Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2001), *15*, 281–293. Printed in the USA.
Copyright © 2001 Cambridge University Press 0890-0604/01 $12.50

Using FMEA models and ontologies to build diagnostic models

BURTON H. LEE
Department of Mechanical Engineering, Stanford University, PO Box 19249, Stanford, CA 94309, USA

(Received November 10, 2000; Accepted November 27, 2000)

Abstract

Product design and diagnosis are, today, worlds apart. Despite strong areas of overlap at the ontological level, traditional design process theory and practice does not recognize diagnosis as a part of the modeling process chain; neither do diagnosis knowledge engineering processes reference design modeling tasks as a source of knowledge acquisition. This paper presents the *DAEDALUS* knowledge engineering framework as a methodology for integrating design and diagnosis tasks, models, and modeling environments around a common Domain Ontology and Product Models Library. The approach organizes domain knowledge around the execution of a set of tasks in an enterprise product engineering task workflow. Each task employs a Task Application which uses a customized subset of the Domain Ontology—the Task Ontology—to construct a graphical Product Model. The Ontology is used to populate the models with relevant concepts (variables) and relations (relationships), thus serving as a concept dictionary-style mechanism for knowledge sharing and reuse across the different Task Applications. For inferencing, each task employs a local Problem-solving Method (PSM), and a Model-PSM Mapping, which operate on the local Product Model to produce reasoning outcomes. The use of a common Domain Ontology across tasks and models facilitates semantic consistency of variables and relations in constructing Bayesian networks for design and diagnosis.

The approach is motivated by inefficiencies encountered in cleanly exchanging and integrating design FMEA and diagnosis models. Demonstration software under development is intended to illustrate how the *DAEDALUS* framework can be applied to knowledge sharing and exchange between Bayesian network-based design FMEA and diagnosis modeling tasks. Anticipated limitations of the *DAEDALUS* methodology are discussed, as is its relationship to Tomiyama's Knowledge Intensive Engineering Framework (KIEF). *DAEDALUS* is grounded in formal knowledge engineering principles and methodologies established during the past decade. Finally, the framework is presented as one possible approach for improved integration of generalized design and diagnostic modeling and knowledge exchange.

Keywords: Bayesian Belief Networks; Design-Diagnosis Integration; Design FMEA; Diagnostic Modeling; Knowledge Engineering Framework; Ontologies

1. INTRODUCTION

1.1. Product design and diagnosis

Product design is a complex human problem-solving activity which has as its goal the synthesis and analysis of concepts in an effort to produce a physical or software artifact that satisfies a stated need and set of performance requirements. Product diagnosis, in contrast, seeks to develop methods and mechanisms for analyzing artifact failures, with the goal of understanding the root cause(s) of such failure, and then restoring the artifact to full functionality. Both activities employ symbolic representations—concepts and concept relations, often aggregated into ontologies—in the course of reaching their goals, and both aggregate such representations into higher level groupings—models. Both endeavors are also empirically witnessed to group and subdivide conceptual manipulation efforts into problem-solving processes—"tasks"—each with its own set of problem-solving methods (PSMs).

Product design and diagnosis, however, are, today, worlds apart. Despite strong areas of overlap at the ontological level, traditional design process theory and practice does not recognize diagnosis as a part of the modeling process

Reprint requests to: Burton H. Lee, Department of Mechanical Engineering, Stanford University, PO Box 19249, Stanford, CA 94309, USA. E-mail: blee@cdr.stanford.edu

chain (Pahl & Beitz, 1995); neither do diagnosis knowledge engineering processes reference design modeling tasks as a source of knowledge acquisition. This paper presents the *DAEDALUS* knowledge engineering framework as a methodology for 1) integrating design and diagnosis tasks, models, and modeling environments around a common Domain Ontology and Product Models Library, and 2) improving the semantic consistency within and between models. Building upon the commonalities shared by design and diagnosis activities, even as we recognize their differences, we describe here our preliminary investigations of a product-knowledge engineering framework intended to serve as a knowledge-level integration backbone for general design and diagnosis tasks.

1.2. Design processes and computer-based support

Product design theory and practice literature often segments the design process into three primary stages (Blessing, 1994):

- **Problem definition stage**. The *problem definition stage* consists of all activities that occur prior to the explicit generation of alternative design solutions. It deals with the analysis of the needs, and the formulation of problems, goals, and requirements. This stage results in a problem statement, and a list of requirements, also termed a design specification.
- **Conceptual design stage**. In the *conceptual design stage*, physical principle solutions for the product and its main elements are generated based on the functions which the product must fulfill. The results of the prior problem definition stage form the starting point of the conceptual design stage.
- **Detailed design stage**. The third, *detailed design stage* starts with one or more concepts generated in the second stage, and proceeds to develop the product through a series of concept alternative synthesis and analysis activities. The result of this stage is a full use-case and technical description of the product that contains all information needed for the subsequent life-cycle phases from manufacturing to disposal.

The process of moving the product from concept to artifact proceeds on the basis of a set of design engineering tasks. Design tasks in the conceptual and detailed design stage are often characterized by the construction of symbolic models for the purposes of computer simulation of product behavior and features. These models typically employ an implicit and well-defined group of concepts and concept relations—a product domain ontology—that can be represented either as directed or undirected graph structures, depending on the type of relationships considered, or as mathematical constructs. Such design models are generally quite detailed and highly granular in their use and manipulation of concepts, inasmuch as they seek to predict with great accuracy future performance parameters of the artifact under varying usage conditions. Examples of various design tasks where such graphical models are employed include functional flow, reliability, CAD and Bayesian network-based FMEA (Lee, 2001), and other failure modeling tasks, among others.

Even as such conceptual models and modeling environments assume increasing importance in industrial design engineering practice, however, it is a fact that design modeling environments, particularly CAD systems, possess effectively no computer-based support at the ontological level. Concepts and relations employed are often implicitly defined within the local context, and are stored local to each software application. As a result, it is in general extremely difficult to share and reuse concepts (design variables) and relations (relationships) across design tasks and tools at the fine granularity level, which is often required in design modeling. The absence of any formal local or global ontologies also leads to inconsistency in semantics across tasks and task variables, and hinders updates or extensions to the design project vocabulary or taxonomy in a consistent, scalable manner across the enterprise.

In examining corporate design practice and literature, it is also apparent that design processes generally do not recognize diagnosis-related modeling tasks with two major exceptions. First, control systems design often involves notions of system diagnosability and state observability. Second, embedded systems frequently consider system and software diagnosis in the allocation and assignment of error codes during coding tasks. In the general case, however, system diagnosis is considered outside the standard product design engineering process and is undertaken after the design is completed. This leads to several consequences. Diagnosis teams—generally belonging to the Customer Support Organization—are usually separate from the design team; design models and other data are, as a result, frequently not available to the diagnosis team for use in constructing diagnosis models. As a result of excluding diagnosis from the formal design process chain, the generation and exchange of knowledge between design and diagnosis tasks remains ineffective and inefficient.

1.3. Diagnosis tasks and modeling

If traditional design process theory and practice does not recognize diagnosis as a part of the product modeling process chain, then neither do diagnosis processes include design modeling tasks in their respective view. Traditional diagnosis process stages and tasks are drawn from expert system theory and practice, and include the following:

- **Knowledge elicitation and acquisition**: working with the expert to encode his/her knowledge in a knowledge base, using a chosen knowledge formalism;
- **Knowledge base test and verification**: verification that the encoded knowledge base does, in fact, combine evidences and generally provide inferencing results that are consistent with the expert's knowledge.

This historical focus on the expert as the primary source of relevant knowledge is one reason why the diagnosis community has neglected the use of product design models as a key knowledge input to diagnostic model-building activities. A second reason has been the absence of adequate formal design models that could serve as a basis for the construction of diagnostic models.

