EON: A Component-Based Approach to Automation of Protocol-Directed Therapy

Mark A. Musen, Samson W. Tu, Amar K. Das, and Yuval Shahar

Section on Medical Informatics
Stanford University School of Medicine
Stanford, California 94305-5479

Abstract

Provision of automated support for planning protocol-directed therapy requires a computer program to take as input clinical data stored in an electronic patient-record system, and to generate as output recommendations for therapeutic interventions and laboratory testing that are defined by applicable protocols. This paper presents a synthesis of research carried out at Stanford University to model the therapy-planning task, and to demonstrate a component-based architecture for building protocol-based decision-support systems.

We have constructed general-purpose software components that (1) interpret abstract protocol specifications to construct appropriate patient-specific treatment plans; (2) infer from time-stamped patient data higher-level, interval-based, abstract concepts; (3) perform time-oriented queries on a time-oriented patient database; and (4) allow acquisition and maintenance of protocol knowledge in a manner that facilitates efficient processing both by humans and by computers. We have implemented these components in a computer system known as EON. Each of the components has been developed and evaluated independently. We have evaluated the integration of the components as a composite architecture by implementing T-HELPER, a computer-based patient-record system that uses EON to offer advice regarding the management of patients who have AIDS. A test of the reuse of the software components in a different clinical domain demonstrated rapid development of a prototype application to support protocol-based care of patients who have breast cancer.

1 Component-Based Software and Protocol-Based Care

The complexity of developing robust software systems has led to the emergence of new approaches to computer programming in which software can be made more modular and in which potentially reusable functionality can be encapsulated as discrete components. Although the ability to reuse most computer software to develop new application is extremely limited,[1] researchers are creating novel software architectures that may reduce the enormous costs of software development. The advent of communication standards such as CORBA and OLE that allow distributed objects and software modules to interoperate across programs and across networks are important advances that provide an infrastructure for this work. This object-level infrastructure alone, however, is not sufficient to address the demands of workers in medical informatics, who seek to develop general solutions to recurring computational problems in medicine, and to ease the development and maintenance of complex clinical systems. For clinical software to be reusable and maintainable in a proficient manner, higher-level software abstractions are necessary to capture the computational requirements of the stereotypic tasks that are found in the clinical setting; we therefore must address the content and functionality of the corresponding software components, not just their form and interoperability.

Protocol-based care is a complex medical problem area that entails a number of tasks. Each of these tasks may be addressed by one or more self-contained software components. Development of such component-

based software systems for automation of protocol-directed therapy has been a major focus of work in our laboratory. We have built decision-support systems for protocol-based care in domains such as oncology,[2, 3] hypertension,[4] AIDS,[5] and diabetes.[6] Construction of these diverse systems has allowed us to identify the computational requirements of automating protocol-based care in a wide range of application areas, and to test the reusability of the software components that we have built. Our work has led us to recognize that protocol-based care does not necessitate automation of only a single task. At any given moment, clinicians may be interested in tasks such as identifying patients who are potentially eligible for new protocols or guidelines, retrospectively auditing of past care, obtaining high-quality advice concerning the appropriate interventions to make, or creating high-level summaries of patient data. The variety of possible tasks suggests that system builders need a flexible and robust framework for constructing multiple applications easily. So that we can avoid the high cost of building knowledge bases from scratch, applications should share as much domain knowledge as possible across all the required tasks.[7] For similar tasks in different medical specialties, developers should at least be able to reuse the problem-solving algorithms, modified appropriately, for these new application areas. Thus, a general approach to the development of protocol-based decision-support systems should incorporate both a means to represent the domain knowledge in a manner that facilitates reuse of that knowledge in new situations, and a methodology for the rapid construction of knowledge-based applications from reusable system components.[8]

Our analysis of dozens of protocols for clinical trials and for clinical practice guidelines suggests that automation of protocol-based care requires a computational architecture that (1) supports alternative problem-solving approaches that address the different tasks associated with the use of clinical protocols; (2) facilitates creation of temporal abstractions from point-based patient data; (3) allows querying of those data and abstractions in a well-structured manner; and (4) permits clinicians to browse, enter, and edit protocol descriptions that can be used by the computer-based problem solver. In this paper, we describe an architecture for protocol-based decision support known as EON. Our architecture integrates four components that satisfy these requirements (Figure 1). The components are: (1) problem-solving systems that interpret the abstract protocol specifications, and that apply those specifications to individual patients; (2) a temporal-reasoning system, RÉSUMÉ, that can infer from time-stamped patient data the presence of higher-level, interval-based, abstract concepts; (3) a temporal query system, Chronus, that can perform time-oriented queries on a time-oriented patient database; and (4) the domain-specific knowledge bases (i.e., protocol specifications) required by the other components.

Each of the components in the EON architecture as been presented previously in the literature. Each element has been carefully evaluated on actual clinical data; the results of those evaluations have been reported elsewhere.[2, 6, 9, 10, 11] Description and evaluation of the individual components, however, do not convey how these components interact in a coordinated fashion to address the more general goals of automating protocol-based therapy. Whereas previous publications have taken a reductionistic view of each component in the EON architecture, our goal in this paper is to describe the functionality of the architecture overall. We present EON as a unified approach that facilitates acquisition and maintenance of protocol knowledge, and that addresses the complex computational requirements of protocols in the range of clinical domains that we have examined.

2 The Problem-Solving Architecture

The functional requirements for automating protocol-based care are addressed by the components of the EON architecture. These components represent independent, reusable software modules that developers can assemble to build systems that solve tasks related to the administration of protocol-directed therapy.

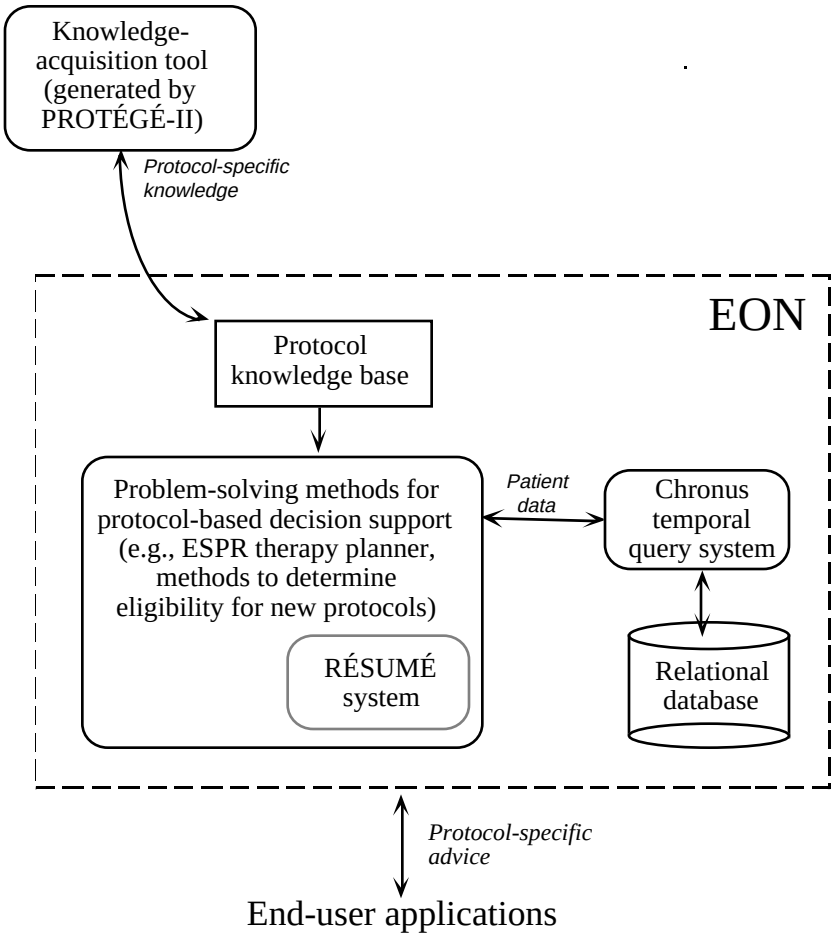

Figure 1: The EON architecture comprises a set of components that interoperate to automate various tasks associated with protocol-based care. These components include modular, problem-solving methods, such as the *episodic skeletal-plan refinement* (ESPR) method, which incorporates the RÉSUMÉ temporal-abstraction system.

Before discussing the specific components, however, it is necessary to describe the design principles that underlie the EON system.

