A Rational Reconstruction of INTERNIST-I using PROTÉGÉ-II

Mark A. Musen, John H. Gennari and Wesley W. Wong
Section on Medical Informatics
Stanford University School of Medicine
Stanford, CA 94305-5479, U.S.A
<musen, gennari, wwong@camis.stanford.edu>

PROTÉGÉ-II is a methodology and a suite of tools that allow developers to build and maintain knowledge-based systems in a principled manner. We used PROTÉGÉ-II to reconstruct the well-known INTERNIST-I system, demonstrating the role of a domain ontology (a framework for specification of a model of an application area), a reusable problem-solving method, and declarative mapping relations in creating a new, working program. PROTÉGÉ-II generates automatically a domain-specific knowledge-acquisition tool, which, in the case of the INTERNIST-I reconstruction, has much of the functionality of the QMR-KAT knowledge-acquisition tool. This study provides a means to understand better both the PROTÉGÉ-II methodology and the models that underlie INTERNIST-I.

1. Introduction

Despite substantial progress in the development of principled methodologies for the construction of knowledge-based systems, the vast majority of such systems continue to be built using software-engineering approaches of the 1970s. Most developers equate knowledge engineering with the elaboration of large, poorly structured rule bases; the manner in which the rules interact in the course of problem solving is rarely well defined, and system maintenance becomes a matter of updating, deleting, or adding new rules or other data structures to the amorphous knowledge base. In most situations, systems are built using rapid-prototyping approaches, and knowledge-base components rarely can be reused to engineer new systems [1].

For several years, researchers in our laboratory have been defining a comprehensive methodology and a set of tools for building knowledge-based systems. Our approach, known as PROTÉGÉ-II [2], allows developers to construct new knowledge-based systems from reusable components, and makes explicit many of the epistemological assumptions that are hidden in the design of traditional knowledge-based systems. We have used the PROTÉGÉ-II methodology to build systems in a wide variety of application domains, including protocol-based medical care, determination of possible three-dimensional conformations of bimolecular structures, and configuration of elevators based on architectural and engineering constraints [3]. Although our experience suggests that significant advantages accrue when developers build and maintain knowledge-based systems using PROTÉGÉ-II, many of the distinctions that we make about system architecture and about the development life cycle are not always intuitive to software engineers who have not thought about these kinds of abstractions previously. PROTÉGÉ-II may thus seem overly complicated and arcane at first glance.

A particularly helpful way to understand new approaches is to visualize how they might apply to an example that already is relatively familiar. INTERNIST-I is a well-known knowledge-based system that was developed at the University of Pittsburgh in the 1970s [4, 5]. The program was reengineered in the 1980s to become Quick Medical Reference (QMR), which now is widely distributed commercially for use on personal computers [6]. The structure of the INTERNIST-I knowledge base, and the strategy with which INTERNIST-I addresses the task of diagnosing patient conditions given a set of disease manifestations, are both well described in the literature [4, 5, 7]. Use of PROTÉGÉ-II to reconstruct the INTERNIST-I system consequently provides a useful case study by which to understand the principles of knowledge-base development supported by the PROTÉGÉ-II methodology.

2. Knowledge Engineering with Protégé-II

PROTÉGÉ-II comprises both a methodology for engineering knowledge-based systems and a suite of computer-based tools that support that methodology [2]. In PROTÉGÉ-II, the steps for developing a knowledge-based system include the following: (1) defining a **domain ontology** that describes the classes of concepts in the application area and attributes of elements of those classes; (2) constructing a **problem-solving method** that provides a computational strategy for solving the task to be automated; (3) modifying the domain ontology, if necessary, to create an **application ontology** that may include additional distinctions about the domain knowledge required by the problem-solving method; (4) defining **mapping relations** that identify how elements of the application ontology can satisfy the data requirements of the problem-solving method; (5) generating automatically a **knowledge-acquisition tool** from the application ontology, which allows developers to enter the detailed content knowledge needed to create new knowledge bases.

0195-4210/95/$5.00

Our approach supports the use of libraries of reusable components that facilitate the construction of new knowledge-based systems. In particular, both domain ontologies and problem-solving methods can be archived and adapted for use in new applications [1].

