME.

Goel, A., Bhatta, S., & Stroulia, E. (1996*a*). KRITIK: an early case-based design system. In *Issues and Applications of Case-Based Reasoning to Design* (Maher, M., & Pu, P., Eds.). Mahwah, NJ: Erlbaum.

Goel, A., Gomez, A., Grue, N., Murdock, J.W., Recker, M., & Govindaraj, T. (1996*b*). Explanatory interface in interactive design environments. In *Artificial Intelligence in Design* (Gero, J.S., Ed.). Boston: Kluwer Academic.

Grosso, W.E., Eriksson, H., Fergerson, R.W., Gennari, J.H., Tu, S.W., & Musen, M.A. (1999). *Knowledge Modeling at the Millennium (The Design and Evolution of Protégé-2000)*. Stanford's Medical Informatics Report No. SMI-1999-0801. Stanford, CA: Stanford University.

Grower, M.D. (1982). *A Pragmatic Knowledge Acquisition Methodology*. Redondo Beach, CA: TRW Defense Systems Group.

Gruber, T.R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition 5(2)*, 199–220.

Gruber, T., & Olsen, G. (1994). An ontology for engineering mathematics. *Proc. Fourth Int. Conf. Principles of Knowledge Representation and Reasoning* (Doyle, J., Torasso, P., & Sandewall, E., Eds.), pp. 258–269. San Mateo, CA: Morgan–Kaufmann.

Hazelrigg, G.A. (1996). *Systems Engineering: An Approach to Information-Based Design*. New York: Prentice–Hall.
Henson, B., Juster, N., & de Pennington, A. (1994). Towards an integrated representation of function, behavior and form, computer aided conceptual design. *Proc. 1994 Lancaster Int. Workshop on Engineering Design* (Sharpe, J., & Oh, V., Eds.), pp. 95–111. Lancaster: Lancaster University EDC.
Holzhauer, D., & Grosse, I. (1999). Finite element analysis using component decomposition and knowledge-based control. *Engineering with Computers 15*, 315–325.
ISO 10303-104. (1994). *Industrial Automation Systems and Integrations—Product Data Representation and Exchange—Part 104: Integrated Application Resource: Finite Element Analysis*. ISO TC184, SC4. New York: ISO, ISO Technical Committee 184, Subcommittee 4.
Iwasaki, Y., & Chandrasekaran, B. (1992). Design verification through function and behavior-oriented representations: bridging the gap between function and behavior. In *Artificial Intelligence in Design* (Gero, J.S., Ed.), pp. 597–616. Boston: Kluwer Academic.
Java Native Interface Specification. (1997). Cupertino, CA: Sun Microsystems, Inc.
Mackie, R.I. (1992). Object oriented programming of the finite element method. *International Journal for Numerical Methods in Engineering 35*, 425–436.
McDermott, J. (1980). R1: an expert in the computer system domain. In *Proc. National Conf. Artificial Intelligence*, AAAI, pp 269–271.
Miller, R.A., Pople, H.E., & Myers, J.D. (1982). INTERNIST—I: an experimental computer based diagnostic consultant for general internal medicine. *New England Journal of Medicine 306(8)*, 468–476.
Musen, M.A. (2000). Ontology-oriented design and programming. In *Knowledge Engineering and Agent Technology* (Cuena, J., Demazeau, Y., Garcia, A, & Treur, J., Eds.). Amsterdam: IOS Press.
Noy, F. N., & McGuinness, D.L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford, CA: Stanford University.
Paredis, C.J.J., Diaz–Calderon, A., Sinha, R., & Khosla, P.K. (2000). Composable models for simulation-based design. *Engineering with Computers 17*, 112–128.
Peak, R.S. (2000). *X-Analysis Integration Technology*. Technical Report EL002-2000A. Atlanta, GA: Georgia Institute of Technology, Engineering Information Systems Lab.
Qian, L., & Gero, J.S. (1996). Function–behavior–structure paths and their role in analogy based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(4)*, 289–312.
Ranta, M., Mantyla, M., Umeda, Y., & Tomiyama, T. (1996). Integration of functional and feature based product modeling—the IMS/GNOSIS Experience. *Computer-Aided Design 28(5)*, 371–381.
Reed, J.A., & Afjeh, A.A. (1998). An object-oriented framework for distributed computational simulation of aerospace propulsion systems. *Proc. 4th USENIX Conf. Object-Oriented Technologies and Systems (COOTS)*, Santa Fe, NM, April 27–30.
Rohl, P.J., Kolonay, R.K., Irani, R.M., Sobolewski, M., Kao, K., & Bailey, M.W. (2000). A federated intelligent product environment. In *AIAA-2000*, pp. 5–6. Paper No. AIAA-2000-4902.
Shanbhag, S. (2001). *Metaobject based finite element modeling*. MS Thesis. Amherst, MA: University of Massachusetts.
Shanbhag, S., Grosse, I.R., Wileden, J.C., & Kaplan, A. (2001). Meta-object based finite element analysis. Paper No. DETC2001/DAC-21062. *Proc. ASME 2001 Design Automation Conf.* Pittsburgh, PA: ASME.
Sheehy, M., & Grosse, I. (1997). An object-oriented blackboard based approach for automated finite element modeling and analysis of multichip modules. *Engineering with Computers 13*, 197–210.
Sinha, R., Liang, V.C., Paredis, C.J.J., & Khosla, P.K. (2001). Modeling and simulation methods for design of engineering systems. *Journal of Computing and Information Science in Engineering 1*, 84–91.
Shephard, M.S., Bachmann, L., Georges, M.K., & Korngold, E.V. (1990). Framework for reliable generation and control of analysis idealisations. *Computer Methods in Applied Mechanics and Engineering 82(1–3)*, 257–280.
Szykman, S., Fenves, S.J., Keirouz, W., & Shooter, S.B. (2000). A foundation for interoperability in next generation product development systems. *Proc. 2000 ASME Design Engineering Technical Conf.*, Baltimore, MD, September, pp. 10–13. New York: ASME.
Tatsubori, M. (1999). *An extension mechanism for the Java language*. MS Thesis. Ibaraki, Japan: University of Tsukuba.
Tomiyama, T., Kiriyama, T., Takeda, H., & Xue, D. (1989). Metamodel: a key to intelligent CAD systems. *Research in Engineering Design 1*, 19–34.
Turkiyyah, G.M., & Fenves, S.J. (1996). Knowledge-based assistance for finite element modeling. *AI Applications in Civil and Structural Engineering 11(3)*, 23–32.
Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., & Tomiyama, T. (1996). Supporting conceptual design based on the function–behavior–state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10*, 275–288.
van Melle, W. (1978). *A Domain Independent System That Aides in Constructing Consultation Programs*. Report HPP-78-19. Stanford, CA: Stanford University, Computer Science Department.
Wujek, B., Koch, P., McMillan, M., & Chiang, W.-S. (2002). A distributed, component-based integration environment for multidisciplinary optimal and quality design. *9th AIAA/ISSMO Symp. Multidisciplinary Analysis and Optimization*, Atlanta, GA, September.
Zimmermann, T., Dubois–Perlerin, Y., & Bomme, P. (1992). Object-oriented finite element programming: I. Governing principles. *Computer Methods in Applied Mechanics and Engineering 98*, 291–303.