Even as design and diagnosis at the modeling level remain far apart, at the ontological level they share many ontologies within the product life-cycle domain. Concepts and relations describing product function, structure (component hierarchy), states, events, reliability, and failures, for example, are common to vocabularies and taxonomies required for building models in both disciplines. The formal development and utilization of ontologies for diagnosis modeling is significantly advanced over that for design modeling, however, due to the relatively strong knowledge engineering focus of the diagnostic research community. Like design modeling environments, diagnostic modeling environments today suffer a comparable absence of practical computer-based support.

For these reasons, design and diagnosis tasks are generally seen by both communities as unrelated tasks belonging to two disparate process chains with little in common at the ontological or modeling levels. The alternative view given in this paper presents design and diagnosis as two sets of related tasks, sharing a common domain ontology and a set of models, within a larger continuum of product engineering knowledge acquisition, generation, and life-cycle management processes.

1.4. Related research

Contemporary process-based computer-supported engineering design research is largely focused on the development of approaches and architectures aimed at solving the problem of proper organization of information systems to support designers in the execution of design tasks, and in the capture and reuse of design data. The Process-based Support System (PROSUS) developed by Blessing (1994) is a leading example of this approach. PROSUS, like many systems proposed by the design research community, is a very high-level system, adopts a traditional view of the design process, and focuses most heavily on the conceptual design stage. Of interest here is that it does not deal explicitly with the problem of design and diagnosis model exchange and sharing, and makes no effort to develop or work with a formal design or design process domain ontology. In this sense, such systems do not address some very fundamental knowledge acquisition, generation, and maintenance issues, preferring to instead take a data-centric viewpoint.

The knowledge-based engineering design research community has taken a recent interest in the development of knowledge-intensive CAD (KIC) and knowledge-intensive design (KID) systems. The goal of such systems is to augment and extend current industrial design tools with a wide range of knowledge, ranging from commonsense knowledge to knowledge of physical processes, artifacts, and behaviors. Tomiyama's Knowledge Intensive Engineering Framework (KIEF) is the best known of these efforts, and relies heavily on an ontological approach to unify and bridge design modeling activities across the full product-knowledge life cycle, from design through manufacturing, operations and maintenance, and disposal (see Section 6; Tomiyama & Hew, 1999). Presumably, this also includes the use of ontologies to bridge design and diagnosis models, although this specific aspect of his method is not explicitly addressed or demonstrated.

Finally, the knowledge engineering community has, since the mid-1980s, been engrossed in the development of comprehensive methodologies, frameworks, and tools for building and organizing knowledge-based and expert systems, and more generally for organizing problem-solving processes and tasks. These initiatives have, from the beginning, paid strong attention to the engineering of diagnostic expert systems, often with a strong reliance on ontologies as a foundational underpinning of the architecture and conceptual framework. The most relevant investigations here are those of the CommonKADS and Protégé projects, which both employ a mix of ontology- and task-based approaches to organizing domain knowledge and problem-solving processes and methods (Studer et al., 1999).

Notwithstanding these various research efforts focused on design process and expert system architectures and computer-based support, the design and knowledge engineering communities have yet to bring design and other knowledge-intensive activities under a common knowledge framework. The emphasis on *passive knowledge elicitation and acquisition* by experts contrasts sharply with the designer-focused *active knowledge generation paradigm* employed in design theory and practice circles, and continues to hinder collaboration between these two communities.

1.5. Ontology-based computer support for design and diagnosis modeling tasks

The *DAEDALUS* framework aims to provide a basis for bridging of design and diagnosis knowledge and tasks through 1) utilizing a common central ontology base (the "Domain Ontology") as a concept dictionary, and 2) capturing newly generated task-level knowledge in the Domain Ontology for sharing across the Product Models Library (see Fig. 1). The investigations presented are motivated by, and build upon, earlier design research results derived from the *PHOENIX* project (Lee, 2001).

The following section thus presents a brief review of the *PHOENIX* project, its goals, and results. In Section 3 we present details of the *DAEDALUS* framework. Section 4 outlines our proposed application of *DAEDALUS* to integrating Bayesian network-based FMEA and diagnostic models around a common Domain Ontology. In Section 5, we discuss limitations of the *DAEDALUS* framework and comment on the applicability of the approach to generalized knowledge sharing and exchange between design and diag-

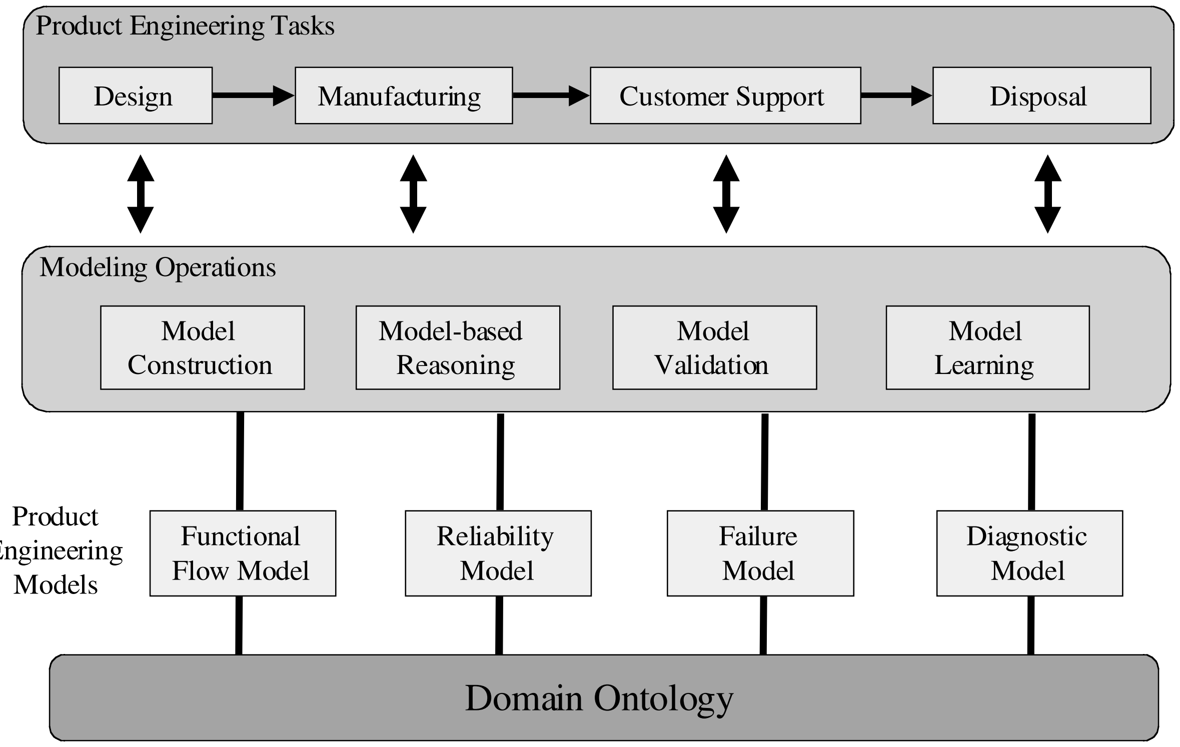

Fig. 1. The *DAEDALUS* framework.

nostic systems. Section 6 reviews some of the related research literature in this area.

2. THE PHOENIX SYSTEM: BUILDING AND EXCHANGING BN-FMEA MODELS

The original motivation for creating the *DAEDALUS* framework comes from our experience in developing a new Bayesian network-based Failure Modes and Effects Analysis (BN-FMEA) system called *PHOENIX*. The *PHOENIX* system was developed and prototyped at Stanford University in an effort to bridge the design–diagnosis modeling gap by moving the construction of belief network-based failure models upstream, that is, forward into design, in the design process chain (Lee, 2001). The industry-standard spreadsheet FMEA modeling task (Bowles, 1998) was selected as the starting point for this exercise, and it was successfully demonstrated that a spreadsheet failure causal model could be expressed in a Bayes net format without loss of information (see Fig. 2).