2.1 Problem-Solving Methods and Domain Ontologies

Developers of knowledge-based systems increasingly have sought to construct software architectures in which the purpose of each entry in the knowledge base is unambiguous, and in which discrete, well-understood problem-solving procedures operate on explicit, declarative models of the relevant application domain.[8] The problems of unpredictable behavior in rule-basedrepresentations, [12] and the difficulties of long-term maintenance of large knowledge bases encoded as rules,[13] are widely recognized. Contemporary methodologies for the development of knowledge-based systems thus abandon the use of traditional rule-based frameworks, and instead emphasize the use of software modules that generate well-defined problem-solving behaviors.[14]

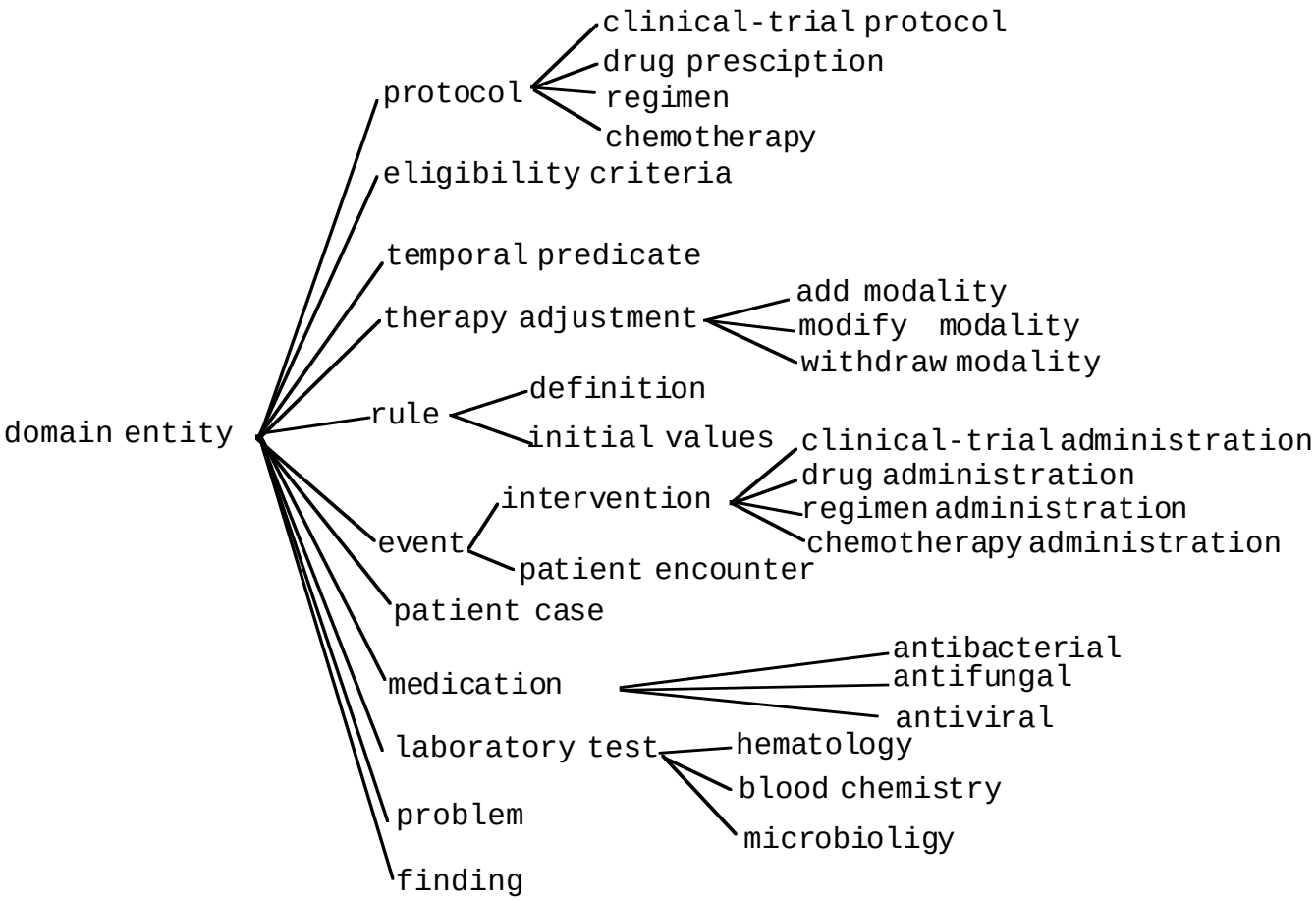

Figure 2: Part of the domain ontology for protocol-based care of patients who have AIDS.

In our laboratory, we have developed a framework for building knowledge-based systems known as PROTÉGÉ-II.[7, 9, 15] PROTÉGÉ-II provides a methodology and an extensive set of tools that allow developers to build knowledge-based systems from reusable components in a principled manner. When using PROTÉGÉ-II, system builders first specify a domain model that defines the terms and relationships of the general application area—for example, the domain of protocol-based care for AIDS patients, which includes concepts such as protocols, medication prescriptions, laboratory tests, and so on. Because the domain model specifies abstract relationships that are always true, independent of particular states of affairs (e.g., the relationships between protocols in general and drugs in general), but does not define specific relationships that might be or might not be true (e.g., how the drug zidovudine is used in a particular protocol), we refer to the model as an **ontology** of the application area.[16, 17, 18] An ontology comprises a set of classes of concepts, and relationships among those concepts, that may be useful for constructing models of specific application tasks. Our ontology of AIDS protocols, for example, defines concepts such as clinical trials, drug regimens, medication prescriptions, laboratory tests, and relationships among them (Figure 2). The ontology, appropriately, does not indicate *which* drug regimens might be used in a particular protocol; the ontology specifies only the classes of concepts that are relevant in an application area, rather than the instances of those classes. In the PROTÉGÉ-II approach, a computer-based tool uses the domain ontology to generate automatically a domain-specific knowledge-acquisition tool that clinicians can use to enter the details of specific protocols, thus allowing the clinicians to instantiate the concepts defined in the ontology for the specific protocols that they wish to encode (see Section 6).

System builders who use PROTÉGÉ-II must select from a library of predefined **problem-solving methods** a domain-independent procedure (or set of procedures) that can automate the application task to be solved. Problem-solving methods are implemented in software modules that address well-described computational tasks, such as constraint satisfaction, classification, or planning. A problem-solving method—like a piece of software from a mathematical-subroutine library—provides a standardized computational algorithm that can be reused in a variety of contexts. A mathematical subroutine generally has an easily understood goal (e.g., calculate a hyperbolic cosine, or perform a fast Fourier transform), where the relationships between the inputs to the subroutine and the outputs conform with specific mathematical relationships (e.g., the output is the fast Fourier transform of the input). A problem-solving method, on the other hand, often has a more conceptually abstract goal (e.g., select the most likely classification given a set of input data, generate a plan given a set of constraints, generate an abstraction given a set of primary data), and has inputs and outputs that conform with certain logical, often nonmathematical relationships (e.g., the output is a set of classifications of the input data that satisfies a set of classification rules that also are inputs to the method). One of the problem-solving methods that we employ in EON is used to plan protocol-directed therapy. As we shall describe in Section 3, this problem-solving method, known as *episodic skeletal-plan refinement*, implements an extremely general control strategy that recursively decomposes abstract plan specifications into constituent specifications, which themselves may be decomposable into specifications of still-further-refined granularity.

A problem-solving method may define a number of *subtasks* that must be addressed by yet other problem-solving methods (Figure 3). Atomic problem-solving methods that do not pose further subtasks are called problem-solving *mechanisms* (following a naming convention adopted by other authors[19]).

When using PROTÉGÉ-II, developers must indicate how the data requirements of the domain-independent problem-solving methods (e.g., skeletal plan specifications, temporal abstractions) relate to the various concepts defined in the relevant domain ontologies (e.g., clinical protocols, episodes of anemia). Constructing a knowledge-based system is thus a matter of defining explicit mappings between the concepts represented in the domain ontology (e.g., the ontology of AIDS protocols) and the abstract data requirements of the particular problem-solving method that has been chosen to solve the application task at hand (e.g., episodic skeletal-plan refinement).[20] The data requirements of the problem-solving method are like the formal parameters of a subroutine in a procedural programming language; to construct a working knowledge-based system, developers must specify how those formal parameters can be satisfied by particular elements of the domain knowledge and by the run-time input data, which are always defined in terms of the relevant domain ontology. Figure 4, for example, depicts diagrammatically how concepts in a clinical-protocol domain ontology map to the requirements of the episodic skeletal-plan–refinement method.

Thus, in the PROTÉGÉ-II approach, system builders (1) define a domain ontology that enumerates the concepts in the application area, (2) configure a problem-solving method that implements some stereotypic control strategy, and (3) map the concepts defined in the domain ontology onto the data requirements of the problem-solving method. The result is that intelligent systems can be constructed from reusable building blocks (i.e., domain ontologies and problem-solving methods) in a flexible, yet principled, manner. The ensuing component-based architecture clarifies the role that each entry in the knowledge base plays in problem solving, thus aiding ongoing maintenance of the system.

Both the problem-solving methods in the present PROTÉGÉ-II library and the domain ontologies that system builders map to those methods are encoded in CLIPS, a popular C-based knowledge-representation language developed by NASA. As a result, PROTÉGÉ-II ultimately generates knowledge-based systems that execute in C, aiding developers who might wish to embed those systems within a larger software framework.