Our approach also emphasizes the automatic generation of a domain-specific knowledge-acquisition tool directly from the application ontology. The resulting knowledge-acquisition tool is custom-tailored to the application area at hand, and supports the entry and maintenance of the content knowledge (instances of classes in the application ontology) required to fill out a complete knowledge base. Construction of a domain ontology—and modifying that ontology, if necessary, to create an application ontology—are creative processes that are best performed by experienced analysts who are proficient at conceptual modeling; the entry and maintenance of content knowledge, however, is a task that often can be performed independently by application specialists who have no knowledge of programming. PROTÉGÉ-II thus offers a divide-and-conquer approach, in which analysts work with content experts to construct the appropriate ontologies, and then the content experts can use the corresponding knowledge-acquisition tools to create the required knowledge bases.

3. RECONSTRUCTION OF INTERNIST-I

We used PROTÉGÉ-II to develop a knowledge-based system that reproduces much of the behavior of INTERNIST-I. Our reconstruction is based on written descriptions of INTERNIST-I [4, 5, 7] and on a version of the INTERNIST-I knowledge base that had been made available to our research group in 1984. We did not have access to a working implementation of INTERNIST-I, and thus we had no way to compare the behavior of our reconstructed system to that of the original program. Our goal, however, is not to reproduce precisely all the nuances of INTERNIST-I, but rather to demonstrate how developers might build a system like INTERNIST-I using the PROTÉGÉ-II approach.

There are a variety of problem-solving approaches that can be used to automate the task of medical diagnosis. Recent work on formal probabilistic inference, for example, has led to computationally efficient reasoning strategies that are quite different from the ad hoc heuristic methods devised for the original INTERNIST-I system. Some researchers might argue that, given current advances in diagnostic reasoning methods, one would not want to build new knowledge-based systems fashioned in the manner of INTERNIST-I. We believe, however, that appreciation for modern computational approaches can best be achieved by means of familiar examples.

3.1 Domain Ontology

The first step in constructing a knowledge-based system using PROTÉGÉ-II is to create a domain ontology. The domain ontology specifies the classes of concepts that are relevant in the application domain, attributes of instances of those classes, and data types for all attributes. The domain ontology thus defines concepts in the application area (e.g., the notion of "diseases"), but does not mention any *instances* of those concepts (e.g., specific diseases such as "tuberculosis"). A tool known as MAÎTRE allows developers to build and edit these ontologies. Figure 1 shows our domain ontology for the INTERNIST-I application.

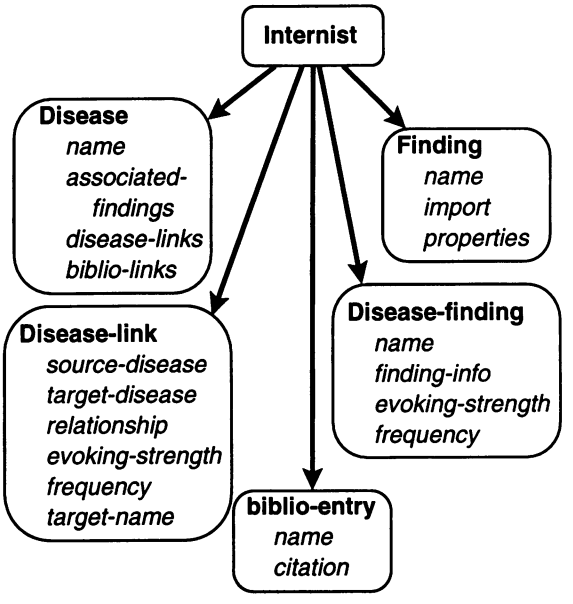

Figure 1. The domain ontology for INTERNIST-I. Each box refers to a different class in the ontology. Attributes of classes appear in *italic* font.