Failure scenarios ("chains") are constructed out of a set of basic variable types comprising functions, components,

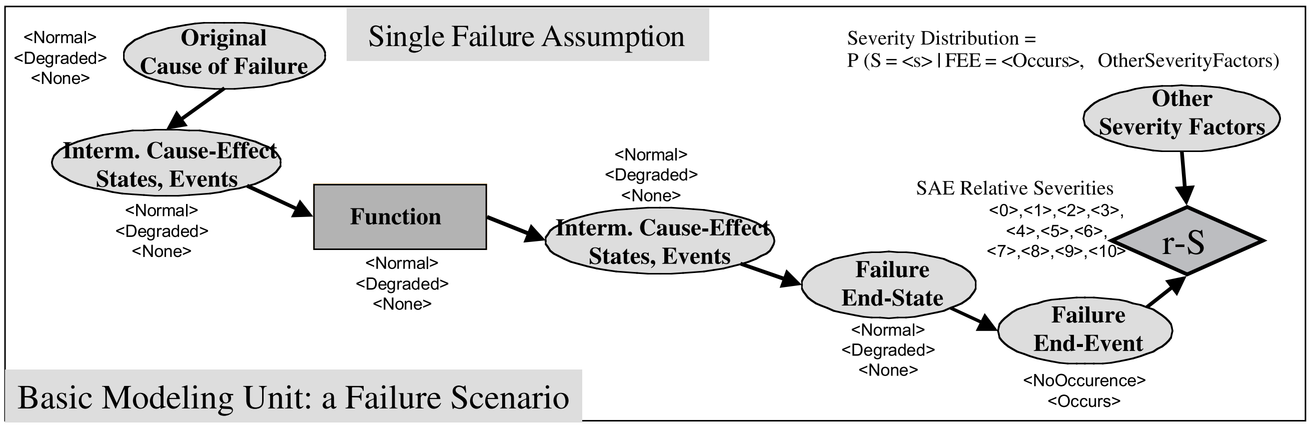
Single Failure Assumption
<Normal> <Degraded> <None>
Original Cause of Failure
Interm. Cause-Effect States, Events
<Normal> <Degraded> <None>
Function
<Normal> <Degraded> <None>
<Normal> <Degraded> <None>
Interm. Cause-Effect States, Events
Failure End-State
<Normal> <Degraded> <None>
Failure End-Event
<NoOccurence> <Occurs>
Severity Distribution = P (S = <s> | FEE = <Occurs>, OtherSeverityFactors)
Other Severity Factors
SAE Relative Severities
<0>,<1>,<2>,<3>, <4>,<5>,<6>, <7>,<8>,<9>,<10>
r-S
Basic Modeling Unit: a Failure Scenario

Fig. 2. Modeling a single-failure scenario in a BN-FMEA model.

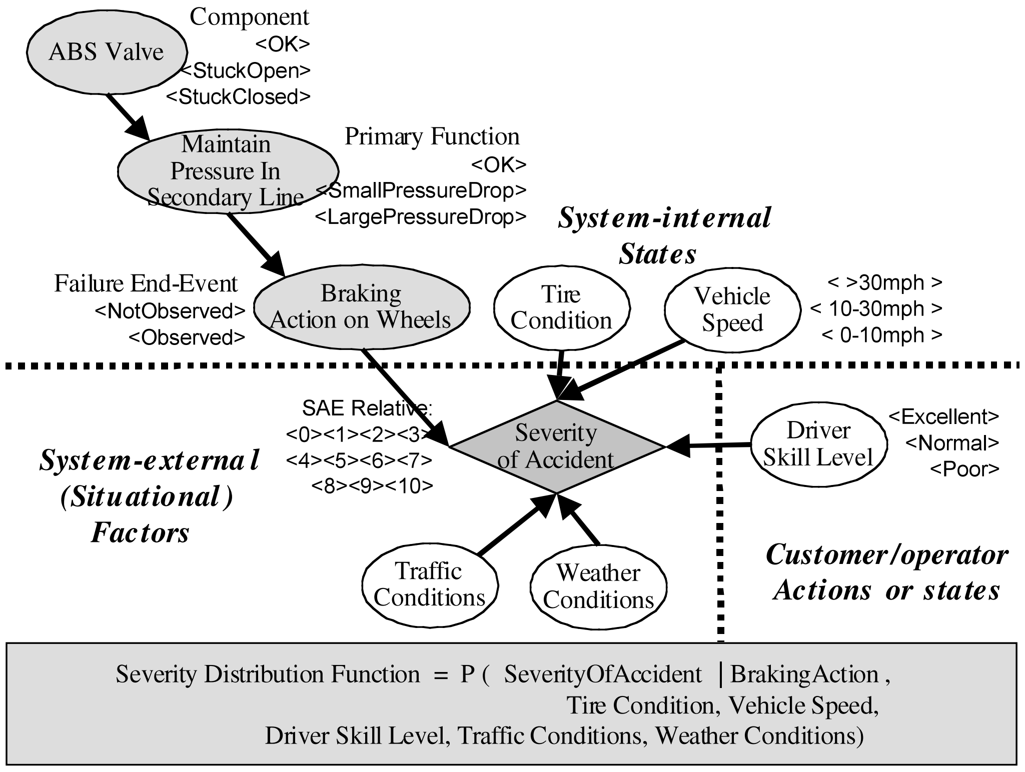

Fig. 3. Relative conditional severity.

system states and events, and physical variables, among others. A new class of severity variable is attached to a variable representing a scenario's failure end-event (FEE); severity variables can have other parents representing system-internal and -external influences on the failure severity in question (see Fig. 3). FMEA models are then assembled from these chains. The BN-FMEA models so produced can be loaded into a standard Bayes network editing environment, where they may be edited by hand from FMEA models into diagnostic (BN-DIAG) models (see Fig. 4). Difficulties in representing and reasoning with function models for mechatronic systems precludes the automated generation of BN-FMEA models from functional schematics. For this reason, it is assumed that the original BN-FMEA model is generated by hand.

Although the work achieved the stated goals of introducing Bayesian network models and inferencing to design-phase engineering analysis tasks, the process of manually exchanging and editing BN-FMEA models into Bayesian network diagnostic (BN-DIAG) models proved cumbersome and haphazard. For large models, the removal by hand of large numbers of severity variables, for example, along with their attendant failure end-events and other parents was unwieldy and subject to error. The problems encountered with this phase of the *PHOENIX* project motivated consideration of a more principled approach—based on ontologies—to the construction and exchange of Bayesian network FMEA and diagnostic models across the design–diagnosis interface.

3. THE *DAEDALUS* FRAMEWORK

The experience with the *PHOENIX* BN-FMEA tool motivated the design and implementation of a second generation environment to explore and illustrate an ontological-level approach to design–diagnosis modeling and knowledge exchange. The *DAEDALUS* framework supports knowledge acquisition, generation, and sharing of BN-FMEA and BN-DIAG models based upon a common Domain Ontology (see Fig. 1). Knowledge acquisition and generation are performed at the task level in a task-specific Task Application through the construction of graphical product models and associated updating of the common ontology; with each Task Application is associated a problem-solving method and a model-to-PSM Mapping (MPM) which is used to perform inferencing with said models.