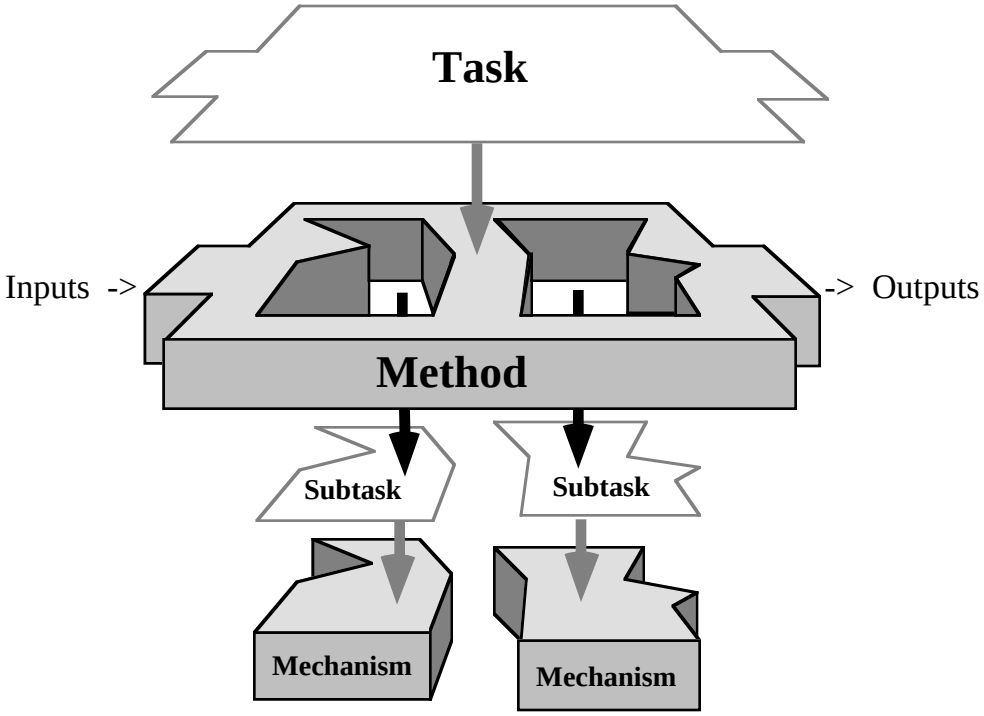

Figure 3: PROTÉGÉ-II automates application tasks by using domain-independent *problem-solving methods* that operate on domain-specific knowledge. Problem-solving methods may designate specific subtasks (i.e., computational goals), which themselves may be solved by yet other problem-solving methods. An atomic problem-solving method that does not entail additional subtasks is a problem-solving *mechanism*.

2.2 Problem Solving in the EON System

The architecture that we have designed for automating protocol-based therapy is constructed from reusable problem-solving methods that are generic components of the PROTÉGÉ-II method library. The current EON system (see Figure 1) includes several problem-solving methods. We have developed generic problem solvers for determining patients' eligibility for protocols,[21] and for establishing appropriate protocol-directed therapy. Another problem solver, RÉSUMÉ, constitutes an implementation of another generic problem solving method, *knowledge-based temporal abstraction*,[6, 22] which takes as input primary time-stamped patient data (e.g., hemoglobin values), and generates as output relevant time-dependent abstractions of those data (e.g., episodes of anemia). The collection of problem solvers in EON can be viewed as a task-specific architecture for protocol-based care that operates on protocol knowledge bases that developers create using PROTÉGÉ-II.[23]

3 Episodic Skeletal-Plan Refinement

We have implemented two knowledge-based problem solvers that provide protocol-based decision support: (1) a program that determines a patient's eligibility for various protocols and guidelines,[21] and (2) a program that generates therapy recommendations for each clinic visit of a given patient, based on the current clinical situation and appropriate protocols according to which the patient is being treated.[9] In this section, we describe how the latter program—the therapy planner—is implemented within the EON

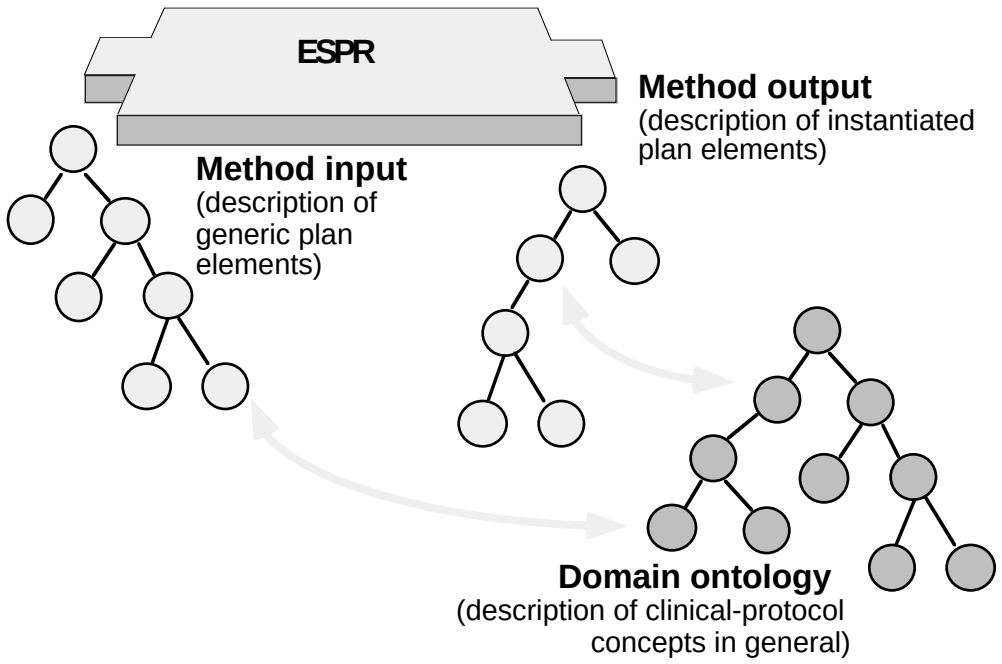

Figure 4: In PROTÉGÉ-II, the data requirements of problem-solving methods are satisfied by the domain-specific knowledge that instantiates the domain ontology (see Figure 2) for a particular application. The approach uses declarative *mapping relations* to specify how instances of concepts represented in the domain ontology relate to specific input–output requirements of the method.

framework. Although the program that determines patients' eligibility for protocols is implemented in a manner similar to that of the therapy planner (i.e., as an assembly of problem-solving methods), we discuss only the therapy planner in this paper, due to space limitations.

At the core of the therapy planner is the problem-solving method known as episodic skeletal-plan refinement (ESPR; Figure 5). The ESPR method was inspired by the behavior of the ONCOCIN system for protocol-based care in oncology,[2] and has been the subject of ongoing research in our laboratory for more than a decade. Unlike a goal-oriented classical planner, the ESPR method assumes that there already exists a *skeletal plan* that needs its details to be fully specified at a particular time. The skeletal plan is hierarchical, consisting of plan elements that specify the actions to be performed at different levels of abstraction (e.g., protocols comprise drug regimens; drug regimens comprise prescriptions; prescriptions comprise the administration of particular drugs). The plan is *skeletal* in nature because the attributes of the planning entities (e.g., the precise doses of drugs; the particular laboratory tests to order) may not be fully specified initially, but rather may depend on the current clinical situation and on additional domain knowledge. Instantiating a skeletal plan requires decomposing it into its current constituent parts, and determining their attributes by using refinement knowledge, so that the problem solver can generate set of fully specified actions that the user can carry out. The precise decomposition of the plan at a particular time (e.g., the specific drug regimens, and thus the specific prescriptions to administer on a given day