Our INTERNIST-I domain ontology is relatively simple, with classes that specify the notions of diseases, findings, and so on. Note that one class represents "findings" as isolated entities, whereas another class ("disease-finding") represents the *relations* between instances of findings and instances of diseases (defining the corresponding *frequencies* and *evoking strengths*). The "disease-finding" class thus provides a specification for the knowledge contained in INTERNIST-I *disease profiles*. The domain ontology does not include any information about individual diseases; knowledge-base developers enter descriptions of disease instances using the domain-specific knowledge-acquisition tool that PROTÉGÉ-II generates programmatically from the class desciptions in the domain ontology.

Although these were not features of the original INTERNIST-I knowledge base, our domain ontology defines the additional concepts of disease–disease links and of bibliographic citations that justify particular disease–finding relationships. These concepts have become important components of the QMR knowledge base [6].

The knowledge base of INTERNIST-I contains a variety of special *properties* that define ad hoc relationships among disease findings [7]. These properties streamline the manner in which the program asks questions of its users, and assure that the program's requests for laboratory stud-

ies are consistent with a medically appropriate and cost-effective work-up. These properties are useful only in the context of the original INTERNIST-I algorithm, however. Our domain model omits these distinctions because the properties have not been conceptualized in a manner that allows them to be reused by alternative problem-solving approaches. Although we have not done so, it would be appropriate to create a PROTÉGÉ-II *application ontology* that includes descriptions of these concepts, since application ontologies are intended to be mapped to specific problem-solving methods.

3.2 Problem-Solving Method

PROTÉGÉ-II supports a library of domain-independent problem-solving methods. Our goal is to be able to reuse these methods to build new applications. For example, one of our methods—a constraint-satisfaction problem solver known as *propose-and-revise*—has been used to solve the tasks of both configuring new elevators and proposing plausible three-dimensional conformations for ribosome subunits [3]. None of the previously developed problem-solving methods in the PROTÉGÉ-II library was suitable for the INTERNIST-I task. We therefore had to program such a method expressly for this purpose. We constructed a new method, known as *quasi-probabilistic abduction*, based on descriptions of the INTERNIST-I algorithm presented in the literature [4, 5].

We can view the program code that implements the quasi-probabilistic-abduction method as a "black box." The only aspect of the method that system builders must understand is the *method ontology*, which defines in a declarative manner all the data on which the problem-solving method operates (Figure 2). The method ontology for the quasi-probabilistic-abduction method defines the inputs to the problem-solving method and the data *stores*, which are used internally when the method executes. For example, the "working-hypothesis" store contains the dynamic list of hypotheses that the problem-solving method is considering at any given time. The method ontology—which also is created using the MAÎTRE tool—constitutes a complete data model for the method.

The quasi-probabilistic abduction method is potentially reusable because, like all problem-solving methods in the PROTÉGÉ-II library, it is linked to the domain knowledge on which it operates via explicit *mapping relations*. These mapping relations are defined by the analyst who uses PROTÉGÉ-II. In the case of the INTERNIST-I task, one mapping relation declares that instances of the class "all-hypotheses" in the method ontology are derived from a simple transformation of instances of the "disease" class in the domain ontology; another mapping indicates that instances of "findings-list" in the method ontology simply are "findings" in the domain ontology. The PROTÉGÉ-II user specifies mappings between the domain ontology and the method ontology; a mapping interpreter applies these declarative mappings to the domain knowledge classes and