The software environment is comprised of the following principal elements: The Domain-Task Ontology and the Product Models Library, on the one hand, represent the system domain knowledge. Secondly, a set of software applications manages and facilitates the ontology- and model-building activities; these comprise the Ontology Manager, the Task Applications, the Task Manager, and the Browser Suite (see Fig. 5). After discussing the general requirements

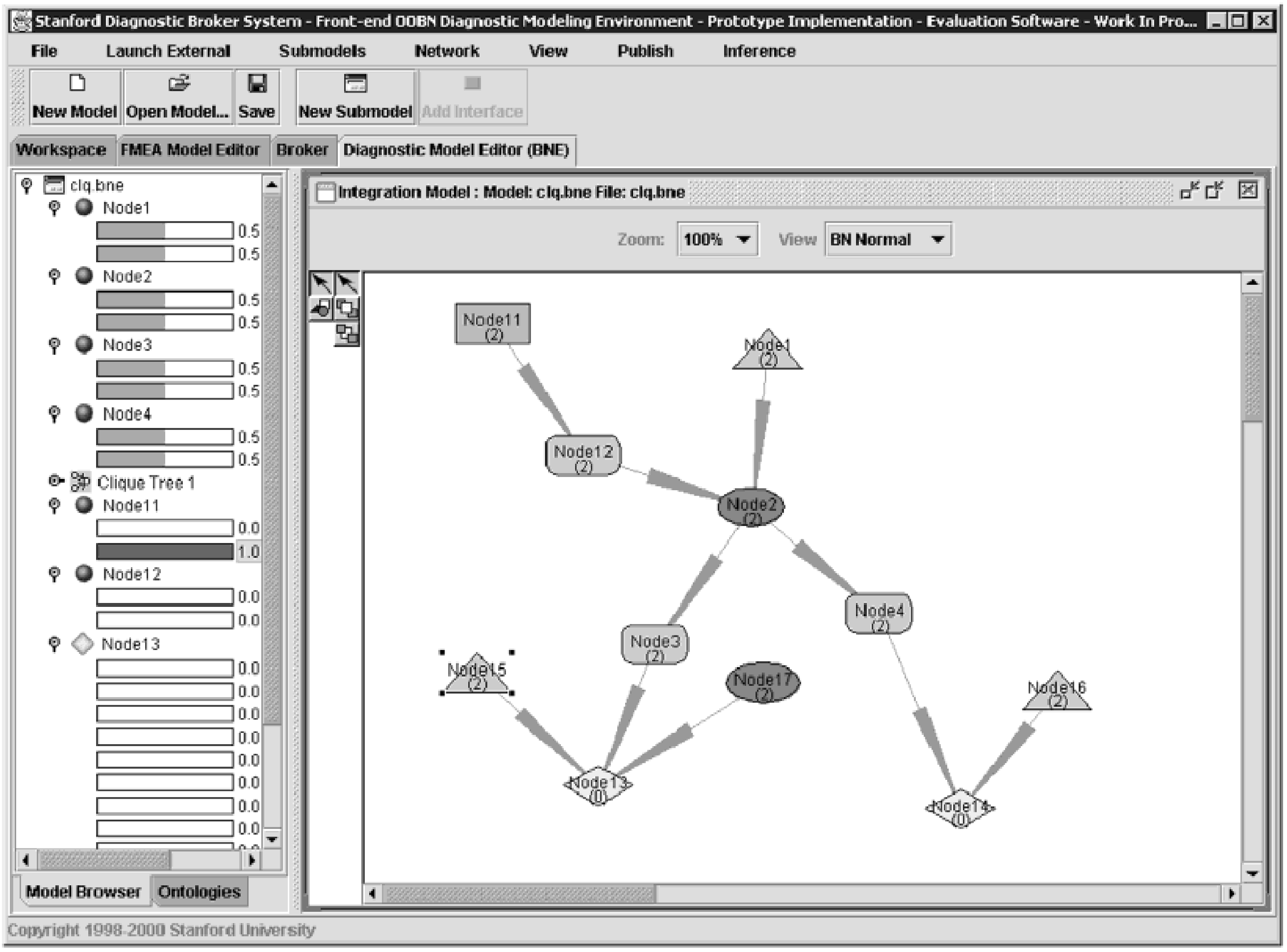

Fig. 4. *PHOENIX* BN-FMEA modeling environment.

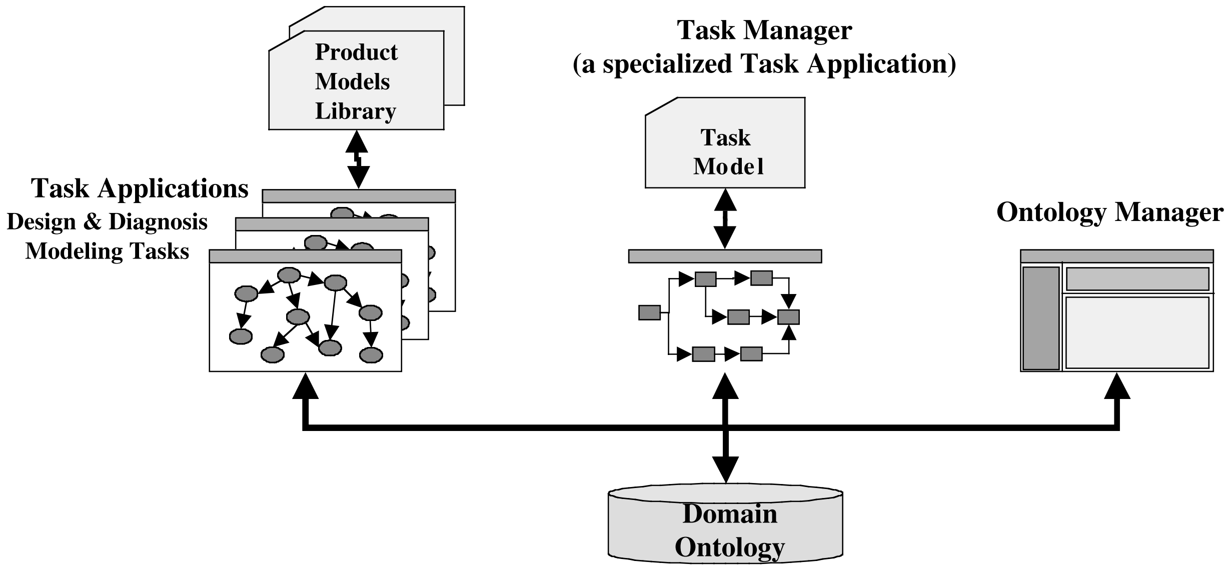

Fig. 5. The *DAEDALUS* framework and architecture.

of *DAEDALUS*, the framework elements are examined in turn.

3.1. System requirements

The goal of the *DAEDALUS* system is to provide ontology-based computer support for artifact design and diagnosis modeling tasks on a task-by-task basis. The system should support generalized knowledge acquisition and generation within and across individual product design and diagnosis tasks, as well as knowledge sharing and reuse at the ontology and model levels. Knowledge generated during a given model construction process should, if possible, be captured by the system for sharing across other tasks and their associated models. The system should be flexible enough to scale to future design and diagnosis tasks that are not now part of industrial practice at most firms, such as function–structure modeling and BN-FMEA modeling.

3.2. The framework elements

The *DAEDALUS* framework is comprised of two main elements: the Domain Knowledge and the Domain Knowledge Managers. We proceed to describe each of these elements.

3.2.1. Domain knowledge

Domain knowledge includes the Domain Ontology and the Product Models Library, comprised of the Application Models and the Task Model. The Ontology and the Models partition domain knowledge between generalized "template" knowledge and specific "case" or "instance" knowledge. The Domain Ontology is the locus for knowledge about domain concepts and relations stored in template form; the Application Models represent knowledge as instances of ontology templates. For any given task, the locus of domain knowledge starts primarily within the Ontology and then shifts to the Model as the task is executed and completed, feeding back into the Ontology as new concepts are created during modeling tasks. This circular flow of knowledge from Ontology to Model to Ontology contrasts with the traditional unidirectional Ontology-to-Model flow in most knowledge-based systems.

Domain and task ontologies. The Domain Ontology is an enterprise-level knowledge base comprising the collection of concepts and relationships employed in the course of the product design project, across all tasks and models. Major subontologies used would include the functional, component, and physical principle and variable ontologies, for example. The Ontology is a hierarchical and dynamic entity that changes and grows through the execution of each Task Application. As a concept dictionary, the Ontology relies on the use of the ontology object name, relative location in the hierarchy, and relationships to other objects to define the semantics of any given object class in the Ontology. In its fully deployed form, we envision that an Ontology Broker entity would manage access to the different subontologies, which together comprise the full Domain Ontology; management of the different subontologies is performed on a local basis by one or responsible organizations (see Fig. 6).