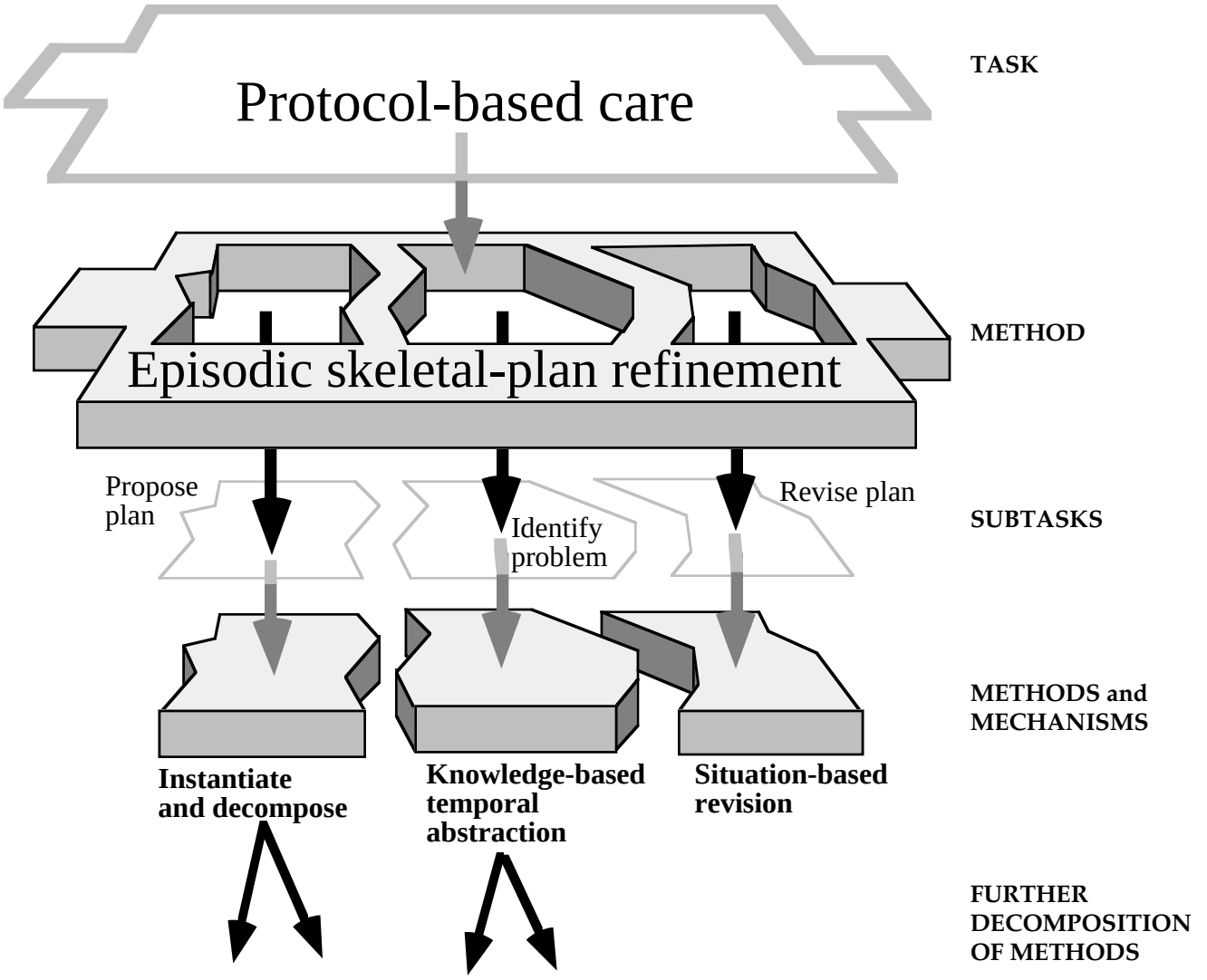

Figure 5: The method of episodic skeletal-plan refinement (ESPR) solves the task of generating a protocol-based treatment plan by calling three subtasks, each of which is addressed by appropriate problem-solving methods.

to meet the requirements of the relevant protocol) needs to be resolved by the planner at run time. Planning is *episodic* in the sense that the planner is invoked multiple times during the execution of the skeletal plan (typically once during each patient encounter). At each planning session, the planner examines the state of the world—including actions that have been carried out and actions that still are in progress—and determines the appropriate actions to take at the time. When the ESPR method is used, determination of appropriate therapy is therefore construed as a process of refining an abstract, skeletal plan (i.e., a clinical protocol) that can be decomposed into one or more constituent plans (e.g., administration of particular treatment regimens) that are each more detailed than is the abstract plan. The output of the planning process is a fully specified (instantiated) execution plan—which, in the domain of protocol-based care, represents a set of treatment recommendations for the practitioner to follow.[9]

Several well-known medical knowledge-based systems have used skeletal-plan refinement as the underlying problem-solving method. For example, Friedland[24] developed the initial description of skeletal planning, which formed the basis of his version of the MOLGEN system for designing laboratory

experiments in molecular genetics. (Friedland's MOLGEN started with an abstract experimental design, which it then fleshed out in a top–down manner to elaborate a detailed experimental research plan.) The Digitalis Therapy Advisor[25] also used skeletal-plan refinement to instantiate each recommendation that the program would make. Unlike EON, these previous systems performed skeletal planning as a one-time event. There was no explicit long-term plan (i.e., protocol) that either MOLGEN or the Digitalis Therapy Advisor attempted to follow in an episodic manner.

The operation of the ESPR method requires as input (1) a skeletal plan to be instantiated, (2) knowledge necessary for refining a detailed plan from the skeletal plan, (3) data from the environment that define the current case, and (4) the current time. The skeletal plan may entail algorithms that take place over time (e.g., a clinical protocol that specifies a sequence of interventions), making the appropriate instantiation of the skeletal plan time dependent. As will become evident, both the need to represent the situation for a given case over time and the time-dependent nature of the skeletal-planning process itself make temporal reasoning an important element of the ESPR method.

Instead of formulating the ESPR method as one monolithic problem solver, we have chosen to decompose the method into three main subtasks: (1) *propose plan*, (2) *identify problem*, and (3) *revise plan* (see Figure 5). The **propose-plan** subtask involves determining the standard plan given the results of the previous planning episodes and the current point in time Thus, this subtask might determine that the patient should be treated according to a particular AIDS protocol; and, given that, that the patient should be treated according to a particular phase of the protocol; and, given that, that the patient should be treated with a particular set of medication prescriptions, and so on. The **identify-problem** subtask identifies characteristics of the current case that might require the problem solver to modify the standard plan (e.g., episodes of anemia or other side effects, progression of disease despite therapy). The **revise-plan** subtask alters the standard plan in response to any problems that have been identified (e.g., by eliminating toxic drugs or reducing their doses, or transferring the patient to an alternative protocol). These three subtasks parallel the parts of clinical protocols that (1) define the standard treatment plan for typical patients, (2) identify possible side effects of the treatment or other predictable clinical situations that may require modification of the standard procedure, and (3) dictate ways of modifying the standard treatment when aberrant situations occur. The subtasks provide a convenient means to decompose the ESPR method into simpler building blocks; they certainly are not the only means by which we might have chosen to break up the ESPR method.

When applied to the application task of designing therapy for patients being treated according to clinical protocols, the ESPR method deduces a treatment plan following a strategy analogous to the propose–critique–modify approach that has been applied to a number of design tasks.[26] First, the method examines the basic protocol algorithm to determine the set of clinical interventions that normally should be administered to a patient, given the patient's history of previous treatment. This "standard plan" may be appropriate for an uncomplicated patient, but frequently there are special situations that mandate plan modification (e.g., a reduction in the usual dose of AZT in an AIDS patient who develops anemia). Thus, the *identify-problem* subtask determines whether there are any predefined extenuating patterns in the data that are to be avoided. If there are, then the *revise-plan* subtask makes an appropriate adjustment to the basic treatment plan that initially had been proposed.

The three computational subtasks of the ESPR method are solved by additional domain-independent problem-solving methods from the PROTÉGÉ-II library, as indicated in Figure 5. We have developed an *instantiate-and-decompose* method to flesh out the skeletal plan and to generate the standard plan, and a *situation-based–revision* method to map problem patterns to a set of plan-modification operators. The *knowledge-based temporal-abstraction* method, which is described in Section 4, solves the *identify-problem* subtask. The ESPR method, and the three methods that are in turn called by the ESPR method, work together to provide the computational machinery required to automate the task of protocol-based therapy planning.

3.1 Instantiate and Decompose Method

In using the generic ESPR method specifically for the task of planning protocol-based therapy, we construe the instantiate-and-decompose method to determine the appropriate treatment and associated diagnostic testing (e.g., standard drugs to administer, laboratory test to order) required by the relevant protocol. The instantiate-and-decompose method operates by determining the values of attributes—such as the *frequency* and *dose* of a drug—of those elements of the skeletal plan that are relevant at that point in time. The method uses protocol-independent refinement knowledge—such as the formula to estimate a patient's body surface area (BSA) from her height and weight—and protocol-specific knowledge—such as the standard dose of a drug, given in mg/m^2 of BSA—to determine an attribute value, such as the dose of the drug for a particular patient. The decomposition part of instantiate-and-decompose method requires asone of its inputs a domain-specific algorithm for decomposing each plan element (e.g., protocol specification) into that planning element's constituent parts (e.g., drug regimens). Such an algorithm for a clinical-trial protocol, for example, may specify that a patient should receive alternative courses of two experimental drugs for a period of time, and then receive post-therapy evaluation for an additional interval (Figure 6). The algorithm is represented internally as a state-transition graph whose nodes are elements of the skeletal plan (e.g., drug regimens, prescriptions), and whose transitions specify the conditions for moving to the next plan element (e.g., changing from one drug regimen to another). These transition conditions are written as predicates that check for particular temporal patterns in the case data (e.g., "whether there has been moderate anemia lasting 2 weeks since AZT therapy was last suspended"). These state-transition tables have a format similar to that used in other approaches for automating protocol-based care such as GEODE-CM,[27] but differ in that, in those approaches, making a transition to a new state usually reflects a change in the system's knowledge about the patient, rather than the beginning of a new protocol-specified action.

3.2 Knowledge-Based Temporal-Abstraction Method

A traditional planning algorithm selects among alternative plan actions by forecasting the consequences of carrying out those actions, and by evaluating which alternative is most likely to lead to satisfaction of the planner's goal. The ESPR method, however, uses predefined temporal patterns in the case data to determine the current actions to take, given a skeletal plan. The ESPR planner thus "fine tunes" the standard, skeletal plan without the explicit need to predict the affects of competing actions This reliance on predefined patterns to select appropriate actions requires the ESPR method to have some means to identify external events referenced in the temporal patterns (e.g., changes in the patient's condition) that may dictate modification of the standard plan (e.g., alteration of the dose of a drug). ESPR uses the knowledge-based temporal-abstraction method for this problem-identification task. The knowledge-based temporal-abstraction method takes as input time-stamped and interval-based data (e.g., hemoglobin values) and different classes of knowledge required to perform temporal abstraction (e.g., knowledge about the definition of anemia in terms of hemoglobin values), and generates as output episodes of patient problems that may cause the ESPR therapy planner to adjust the standard treatment plan (see Section 4).