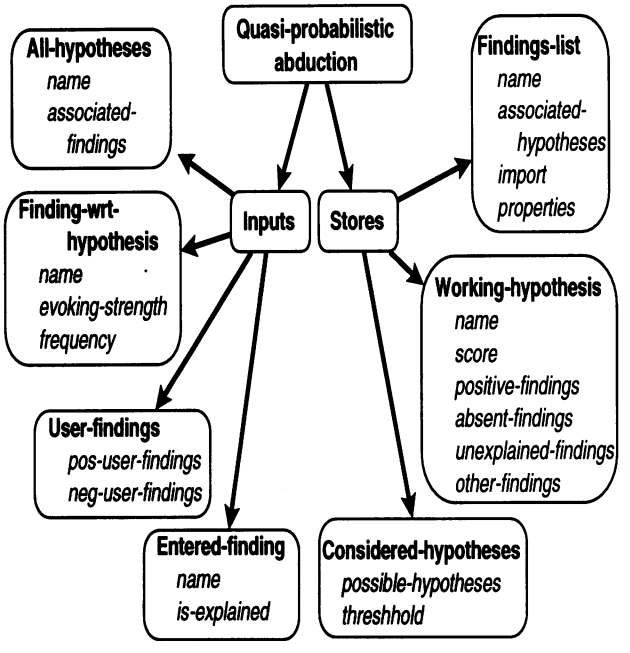

Figure 2. The method ontology for quasi-probabilistic abduction. The inputs to the method include the complete set of possible hypotheses and the findings associated with the current case. The *stores* are data elements required internally for problem solving. The output of the method comprises instances of the "working-hypothesis" class.

instances so that the problem-solving method accesses the appropriate data elements defined in the method ontology.

Traditional knowledge-based systems distinguish between domain knowledge and a reusable inference engine. This separation, although important, does not allow reusability of problem-solving methods and of domain ontologies, as is possible with PROTÉGÉ-II. Typical expert-system–building shells require the developer to fashion a problem-solving method implicitly from the primitives available in the data elements (e.g., production rules) on which the inference engine operates. The problem-solving method thus becomes inextricably bound up with the same data elements that the developer used to represent domain knowledge. For example, it is extremely difficult to examine the production rules in the MYCIN system and to identify how the rules might relate to the heuristic-classification problem-solving method—just as it is nearly impossible to examine the MYCIN knowledge base and to discern the ontology of infectious diseases. The use of explicit domain ontologies, method ontologies, and mapping relations in PROTÉGÉ-II allows developers to regard domain ontologies and problem-solving methods as well-defined building blocks for the creation of intelligent systems; the construction of explicit mapping relations allows developers to "glue together" reusable domain ontologies and problem-solving methods when assembling new applications.