Each Task Application utilizes a relevant subset of the Domain Ontology called the Task Ontology. The Task Ontology is generated from the principal Domain Ontology using a set of Ontology Operators. The operators provide several ontology-level operations using an ontology algebra (Wiederhold, 1994) which can be performed on the Domain Ontology, such as "Allow All," "Add," and "Prohibit." "Allow All" permits access to the full Domain Ontology, while "Add" adds a single subontology or ontology class to the list of currently accessible subontologies; "Prohibit" denies access to a particular subontology or ontology class by a given Task Application. Use of these aggregation and filtering operators permits customization of the Task Ontology for each particular Task Application.

The Ontology Browser takes the Task Ontology as its input (see Section 3.2.2) and aids in navigation and manipulation of the Ontology.

The product models library. The library is comprised of the following model groups and other elements:

- **Product Models**. Each Task Application generates a Product Model. Models are constructed and viewed by the designer in the Task Application's Graphical Object Editor (GOE) using the Task Ontology as the source of its variables and relations (connections between variables). Using the Model-to-PSM-Mapper, the task PSM operates on the Application Model to produce task-specific inferencing results. By working off a common Domain Ontology, Product Models of different tasks can employ a consistent set of variable and variable relation semantics.

  The process of constructing a Product Model can be seen as 1) shifting—or specializing—the locus of knowledge from the Task Ontology to the model, and 2) generating new concepts and relations for inclusion in the Task Ontology. In the course of building a particular model, ontology classes and slots may be added to or deleted from the Task Ontology, or otherwise edited, as variables and variable relationships are added to the model. Variables and relations local to the task which are not initially resident in the Task Ontology must be entered in the Ontology before they can be entered into the model.
- **Task Model**. The Task Model comprises the set of tasks which are required to achieve the design meta-goal; more than one task ordering may satisfy the meta-goal requirements. To each task is assigned one or more Task Applications responsible for 1) building a relevant Product Model, and 2) operating on that model with at least one PSM. The Task Model is constructed by the designer in the Task Manager; alternatively, one

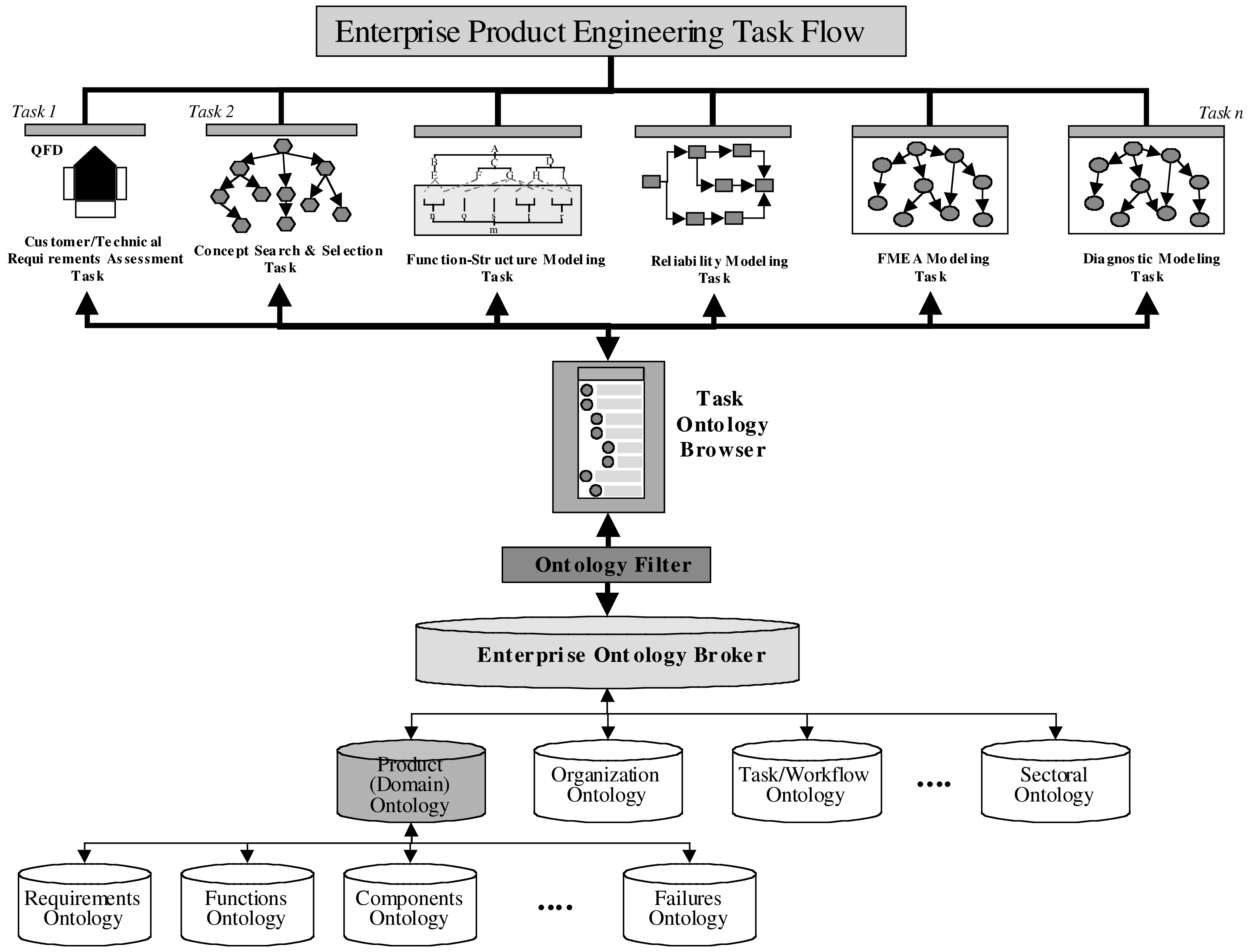

Fig. 6. Enterprise ontology-based architecture of *DAEDALUS*.

or more Task Models may be generated by a Task Planner module. The Browser Suite's Task Browser takes the Task Model as its input, and aids in navigation between tasks (see Section 3.2.2).

3.2.2. Domain knowledge managers

The Knowledge Managers are responsible for capturing, generating, and maintaining the Domain Knowledge at the ontology and task levels, and across the task workflow. The Managers include the Task Applications, the Ontology Manager, the Task Manager and the Browser Suite ("the Suite").

- **Task Applications**. The Task Applications comprise the set of tools used to accomplish the various product modeling tasks in the design project. Each Application includes the following elements: the Graphical Object Editor (GOE), an inferencing Problem-solving Method, a Model-PSM Mapper, and the Product Model (see Fig. 7). Using a set of Application-specific Ontology Operators, it has access to that Task Ontology which is relevant for constructing a given Product Model. For purposes of this investigation, we limit ourselves to considering only Applications which can construct graphical models.
- **Ontology Manager**. The Ontology Manager is responsible for navigating and manipulating the complete Domain Ontology for the design project. Using the Manager, the designer adds, edits, and deletes ontology objects or classes—concepts and relations—from the Domain Ontology, and also edits their attributes (slots) in a similar manner. The Browser Suite's Ontology Browser is a compact subset of the Ontology Manager and may be used for limited editing of the Domain Ontology as well. The Manager has a similar look and feel to the Protégé-2000 software interface.
- **Task Manager**. The Task Manager is a specialized, meta-level Application responsible for generating the Task Model for the design project. The Manager may include a Task Planner that suggests task orderings based on assigned task goals and task constraints such as available resources. Task orderings and changes can be effected only directly in the Task Manager, however, and not in the Task Browser which can only view the Task Model.
- **Browser Suite**. The Browser Suite is responsible for collecting in a single simple interface the management and selection of the Task Model, Product Model, and the Task Ontology. The Task Browser takes the Task Model as its input and is used to navigate between different tasks and their associated applications. Task-level changes, however, can be effected only directly in the Task Manager, and not in the Task Browser. The Model Browser shows a given Task Application's Product Model, and facilitates modifications to and navigation around such model. The Ontology Browser displays the Task Ontology and aids in navigation and manipulation of the Ontology. The Suite is designed to enable rapid and easy navigation between the different Browsers and their respective models.