3.3 Situation-Based Revision Method

ESPR uses the situation-based-revision problem-solving method to modify the standard plan generated by the instantiate-and-decompose method. The situation-based-revision method performs these modifications whenever the knowledge-based temporal-abstraction method identifies certain prespecified temporal patterns in the patient data. The revisions to the current plan are defined with operators that (1) add new plan actions, (2) modify attributes of proposed plan actions, or (3) withdraw proposed plan actions. Thus, a temporal pattern that calls for a reduction in the dose of a drug may require the situation-based-revision method to query for the "second episode of moderate anemia," and then to modify the DRUG_DOSE

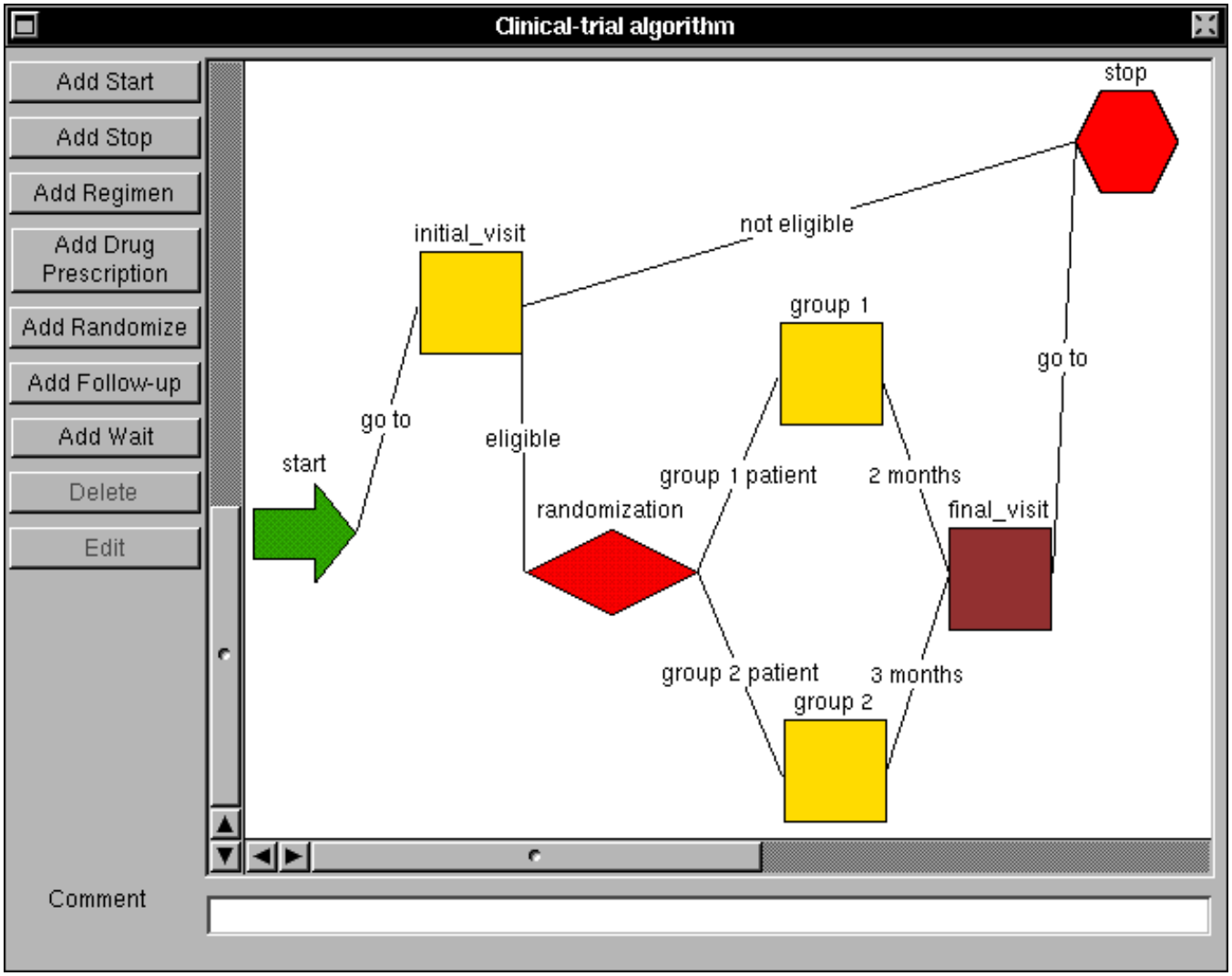

Figure 6: This algorithm for an AIDS clinical-trial protocol shows a portion of the sequence of actions that need to be applied to patients over time. The algorithm is being entered into a knowledge-acquisition tool generated by the PROTÉGÉ-II system from the domain ontology shown in Figure 2. The protocol in the figure calls for patients who are HIV positive and are found to be eligible for the protocol after an initial visit to be assigned at random to one of two groups that receive alternative therapies.

attribute of the proposed drug prescription. When the ESPR *identify-problem* subtask executes, the knowledge-based temporal-abstraction method detects all occurrences of moderate anemia. Identification of the temporal pattern itself— a task that involves ordinal selection (i.e., identification of the *second* episode of moderate anemia)— requires the situation-based-revision method called by ESPR to query the Chronus database system (see Section 5).

3.4 Application of ESPR to Protocol-Based Care

The knowledge required by the ESPR method (a hierarchy of skeletal plan elements; algorithms that specify how each plan element should be decomposed over time; and temporal patterns that invoke plan modification operators) is not specifically clinical in nature. To apply ESPR to the application task of planning protocol-based therapy, the developer needs to define a domain ontology for protocol-based care that can be mapped to the terms used in the ESPR method.* At a minimum, the domain ontology must have (1) a model of clinical protocols, including a description of how clinical algorithms are represented; (2) a schema for representing patient data; (3) a categorization of medications, laboratory tests, and clinical interventions that may be used in the protocols; (4) a patient description vocabulary, including the terms that denote patient problems; and (5) a predicate language for specifying clinical conditions. The mappings from the domain ontology to the data requirements of the ESPR problem-solving method determine how the ESPR method will apply its algorithm to the domain knowledge.[9, 20] The mappings, for example, specify how domain-specific flowcharts that describe clinical algorithms (such as in Figure 6) can be viewed by the ESPR method as generic state-transition tables that have skeletal plan elements associated with each state, and temporal predicates associated with each transition.

The ESPR-based therapy planner is an example of one of the problem-solving components that we have incorporated in the EON architecture; it is responsible for refining protocol specifications into specific treatment recommendations for providers to follow. The ESPR method itself, however, is designed to be domain independent. The problem-solving method is a generic top–down skeletal planner that satisfies the computational requirements of protocol-based therapy planning. Similarly, the three problem-solving methods that automate the subtasks invoked by ESPR also are completely domain independent, and can be used as modular building blocks. There may be times when developers will want to substitute a modified version of one of these finer-grained problem-solving methods into ESPR, to meet unusual problem-solving requirements posed by a particular class of application tasks. The compositional nature of the problem-solving methods in the PROTÉGÉ-II library facilitates this kind of "plug and play" adaptation.[28]

4 Knowledge-Based Temporal Abstraction

Our analysis of a large number of clinical protocols demonstrates that protocol authors often specify complex patterns of clinical events that dictate particular clinical actions that providers should follow.[10] Although human clinicians are good at recognizing complicated patterns in patient data, it is much more difficult for machines to perform the necessary temporal reasoning.[29] Automation of protocol-based therapy planning requires that the computer access time-stamped data stored in a clinical database. Whereas the database contains raw data values, the clinical conditions that may determine a patient's treatment typically are described at a higher level of abstraction, often involving *intervals* of time, in addition to time points. Thus, the protocol for AIDS therapy that we have used in our previous examples calls for changes in zidovudine dosing based on *episodes of anemia*, rather than on the value of a patient's serum hemoglobin on a particular day. The task of abstracting data into higher-level, interval-based concepts is called *temporal abstraction.* Figure 7 presents examples of abstractions of platelet and granulocyte values during administration of a clinical protocol known as PAZ for treating patients who have chronic graft-versus-host disease (CGVHD). The time line in the figure starts with a bone-marrow transplantation (BMT) event. Primitive data elements (e.g., platelet- and granulocyte-count

---

*In PROTÉGÉ-II, such a domain ontology is called an *application ontology*.[20] Developers often create an application ontology by refining a domain ontology that they have created previously, so that the elements of the application ontology can be mapped easily to the problem-solving method that the developers have chosen for the application.