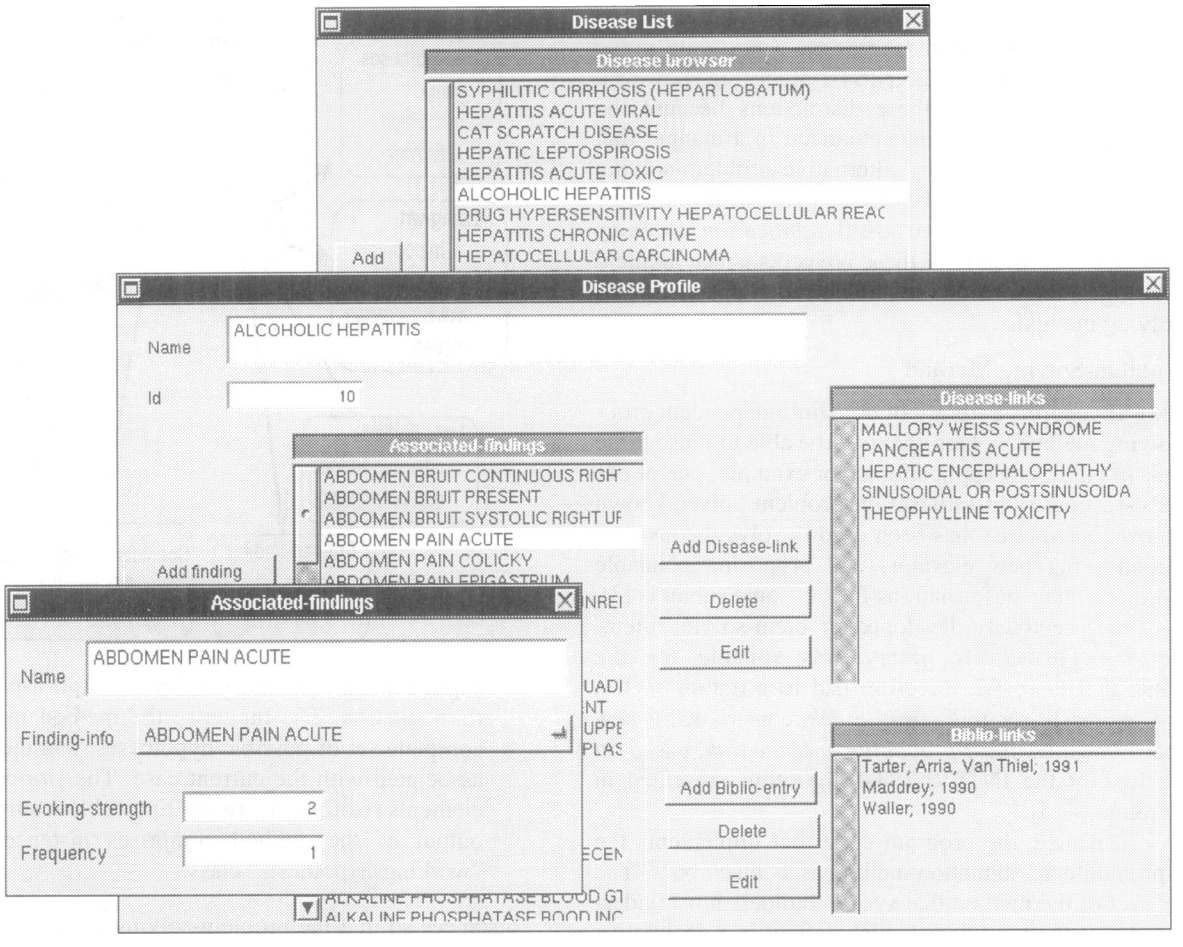

Figure 3. Knowledge-acquisition tool for the reconstructed version of INTERNIST-I. The DASH system generated this tool directly from the domain ontology shown in Figure 1. Here, the user enters information concerning the disease *alcoholic hepatitis*.

3.3 Knowledge-Acquisition Tool

An ontology describes only the classes of concepts relevant for a given purpose, but provides no information regarding specific *instances* of those classes. In PROTÉGÉ-II, system builders enter descriptions of instances using a domain-specific knowledge-acquisition tool that is generated automatically by a subsystem known as DASH. The DASH program takes as input an application ontology (derived from a domain ontology; see Figure 1) and generates as output a corresponding knowledge-acquisition tool (Figure 3). The knowledge-acquisition tool for the INTERNIST-I domain allows developers to enter and edit information regarding individual disease profiles, manifestations of diseases, and bibliographic citations that support the knowledge-base entries.

The knowledge-acquisition tool created automatically by PROTÉGÉ-II from the domain ontology has much of the functionality of QMR-KAT, the knowledge-acquisition tool that Giuse [8] hand crafted to facilitate maintenance of the QMR knowledge base. QMR-KAT, however, incorporates a number of useful, special-purpose features that are not present in the DASH-generated knowledge-acquisition tool (e.g., access to the QMR "term completer" to aid disambiguation of partially entered strings). Whereas the knowledge-acquistion tool generated by PROTÉGÉ-II assures that a user's entries have correct data types and makes it clear to the user when information is missing, the tool does not perform other semantic consistency-checking funtions performed by the original QMR-KAT tool. To allow such consistency checking in other than an *ad hoc* manner would require a means by which developers could define appropriate constraint axioms within the domain ontology; a theorem prover within the knowledge-acquisition tool could then verify that those constraints are not violated by a user's entries. The current PROTÉGÉ-II system does not provide this functionality, however. Despite this limitation, the knoweledge-acquisition tool generated using the PROTÉGÉ-II approach has the significant advantage that, if the domain ontology for the INTERNIST-I task should ever change, it is a relatively trivial matter to create a new

knowledge-acquisition tool that is consistent with the altered knowledge-base format.

4. Discussion

We have used the PROTÉGÉ-II knowledge-acquisition methodology to reconstruct a version of the INTERNIST-I system. Our approach requires development of a declarative domain ontology, definition of mappings between the domain ontology and that of a reusable problem-solving method (in this case, quasi-probabilistic abduction), and automated generation of a domain-specific knowledge-acquisition tool for entry and editing of content knowledge.