The Task Application and the Ontology Browser are used together. Variables and relations to be used in the graphical editor are obtained from the Task Ontology through drag-and-drop actions on individual ontology objects in the Browser (see Fig. 7). Model variables which are not present in the Task Ontology must first be added to the Ontology via the Browser before they can be dropped into the editor and used in a given model. Variable attributes, such as name and states, are initially assigned default values which can be edited by the designer.

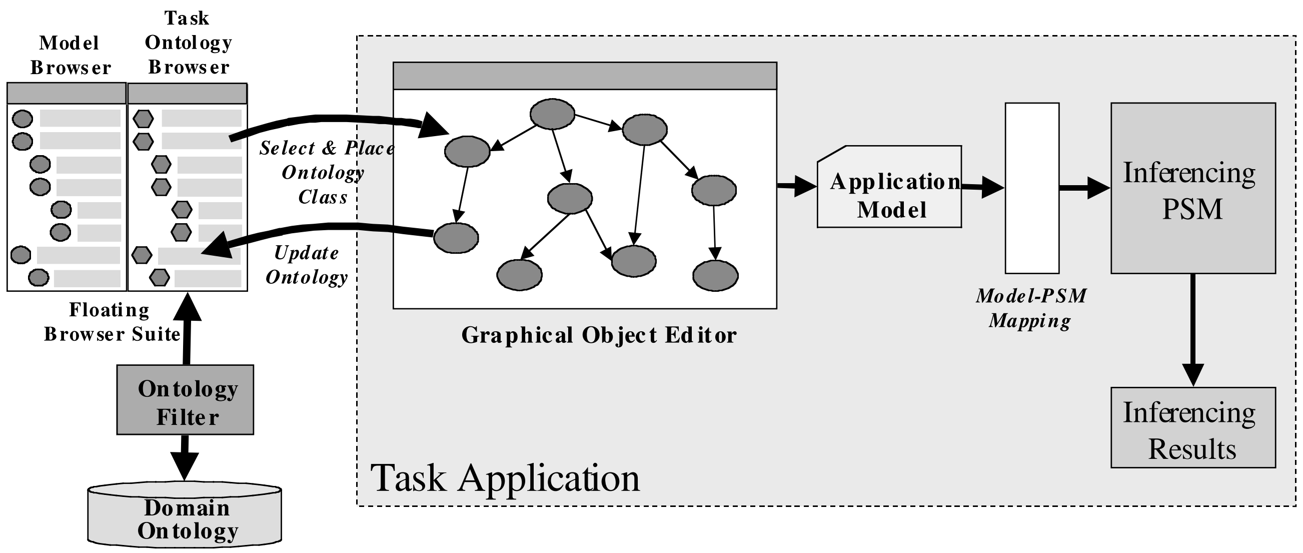

Fig. 7. Task Application architecture, with Browser Suite.

4. PROPOSED APPLICATION OF THE *DAEDALUS* FRAMEWORK

We propose to initially apply the *DAEDALUS* framework to two design and diagnosis tasks, namely the BN-FMEA design FMEA modeling task and the BN-DIAG diagnostic modeling task. These tasks are selected from the perspective of 1) demonstrating a clean ontology-level integration across the design–diagnosis modeling interface, and 2) working within a single knowledge representation and reasoning schema for simplicity' sake, namely Bayesian belief networks.

Once it is demonstrated that the *DAEDALUS* architecture can support knowledge exchange across two tasks which share a common Bayesian inference PSM, it is anticipated that the method can be extended to more complex knowledge sharing between tasks with different PSMs and knowledge representation formalisms. Application of the *DAEDALUS* architecture to the design–diagnosis problem creates a common underlying knowledge and modeling framework which facilitates semantic consistency, model exchange, and common ontology management.

4.1. The BN-FMEA modeling task

The BN-FMEA design FMEA task is configured as an instance of a Task Application with a Bayesian reasoner for its PSM inferencing module. The graphical editor is specialized for the construction of belief network graphs; using it, the designer generates a BN-FMEA Application Model in which the Model slots are BN-variables (i.e., variables with Bayesian states and a conditional probability table) and BN-relations (i.e., Bayesian parent–child causal relations). As previously discussed in Section 2, the BN-FMEA model is constructed by the designer with a set of functional, component, and physical quantity variables at its core; its leaves contain failure end-event and severity variables and their parents (see Fig. 8). The Model is mapped to the inferencing PSM via a Model-Bayes Reasoner mapping module that generates an input-directed acyclic graph to the PSM. The FMEA Criticality Matrix are treated as views on the outputs of the inferencing PSM. The initial version of *DAEDALUS* seeks to reuse as much code from *PHOENIX* as possible, such that the PSM is essentially the same Bayesian inference engine used in *PHOENIX*.

The BN-FMEA Task Application employs a Task Ontology that results from applying the operation "Allow All" on the product Domain Ontology, followed by the ontology operation "Prohibit" on the *Error_Code* subontology. This produces a Task Ontology that contains the following subontologies of direct relevance to this task, among others: *Functions*, *Components*, *Failure_States*, *Failure_Events*, and *Severities*. The BN-FMEA task is restricted from accessing system error codes because they are not typically required in FMEA modeling and analysis. The *Error_Codes* subontology is subsequently made available to the BN-DIAG Task

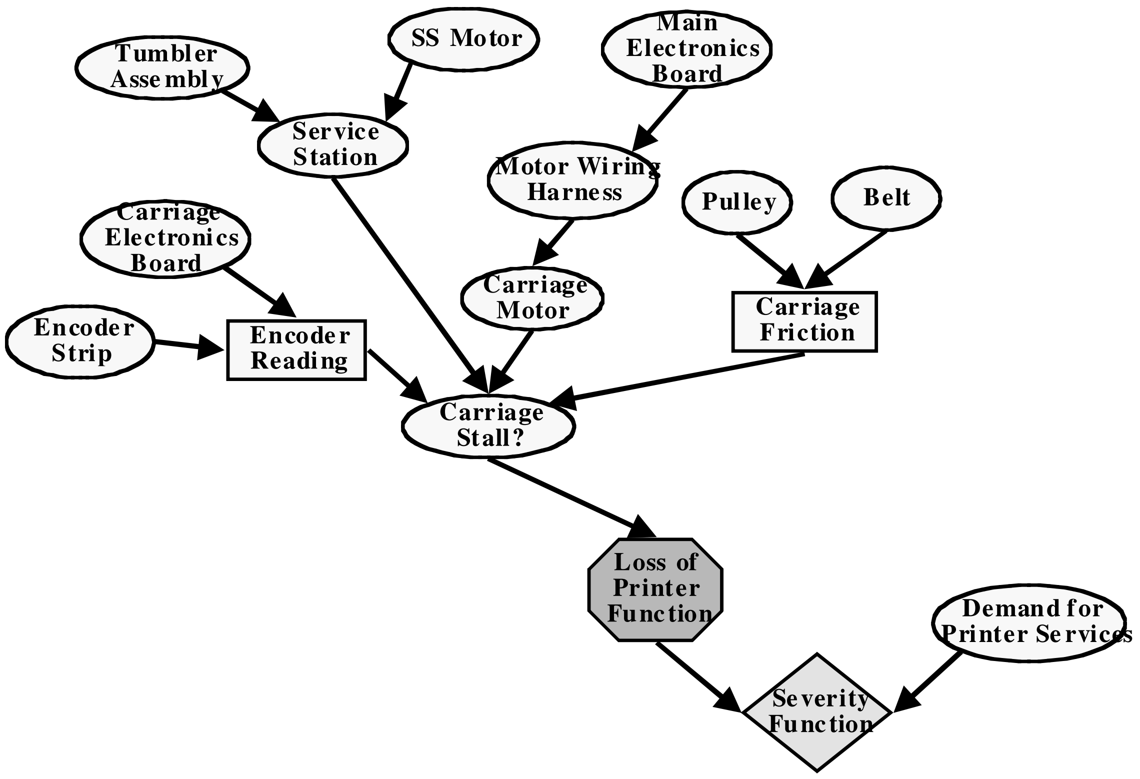

Fig. 8. Portion of a BN-FMEA Model for an inkjet printer.