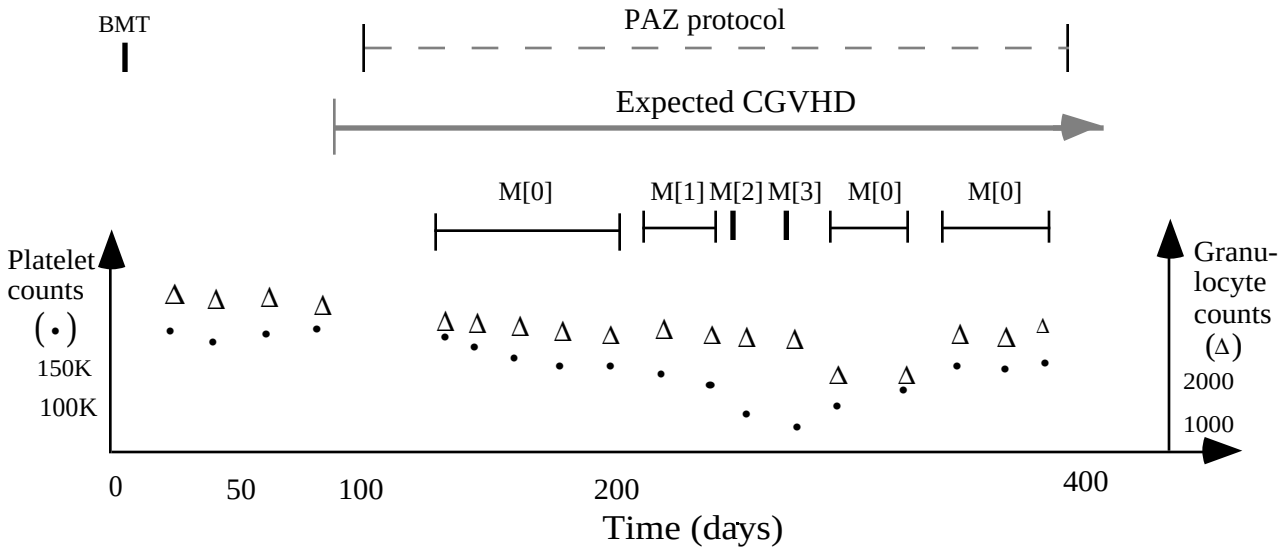

Figure 7: Typical inputs to and outputs of the temporal-abstraction task viewed on a time line.

values) are then abstracted in a context-specific manner into more meaningful summarizations (e.g., grades of bone-marrow toxicity). When appropriate, point-based data or abstractions are joined to create new intervals during which some characterization holds. Further temporal abstraction may allow intervals to bejoined into longer intervals that provide even higher-level conceptual summaries of the patient's course over time.

In the ESPR problem-solving method, the subtask named *identify problem* (see Figure 5) requires temporal abstraction to translate elementary data into meaningful patterns. The *identify-problem* subtask can be solved by the problem-solving method known as *knowledge-based temporal abstraction.*[6,30] This method, like the other methods in the PROTÉGÉ-II library, is domain independent. As used by the EON system, the knowledge-based temporal-abstraction method evaluates and summarizes the state of a patient over a designated time interval (possibly, over the entire record of the patient). The method takes as its input time-stamped data (e.g., individual hemoglobin values and therapy events), and returns as its output abstractions of those data that are interpreted over specific time points or intervals (e.g., periods of anemia or of normal hemoglobin levels).

Previously,[22] we have identified five computational requirements for a method that abstracts time-oriented clinical data:

1. The method must allow input data and output conclusions that are at multiple levels of abstraction (e.g., raw hemoglobin might be abstracted into "moderate anemia"; given additional information, "moderate anemia" might then be abstracted into "pancytopenia").

2. The method must accept and interpret data that arrive in any temporal order—allowing for retraction of abstractions that are no longer valid, and for creation of new interpretations of the past and present state, such as when laboratory reports valid for a previous day suddenly arrive.

3. The method's interpretation of time-oriented data should be sensitive to the context in which the data were obtained. For example, certain blood-glucose values in a patient being treated according to a protocol for diabetes may be considered "normal" if they are measured following meals, but might be "abnormal" if they were recorded when the patient was fasting.

4. The method should allow for maintenance of multiple, concurrent interpretations of the same data. Thus, the problem solver that invokes the method can be informed that a pattern in a patient's serum hemoglobin measurements is consistent *both* with anemia due to progression of disease and with myelotoxicity due to current therapy.

5. The domain knowledge on which the problem-solving method operates must have clear semantics, thus facilitating the acquisition, maintenance, reusability, and sharability of that knowledge.

These requirements are the goals of the knowledge-based temporal-abstraction method and of the implementation of that method in the problem-solving component known as RÉSUMÉ.[22]

4.1 Details of the Temporal-Abstraction Method

The knowledge-based temporal-abstraction method, like other methods in the PROTÉGÉ-II library, is a reusable building block. The method itself identifies a number of subtasks, each of which is solved by other reusable methods or mechanisms in the library.

The knowledge-based temporal-abstraction method (Figure 8) entails five subtasks: (1) temporal context restriction —creation of relevant interpretation contexts crucial for focusing and limiting the scope of inference (e.g., identifying "postprandial" and "fasting" contexts for interpretation of blood-sugar values); (2) vertical temporal inference—inference from contemporaneous propositions regarding raw or abstract data into higher-level concepts (e.g., abstracting a hemoglobin value of 8.0 g/dl into a concept such as "anemia" occurring at the same time point); (3) horizontal temporal inference—inference from similar-type propositions associated with time intervals that cover different, but meeting or overlapping, time periods (e.g., concluding that two episodes of anemia that have consecutive dates can be viewed as a single episode); (4) temporal interpolation—union of nonmeeting points or intervals, associated with propositions of similar type (e.g., concluding that two episodes of anemia that occurred within a short time of each other, given no knowledge of the state of the patient during the intervening dates, can be viewed as a single episode); and (5) temporal pattern matching—creation of intervals by matching of patterns over disjoint intervals, associated with propositions of various types (e.g., using patterns of anemia and leukopenia following bone-marrow transplantation to infer the presence of graft-versus-host disease). These five subtasks of the knowledge-based temporal-abstraction method have been described in detail elsewhere.[6, 30]

Each of the five subtasks is solved by a different temporal-abstraction problem-solving *mechanism*, also stored in the PROTÉGÉ-II library (see Figure 8). The mechanisms, which we have described previously,[6, 22] include one mechanism for creating relevant temporal contexts, three basic temporal-abstraction mechanisms, and one mechanism for matching temporal patterns. The input to the temporal-abstraction mechanisms is a set of point- or interval-based clinical *parameters* (e.g., a particular hemoglobin value; 3 weeks of moderate anemia) and *events* (e.g., administration of regular insulin; therapy according to a particular chemotherapy protocol). Another, special type of input is one or more abstraction goals (e.g., monitoring of insulin-dependent diabetes). The temporal-abstraction mechanisms produce output abstractions of several abstraction types: *state* (e.g., LOW), *gradient* (e.g., INCREASING), *rate* (e.g., FAST), and *pattern* (e.g., CRESCENDO).

The **context-forming mechanism** solves the subtask of temporal-context restriction. The mechanism creates temporal frames of reference that enable the temporal-abstraction mechanisms to create context-specific abstractions. The presence in the case database of particular events (e.g., therapeutic interventions), parameters (i.e., primary data or previously concluded temporal abstractions), or