Ian Grosse is an Associate Professor in mechanical engineering and Director of the I-MAD Laboratory at the University of Massachusetts. He received his BS from Cornell University in 1979 and his MS and PhD degrees from Virginia Polytechnic Institute and State University in 1983 and 1987, respectively. Professor Grosse's research interests include interoperability of engineering analysis models, decision-based trade-offs in engineering design and analysis modeling, and the development of object-oriented frameworks for supporting engineering analysis applications. Dr. Grosse also codirects the Center for e-Design and Realization of Engineered Products and Systems, an NSF-sponsored multiuniversity and industry research collaboration.

John Milton–Benoit is Principal Research Engineer for United Technologies Corporation (UTC). He received his BS and MS degrees in mechanical engineering from the University of Massachusetts in 1990 and 1993, respectively. His interests include improved efficiency in innovation processes and interoperability of engineering analysis models, and finite element error analysis for material nonlinear problems. He is currently an Innovation Project Leader at the UTC Research Center in East Hartford, CT.

Jack C. Wileden received his BA degree in mathematics and his MS and PhD degrees in computer and communications sciences from the University of Michigan in 1972, 1973, and 1978, respectively. Dr. Wileden has been on the faculty of the University of Massachusetts at Amherst since 1978 and is currently a Professor in the Department of Computer Science and Director of the Convergent Computing Systems Laboratory. Professor Wileden's research interests are programming languages and interoperability. His work is primarily directed toward producing tools, techniques, and formal foundations to support development and evolution of maximally seamless systems comprising interoperating components.