Our goal in this work has not been to duplicate the INTERNIST-I system, but rather to demonstrate how a principled approach to knowledge acquisition and maintenance can be applied to the reengineering of a well-known artifact. Although we have not yet tested the behavior of the resulting system using complete clinical cases, our reconstructed version of INTERNIST-I does produce output that demonstrates diagnostic reasoning similar to that shown in published transcripts of INTERNIST-I sessions.

The use of explicit domain and method ontologies aids system builders in understanding the semantics both of the INTERNIST-I knowledge base and of the quasi-probabilistic abduction method. The approach makes it straightforward for developers to map the INTERNIST-I domain knowledge onto *alternative* diagnostic problem-solving methods that we may program in the future (e.g., a belief-network algorithm), or to modify the domain ontology in Figure 1 to generate new knowledge-acquisition tools that can be used to enter the content knowledge needed to support other diagnostic tasks. Moreover, the quasi-probabilistic abduction method is itself potentially reusable for a variety of tasks that involve determination of possible faults when given a set of abnormal symptoms.

Because the PROTÉGÉ-II architecture allows a problem-solving method to invoke other problem-solving methods to solve subtasks posed by the first method [1], it would be possible to embed our quasi-probabilistic abduction method within a larger assembly of problem-solving methods that addressed a more comprehensive application task—such as protocol-directed treatment planning. An overarching problem-solving method responsible for determining the appropriate treatment plan could then invoke the quasi-probabilistic abduction method to solve subtasks that require diagnostic reasoning. In this context, the same method that we used to automate the INTERNIST-I task might now infer the presence of patient conditions that could affect the selection and application of appropriate therapy.

The ability of developers to use PROTÉGÉ-II to select and assemble reusable components from libraries, and to modify those components to meet new system requirements, offers considerable flexibility—not only as an initial knowledge-based system evolves, but also as new systems are conceived and built. Our reconstruction of INTERNIST-I using PROTÉGÉ-II provides a domain ontology, a comprehensive knowledge base, and a new diagnostic problem-solving method with which we can conduct further experiments on the development and reuse of knowledge-base components.

Acknowledgments

This work has been supported by NLM grants LM05157 and LM05304, and by NSF Young Investigator Award IRI-9257578. John Egar and Thomas Rothenfluh performed the initial work to reconstruct INTERNIST-I using the PROTÉGÉ-II architecture, and provided valuable observations that helped us to refine our knowledge-acquisition system. Randy Miller provided us with access to a version of the INTERNIST-I knowledge base.

References

[1] Musen, M.A. Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* **25**: 435–467, 1992.

[2] Musen, M.A., Gennari, J.H., Eriksson, H., Tu, S.W., and Puerta, A.R. PROTÉGÉ-II: Computer support for development of intelligent systems from libraries of components. *Proceedings of Medinfo'95*, pp. 766–770, Vancouver, BC, 1995.

[3] Gennari, J.H., Altman, R.B., and Musen, M.A. Reuse with PROTÉGÉ-II: From elevators to ribosomes. *Proceedings of the ACM-SIGSOFT Symposium on Software Reusability,* pp. 72–80, Seattle, WA, 1995.

[4] Pople, H.E. Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnosis. In: Szolovits, P., ed. *Artificial Intelligence in Medicine*. pp. 119–185, Boulder, CO:Westview Press, 1982.

[5] Miller, R.A., Pople, H.E., and Myers, J.D. INTERNIST-I, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine* **307**:468–476, 1982.

[6] Miller, R.A., McNeil, M.A., Challinor, S.M., Massarie, F.E., and Myers, J.D. The INTERNIST-1/QUICK MEDICAL REFERENCE project—status report. *Western Journal of Medicine* **145**:816–822, 1986.

[7] Massarie, F.E., Miller, R.A., and Myers, J.D. INTERNIST-I properties: Representing common sense and good medical practice in a computerized medical knowledge base. *Computers and Biomedical Research* **18**:458–479, 1985.

[8] Giuse, D.A., Giuse, N.B., and Miller, R.A. Towards computer-assisted maintenance of medical knowledge bases. *Artificial Intelligence in Medicine* **2**:21–33, 1990.