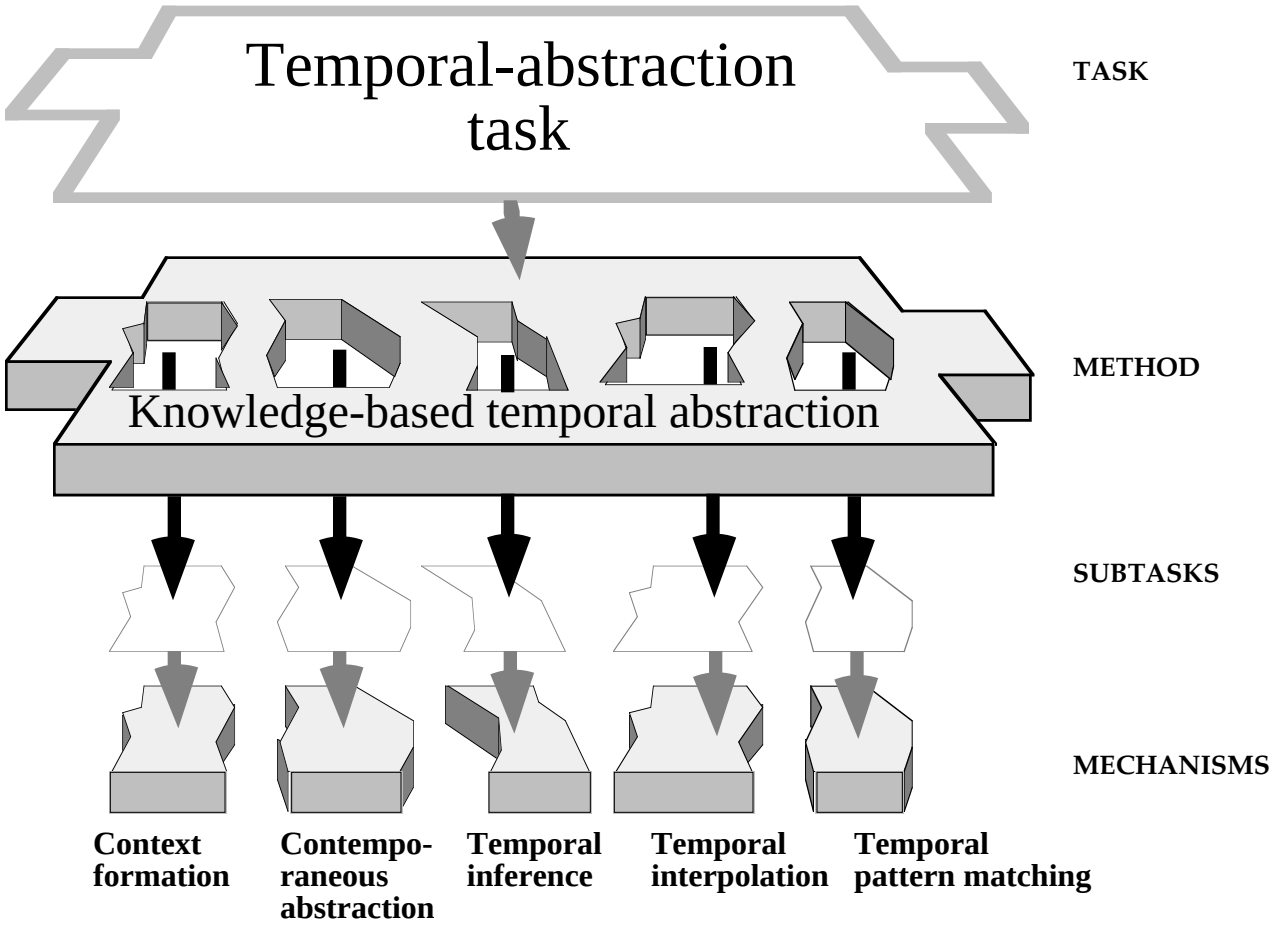

Figure 8: The knowledge-based temporal-abstraction method decomposes the temporal-abstraction *task* into five *subtasks.* Each subtask can be solved by one of five temporal-abstraction *mechanisms.*

abstraction goals may trigger dynamically the generation of a corresponding interpretation-context interval. Thus, an event such as "bone-marrow transplantation" may trigger an interpretation context in which subsequent patient data are considered as possibly defining the presence of graft-versus-host disease (see Figure 7); the presence of one parameter (e.g., "moderate anemia") may invoke a context during which the interpretation of another parameter (e.g., erythrocyte sedimentation rate) is affected.

The **contemporaneous-abstraction mechanism** transforms one or more clinical parameters and their values, attached to contemporaneous time points or time intervals, into the value of a new, abstract parameter (see Figure 8). The mechanism therefore produces output parameters that are associated with precisely the same time points or intervals as are the input parameters. Contemporaneous abstraction, for example, might conclude that a hematocrit of 30% at a particular time point corresponds to "moderate anemia."

The **temporal-inference mechanism** operates on temporal intervals that occur at *different* times, but that meet or overlap. This mechanism deduces specific types of interval-based logical conclusions (e.g., that two meeting intervals that represent some clinical condition actually are equivalent to one longer interval of that condition), given interval-based propositions, using a deductive extension of Shoham's

temporal semantic properties.[31] These semantic properties are defined for every parameter on which the mechanism operates. They enable the temporal-inference mechanism appropriately to join two consecutive periods of anemia that each last 9 months into a single 18-month period of anemia. These semantic properties also enable the mechanism to avoid summarizing two successive episodes of pregnancy, each lasting 9 months, as an 18-month period of pregnancy—since intervals of pregnancy are not concatenable. Similarly, "a weeklong episode of coma" implies that coma occurred during each day of the week; this situation is not necessarily true for an interval of "labile hypertension," during which the blood pressure may be normal throughout one or more subintervals. The temporal-inference mechanism also can conclude the value of a new abstraction that the mechanism creates by joining two interval abstractions that either meet or overlap. For example, given an interval during which the value of a particular parameter is DECREASING, and a second interval during which the value of the parameter is CONSTANT, the mechanism might concatenate the two intervals into one interval and conclude that the value of the parameter throughout the longer interval is NONINCREASING. (The precise behavior is dependent on the domain knowledge that serves as input to the mechanism.)

The **temporal-interpolation mechanism** *bridges gaps* in the data between time points or time intervals, using domain-specific knowledge about the parameters involved. Thus, knowledge about the expected temporal variance in a patient's body weight would allow the method to conclude that, if there are two measurements of weight 1 month apart such that each measurement is ELEVATED, then the patient's weight probably can be summarized as ELEVATED throughout that 1-month interval; conversely, two measurements of ELEVATED heart rate that are 1 month apart do not imply that the heart rate is necessarily increased throughout the interval delimited by the two time points. The temporal-interpolation mechanism uses probabilistic functions that developers can define for each clinical parameter to model belief in the value of the corresponding parameter when there are no specific data in the medical record from which to infer the value.[30] The temporal-interpolation mechanism thus provides a principled means for EON to deal with missing data in the clinical record.

The **temporal-pattern–matching** mechanism matches predefined complex temporal patterns with the abstractions created by the other temporal-abstraction mechanisms. The output of the temporal-pattern–matching mechanism is a complex parameter, such as "rebound hyperglycemia" or "crescendo angina." The temporal pattern-matching task typically requires consideration of multiple intervals of different types distributed in time.

To use the five temporal-abstraction mechanisms in a particular clinical domain, the developer must supply the mechanisms with the necessary domain-specific knowledge. This domain-specific knowledge is represented in an explicit, declarative fashion. Because the knowledge-based temporal-abstraction method is domain-independent, if the method is to be used in a new clinical situation, the developer must supply the requisite domain knowledge concerning those parameters and events that are relevant in the new area of medicine. The program code that implements the method (i.e., the RÉSUMÉ system) does not have to be modified at all.[6, 32]

4.2 The Relationship Between RÉSUMÉ and ESPR

The knowledge-based temporal-abstraction method is implemented within EON as the RÉSUMÉ system. Recall that the therapy planner in EON uses the ESPR problem-solving method to generate specific treatment recommendations from the skeletal plans of clinical guidelines and protocols. Each time that ESPR performs the task of protocol-based therapy planning, it first calls on the *propose-plan* subtask to establish a putative treatment plan for the current patient visit based on the standard protocol. This treatment plan consists of drugs to administer, laboratory tests to order, referrals to make, and so on. The ESPR method then turns to the *identify-problem* subtask, and invokes the knowledge-based temporal-abstraction method in RÉSUMÉ to solve that subtask. The knowledge-based temporal-abstraction method activates the temporal-abstraction mechanisms, which create a set of generalizations that provide a detailed

model of the patient's condition over time. This model includes abstractions that correspond to current and past side effects of therapy, disease status, and anticipated patient problems. Once the relevant temporal abstractions have been generated, they activate the ESPR *situation-based–revision* method, which has been configured for the *revise-plan* subtask; the *situation-based–revision* method takes as input protocol-specific knowledge of various contingencies that might affect the ultimate treatment recommendation, and generates as output modifications to the standard plan that adjust for any patient-related problems that RÉSUMÉ might have detected.

5 Temporal Query System

Automation of protocol-based care requires generation of therapy recommendations based on patient data that are stored in electronic medical-record systems. In processing such data, the decision-support system needs to perform data queries to identify clinically relevant temporal patterns of patient conditions. For example, using both the primitive data stored in the electronic medical record and temporal abstractions that are created by the RÉSUMÉ system, a protocol-based decision-support system may need to verify time-oriented conditions such as

- Whether the patient has had anemia of at least moderate severity *during* the period that the protocol has been active
- Whether that patient had a *second* episode of anemia of at least moderate severity
- Whether that patient had an episode of moderate or severe anemia that *persisted* for more than 2 weeks

In general, these temporal conditions are of one of three types: (1) temporal context, (2) temporal ordering, and (3) temporal duration.[10] In this section, we discuss how the EON architecture translates these three types of temporal conditions into corresponding queries to a time-oriented clinical database.

In developing a temporal query system that can meet the temporal data-management requirements of protocol-based decision support, we have chosen to implement a temporal extension to the relational data model. Relational databases, with their standard Structured Query Language (SQL), have emerged as the prevailing technology for storing and accessing clinical data. Because commercial relational databases represent a mature technology, using the relational model as the basis for a temporal query system is more advantageous than is developing a novel database method from scratch. Use of the relational model as the foundation for our work readily allows us to create portable applications for use on different hardware platforms. Relational database technology also facilitates construction of flexible client–server architectures for decision support. The disadvantage of this approach, however, is that developers first must overcome certain limitations that relational databases have when dealing with temporal queries.

5.1 Addressing Limitations of the Relational Model

Our work to develop extensions to the relational model and to SQL for support of temporal data processing has paralleled the approach undertaken by other investigators.[33, 34] We have defined a format for relational data schemata that can support the necessary temporal distinctions, and have created a *temporal algebra* that can support temporal operations on data represented in such a schema format. In our approach, we have chosen to model all temporal data in the relational database using a single temporal representation format, which we call a *historical relation.* The historical relation associates each time-dependent entry with a *start time* and an *end time* (Figure 9). For interval-based data (e.g., a

Start Time	End Time	Patient	Drug	Dose
Jan 3, 1995	March 6, 1995	1111	Zidovudine	500
March 7, 1995	April 5, 1995	1111	Zidovudine	300
Sept 4, 1994	May 14, 1995	2222	Zidovudine	500
May 14, 1995	now	2222	ddI	400
May 14, 1995	now	2222	Septra DS tablet	1

Figure 9: Chronus operates on relational tables in which each tuple has been augmented with a start time and an end time.

period of anemia or an interval during which a patient was given a particular medication), the start-time attribute captures the starting point of the interval, whereas the end-time attribute stores the stopping point. Figure 9 illustrates an example of a historical relation storing the medication record for two patients. For instant-based data (such as a laboratory-test result), the start-time and end-time attributes are always set to the same value. Events that are still ongoing have an end time of *now*.