Ontology and models (see Section 4.2). It is assumed that this task is executed and completed before the BN-DIAG task, such that the full BN-FMEA Application Model is available to the BN-DIAG task.

4.2. The BN-DIAG modeling task

The BN-DIAG diagnostic modeling task is configured in a similar manner to the BN-FMEA task, except that 1) a different Task Ontology is employed, and 2) the Criticality Matrix is eliminated. Here, the inferencing PSM is the same Bayesian reasoner used in the BN-FMEA task, and the graphical editor is also used to build belief network graphs. The Application Model employs the same knowledge representation, and the same Model-Bayes Reasoner mapping module; the previously generated BN-FMEA model is employed as the initial input to the editor.

In contrast to the BN-FMEA task, the BN-DIAG Task Application employs a Task Ontology that results from applying the ontology rules "Allow All" on the Domain Ontology, followed by the "Prohibit" operation on the *Failure_Severities* and *Failure_End_Event* subontologies; other subontologies may also be prohibited. This produces a Task Ontology that contains the following subontologies of direct relevance to this task, among others: *Functions*, *Components*, *States*, *Events*, and *Error_Codes*. The BN-DIAG task is restricted from working with severity and failure end-event variables because they are typically not required in diagnostic modeling and analysis. The BN-DIAG Task Application is not restricted from working with system *Error_Codes* because they are typically of paramount interest for diagnostic modeling and analysis.

In our use-case scenario, construction of the product BN-DIAG model proceeds after the BN-FMEA model is completed in the BN-FMEA Task Application (see Fig. 9). First, the BN-DIAG Task Application is opened, along with its associated Task Ontology. The BN-FMEA model is imported into the BN-DIAG application, first passing through the application's Task Ontology filter. This strips the incoming model of all severity variables, along with their associated relationships (arcs), as well as the failure end-event variables and their arcs. In addition, any "hanging variables" which have been left standing without any associated relationships are pruned; these include any non-*FEE* parent variables of severity variables.

Filtering out the BN-FMEA leaf variables preserves the core model variables and relationships intact, for example, the component–function combinations and the various system state–event variable combinations, among others. The BN-DIAG model is now ready for construction by the diagnostic modeling engineer, who adds relevant observational variables such as Error Codes and other system state indicators from the Task Ontology. Notions of system and component function, states, and events are preserved across both models and constitute the underlying consistent knowledge representation bridge between the two tasks and models.

5. DISCUSSION

The *DAEDALUS* approach is, in principle, generalizable to other product design and diagnostic tasks and knowledge representation and reasoning formalisms. The use of ontologies for generating and selecting variables in the construc-

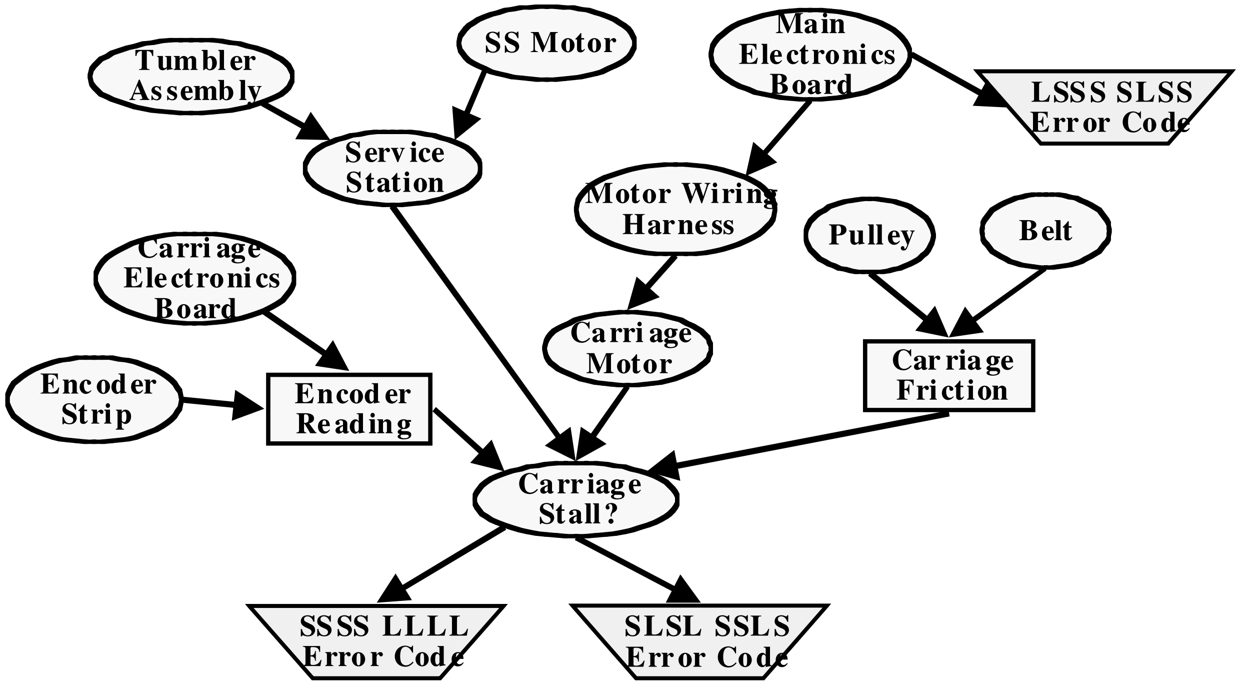

Fig. 9. Portion of a BN-Diagnostic Model for an inkjet printer.

tion and editing of models is a general enough method that it can be used in working with other classes of design and diagnosis models. Secondly, using domain ontologies to filter "in" and "out" selected variables and relationships from models also appears useful in contexts outside the immediate BN-FMEA schema. Third, the inferencing PSM's and their associated domain model mappings can be extended beyond Bayesian networks to other reasoning methods such as bond graphs and constraint-based problem-solving methods.

The *DAEDALUS* framework in its current form addresses model-to-model consistency only at the "shallow" semantic level, variable-by-variable and relation-by-relation. Domain Ontology semantics are informally specified only by variable/relation names and their relative location in the ontology context, and not through the use of any formal Knowledge Interchange Format (KIF)-like first order logic language. This notion and use of ontology can be termed a "sparse" ontological approach, in contrast to the "rich" ontological methodology which employs a formal semantics encoded in logic. By utilizing the Domain and Task Ontologies principally as a KIEF-style concept dictionary, semantic consistency of variables and relationships can only be maintained informally with those in other models. Exact reasoning consistency, however, is not necessarily maintained with this approach, since the inference PSMs and their mappings are not maintained as a part of the Domain Ontology at this stage. The absence of a formal semantics precludes the detection of ontological conflicts in the models or Domain Ontology.

6. RELATED WORK

Using design-phase descriptions or models of physical systems to construct diagnostic procedures or models has been a subject of research for several years. Genesereth (1984), for example, demonstrated in his DART program the use of digital circuit design models containing information about intended structure (parts and their interconnections) and expected behavior (equations or rules relating inputs to outputs) to generate sets of failure candidates from observed symptoms. The domain of model-based diagnosis (Hamscher et al., 1992) uses design-based knowledge of device function, structure, and behavior to do diagnostic reasoning about device malfunction, although this is also typically limited to examples of digital and analog circuitry. More recently, Srinivas (1994) has employed functional design schematics to directly generate Bayesian network diagnostic models of circuits. These approaches have proven difficult to apply to mechatronic artifacts due to the complexity associated with describing and modeling their function, structure, and behavior.

Other work relating product design and diagnosis is the design-for-diagnosability metrics research undertaken by Paasch (Murphy & Paasch, 1997). Here the goal is to identify optimal design alternatives during design or redesign, based on a score ranking the relative ease of isolating the cause of a loss of functionality for each design option. A traditional, spreadsheet-based FMEA model is used to generate a fault tree, from which diagnosability metrics for individual line replaceable units are calculated. This approach is intended for cases where "deep" design models are generally not available, such that a "shallow" metrics-based approach can be applied.

The Knowledge Intensive Engineering Framework proposed by Tomiyama is similar to the *DAEDALUS* framework in several respects. Both propose to integrate disparate design tools by using a central knowledge base and a set of differentiated design modeling environments. Both knowledge bases comprise a master ontology defining concepts used throughout the system (KIEF calls this the "Concept Dictionary"), as well as a library of product models (the "Models Library") that are used by the different modeling environments. KIEF differs from *DAEDALUS*, however, in two key respects. First, its knowledge base also contains a Physical Feature Library which represents and describes physical phenomena and related mechanisms. Secondly, KIEF utilizes a model management mechanism (the "Metamodel") to manage the interactions and data sharing between the different models and modeling systems, particularly where models are interdependent and require consistency (Sekiya & Tomiyama, 1999; Yoshioka et al., 1999).

DAEDALUS can thus be viewed as a specialized subset of the KIEF, in which 1) only the Concept Dictionary and the Model Library are employed (corresponding to the Domain Ontology and the Product Model Library, respectively), 2) the design modeling environments are specialized to models which can be described and constructed as graphs, and 3) model-to-model consistency is implicitly assumed and therefore not handled. The proposed application of *DAEDALUS* to integrating design and diagnosis modeling knowledge via Bayesian networks takes the architecture proof-of-concept in a direction unexplored by the KIEF.

CommonKADS and Protégé are two knowledge engineering research initiatives that have emerged recently with a wide user base of their respective methodologies. The CommonKADS framework distinguishes three different types of knowledge needed to solve a given task: domain-specific knowledge (including a domain ontology), inference-process knowledge (problem-solving methods—PSMs), and task knowledge (a decomposition of tasks into subtasks and inference actions). The domain-specific knowledge is clearly separated from the inference and task knowledge in order to facilitate reuse of domain knowledge and PSMs. CommonKADS is intended as a knowledge acquisition and management framework for the development of traditional knowledge-based systems and does not deal with knowledge generation issues associated with artifact design (Schreiber et al., 2000). Protégé focuses on the use of ontologies for the acquisition of knowledge. A domain ontology defines the concepts and relationships that are used within the domain knowledge base; the method ontology associated with a given PSM defines concepts and relationships used by the PSM for reasoning. Both PSMs and domain knowledge bases are intended as reusable components. The notion of mediators or mappings between PSMs and knowledge bases is proposed. Protégé also limits itself to the development of traditional knowledge-based systems with the attendant restrictions on handling of newly generated knowledge during model construction operations (Studer et al., 1999).

7. CONCLUSIONS

This paper describes the *DAEDALUS* framework and the motivation behind its development. *DAEDALUS* is a knowledge engineering system which employs a central Domain Ontology as a concept dictionary in support of multiple product-modeling environments (Task Applications); these tools are organized around graph-based modeling tasks in an enterprise task workflow. Domain knowledge is partitioned between the Domain Ontology and a Product Models Library. Problem-solving methods and Model-to-PSM Mappers are attached to each Task Application for inferencing support.

An initial verification of the methodology is proposed using the *PHOENIX* project's Bayesian belief network-based FMEA modeling approach and a standard Bayes network diagnostic modeling environment. The goal of the proof-of-concept is to demonstrate knowledge sharing and reuse between a design-phase task (BN-FMEA) and a knowledge engineering-phase task (BN-DIAG diagnosis modeling). Limiting the PSM's to Bayesian inference engines and their associated mappings reduces the degrees of freedom of the experiment without impairing its ability to demonstrate the use of ontologies for model aggregation, filtering, and customization. The approach bears certain similarities to Tomiyama's (1999) KIEF framework, but extends the methodology with Bayesian networks, a specialization of modeling tasks to design FMEA and diagnosis, and the adoption of a formal knowledge engineering architecture drawn from recent work in the CommonKADS and Protégé knowledge engineering communities.

ACKNOWLEDGMENTS

I am pleased to thank the following university, industrial, and U.S. government sponsors of this research: the Alliance for Innovative Manufacturing at Stanford (AIMS), DaimlerChrysler AG, and the U.S. Department of Energy, Integrated Manufacturing Fellowship Program. I also acknowledge my research team for their ongoing assistance in coding of the demonstration software: Janet Tang, Nick Ho, and Adil Sherwani.

REFERENCES

Blessing, L.T.M. (1994). *A process-based approach to computer-supported engineering design*. Dissertation. Enschede, The Netherlands: University of Twente. Cambridge, UK: Black Bear Press Ltd.

Bowles, J. (1998). The new SAE FMECA standard. *Proc. of the 1998 Annual Reliability and Maintainability Symposium*. pp. 48–53. IEEE Press.

Genesereth, M.R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence 24*, 411–436.

Hamscher, W., Console, L., & de Kleer, J. (1992). *Readings in model-based diagnosis*. San Mateo, CA: Morgan Kaufmann Publishers.

Lee, B.H. (2001). Using Bayes belief networks in industrial FMEA modeling and analysis. *Proc. of the 2001 Annual Reliability and Maintainability Symposium* (RAMS 2001), Philadelphia, PA: IEEE Press.

Murphy, M.D., & Paasch, R. (1997). Reliability centered prediction technique for diagnostic modeling and improvement. In *Research in engineering design*, Vol. 9, pp. 35–45. London, UK: Springer Verlag London Limited.

Pahl, G., & Beitz, W. (1995). *Engineering design: A systematic approach*. Berlin: Springer Verlag.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., & Wielinga, R. (2000). *Knowledge engineering and management: The CommonKADS methodology*, Cambridge, MA: MIT Press.

Sekiya, T., & Tomiyama, T. (1999). Case studies of ontology for the knowledge intensive engineering framework. *Knowledge Intensive Computer Aided Design* (*Proc. of the Third Workshop on Knowledge Intensive CAD*, Working Group 5.2 of the International Federation for Information Processing (IFIP), December 1998), (Finger, S., Tomiyama, S., & Mantyla, M., Eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands.

Srinivas, S. (1994). A probabilistic approach to hierarchical model-based diagnosis. *Proc. of the Tenth Conference of Uncertainty in Artificial Intelligence* (UAI '94). Morgan Kaufmann Publishers, San Francisco, CA.

Studer, R., Fensel, D., Decker, S., & Benjamins, V.R. (1999). Knowledge engineering: Survey and future directions. *Proc. of the 5th German Conference On Knowledge-based Systems*, Wuerzburg, Germany, March 1999, Lecture Notes in Artificial Intelligence (LNAI), vol. 1570, Springer-Verlag, Berlin.

Tomiyama, T., & Hew, K.P. (1999). Knowledge intensive computer aided design: Past, present and future. *Knowledge Intensive Computer Aided Design* (*Proc. of the Third Workshop on Knowledge Intensive CAD*, Working Group 5.2 of the International Federation for Information Processing (IFIP), December 1998) (Finger, S., Tomiyama, T., & Mantyla, M., Eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands.

Wiederhold, G. (1994). "An Algebra for Ontology Composition." Proceedings for the 1994 Montery Workshop on Formal Methods.

Yoshioka, M., Shamoto, Y., & Tomiyama, T. (1999). An application of the knowledge intensive engineering framework to building foundation design. *Knowledge Intensive Computer Aided Design* (*Proc. of the Third Workshop on Knowledge Intensive CAD*, Working Group 5.2 of the International Federation for Information Processing (IFIP), December 1998) (Finger, S., Tomiyama, T., & Mantyla, M., Eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands.

Burton H. Lee is a Ph.D. candidate in Mechanical Engineering at Stanford University, with a dissertation research topic of knowledge-based design and diagnosis of mechatronic systems. From 1997 to 1999, he was a member of the technical staff of the Hewlett-Packard Solutions Services Division, Diagnostics Research and Development Laboratory. He possesses an M.S. in Mechanical Engineering (1991) and an M.S. in Engineering Management (1987) from Stanford; his A.B. (1978) in Physics and Economics is from Brown University. Lee is a current U.S. Department of Energy (DOE) Integrated Manufacturing Fellow, a DaimlerChrysler Fellow, and also was a recipient of the NASA Graduate Research Fellowship (1988–1991).