Given this parsimonious relational representation for both instant- and interval-based data, we have the ability to define a query method that can manipulate all temporal relations in a consistent manner. Standard relational query methods, however, are not sufficient to allow the ESPR therapy planner to query a relational database for verification of the complex temporal conditions that drive decision support for protocol-based care. For three important reasons, standard relational databases and their conventional SQL interfaces are not well suited for manipulating such historical relations.

First, SQL does not contain syntactic constructs for making temporal comparisons of time stamps. Although the search conditions in SQL do allow simple comparison operators (e.g., >, <, and =) on time points, the language does not permit a user to express directly conditions that involve interval-based comparisons (e.g., DURING or OVERLAPS). The lack of abstract temporal predicates in SQL makes it extremely awkward to express queries regarding many relatively simple temporal relationships in the patient data, particularly when there is a need to represent relationships that involve time-interval interdependencies among the individual tuples (e.g., a query about whether patient data in two different relations have certain values during overlapping time intervals).

Second, the operators in standard relational algebra, which define the semantics of SQL, are not *closed* for data stored as historical relations. For example, if the ESPR-based therapy planner were allowed to apply the standard relational PROJECT operator to remove the time stamps associated with data such as those in Figure 9, the resultant relation would have no temporal dimension, and thus would not be a valid historical relation. If the therapy planner were to use the JOIN operator to combine temporal data from two relations, the result would be a relation that would have *four* time stamps, rather than two. The therapy planner then would not be able to distinguish which set of time stamps belonged to which attributes. Since the relational model treats all attributes of each relation equally, the model does not provide a special status to the interval stamps associated with each historical relation. Consequently, SQL is not guaranteed to maintain the time stamps in the result of each query. The query language that we have developed for use in EON, however, does recognize the unique role played by the time-stamping information in each tuple.

The third problem is that the set of operators in the standard relational algebra is not *complete* for the types of temporal conditions that are important in protocol-based care. SQL does not have the ability to support nonrelational temporal operations—such as the coalescing of the time periods of two consecutive periods of medication dosages, or the selection of the second episode of anemia from the relation that stores patient problems. We have shown previously that concatenation and ordinal selection are necessary operators for verification of protocol-based conditions.[10] Thus, standard SQL is not sufficient to meet the temporal querying requirements of protocol-based decision-support systems.

5.2 The Chronus System

Our temporal query system, Chronus,[11] addresses the three problems with relational databases and SQL detailed in Section 5.1. Chronus implements a temporal relational algebra that confers special status on the time stamps associated with tuples such as those in Figure 9. This algebra ensures that any operation on a historical relation returns a new relation in which each tuple is associated with exactly one start time and exactly one end time. This temporal relational algebra is an alternative to the standard relational algebra assumed by SQL. The Chronus system retains the SELECTION operator from the standard relational algebra, but substitutes a TEMPORAL PROJECTION operator for the usual PROJECTION operator, to disallow the removal of time stamps from any historical relation. CARTESIAN PRODUCT and standard types of relational JOINs are not allowed in the temporal relational algebra; instead, the algebra provides three new TEMPORAL JOINs that permit the contents of two relational tables to be combined in different ways, depending on temporal relationships between the tuples of the two tables.[11] These TEMPORAL JOINs ensure that, whenever the tuples in two relations are combined, the resulting relations will contain only tuples that have appropriate time stamps.

The Chronus algebra also introduces an additional operator, CATENATION, that has no correlate in standard relational algebra. This new operator merges the data in two temporally adjacent tuples of a relation when (1) the nontemporal elements of the tuples are identical, and (2) the temporal intervals associated with the tuples meet or overlap. Thus, if, after a series of data manipulations, a relation is created in which there are two tuples indicating that a particular drug was administered to a patient at a specific dose, and the time stamps of the two tuples denote *contiguous* periods of drug administration, then the CATENATION operator can replace the two tuples with a single tuple, with new time stamps that denote the complete period during which the drug was administered. The decision regarding whether the semantic properties of these data permit concatenation is determined from the same temporal-abstraction ontology that informs the RÉSUMÉ system (see Section 4.1).

As in traditional relational database systems, users do not query data in the database using the algebra directly; rather, they interact with a more abstract query language that translates their requests into a sequence of appropriate algebraic operations. Chronus provides a query interface that supports a language that we refer to as TimeLine SQL (TLSQL). TLSQL has the syntactic structure of SQL, but incorporates a new WHEN clause that allows the user to select data based either on temporal conditions, such as interval-based comparisons, or on temporal joins, such as an intersection of the time periods in two historical relations.[11] TLSQL also has the ability to apply, to the results of a query, special temporal aggregation operators (e.g., FIRST, SECOND, and LAST), as well as the standard SQL aggregation operators (e.g., MIN and SUM).

5.3 Integration of Chronus within EON

In the EON architecture, an application such as the ESPR-based therapy planner uses a programmatic interface to formulate temporal queries in TLSQL to the Chronus system. Chronus parses those queries, uses its algebra to construct a set of corresponding—and more complex—relational queries in standard

SQL, and then queries a commercial relational database (currently Sybase) where both the primitive and abstracted clinical data are stored. In a postprocessing step, Chronus performs the temporal operations that are not supported in standard SQL (e.g., concatenation and ordinal selection), and passes the results of the query to the ESPR method. Because Chronus is designed to interact with commercial relational databases, the underlying database system can optimize the standard relational queries that Chronus passes to it. The commercial database system also can perform the usual database-administration functions.

EON thus provides an architecture for integrating various components that automate the task of planning protocol-based care. A problem solver based on the ESPR problem-solving method identifies the standard plan that should be applied to the patient, calls the RÉSUMÉ system to identify any existing patient problems that might dictate modification of the standard treatment plan, and then revises the standard plan in response to the identified problems. The Chronus database system stores not only time-stamped patient data, but also abstractions of those data detected by the RÉSUMÉ system, as well as a history of past therapy events. Each of the three components addresses specific computational requirements for automation of protocol-based therapy: ESPR performs top–down planning, RÉSUMÉ performs temporal abstraction, and Chronus stores and retrieves temporal data. Together, the three components provide the means to reason about complex clinical protocols, and to apply those protocols to individual patients. Because all the knowledge of the protocols themselves is stored external to these three components (see Figure 1), ESPR, RÉSUMÉ, and Chronus are reusable for new application tasks that may be outside the domain of protocol-based care—or even outside the domain of medicine. More important, because we store the medical knowledge required by these domain-independent components in a separate knowledge base that has a well-defined structure, we can apply the PROTÉGÉ-II knowledge-acquisition methodology to the definition and maintenance of all protocol knowledge that EON might need to process.

6 Knowledge Acquisition for EON

PROTÉGÉ-II represents a general knowledge-acquisition methodology that we have used to construct a wide variety of knowledge-based systems.[28, 35, 36, 37] Despite the generality of our approach, we have gained particular experience with PROTÉGÉ-II by building knowledge bases of clinical protocols for use with EON.[9] For example, we have used PROTÉGÉ-II to construct a variety of knowledge bases that define clinical protocols regarding therapy for patients with AIDS and HIV-related illnesses. These knowledge bases are used by the decision-support element of an intelligent computer-based patient-record system known as THERAPY-HELPER (T-HELPER).[5]

T-HELPER facilitates data management in the primary care of patients who have AIDS. The system uses the EON architecture to offer active decision support regarding patients' participation in clinical trials of promising new treatments. These clinical trials involve protocols for the administration of experimental therapies that often are carried out over several months or even longer. The clinical-trial protocols typically require the health-care provider to modify the standard therapy whenever patients experience adverse drug-related toxicities (e.g., drug-induced anemia) or changes in their disease status (e.g., progression of AIDS). For the T-HELPER system to generate a therapy plan for a patient participating in a clinical trial for a particular clinic visit, the actions of the various components of the EON architecture must coordinate in a meaningful manner. Thus, to determine whether the dosage for AZT should be modified when the representative protocol in Figure 6 is followed, the ESPR-based therapy planner uses Chronus to access the time-oriented laboratory values that are stored in an archival relational database, and sends these data to the RÉSUMÉ component, so that the latter can create the temporal abstractions that identify relevant patient conditions. The therapy planner uses Chronus to store the abstractions temporarily in the external database, and then formulates temporal queries to determine whether there are conditions that might predicate modification of the standard plan.

EON generates therapy recommendations for T-HELPER in real time. Although we do not have precise performance data for the T-HELPER system overall, EON produces in a matter of seconds precise

recommendations about the drugs to administer, laboratory tests to order, and necessary follow-up appointments. (A detailed performance analysis of the Chronus component has been reported elsewhere.[11])

We have used PROTÉGÉ-II to develop all the knowledge bases of AIDS clinical-trial protocols on which EON operates within the T-HELPER system. These knowledge bases include not only descriptions of the procedures that health-care workers should follow when caring for patients enrolled in these protocols (see Figure 6), but also the knowledge of clinical parameters, such as hemoglobin and anemia, that are required by RÉSUMÉ to generate temporal abstractions (see Section 4.2). Because PROTÉGÉ-II provides a general knowledge-acquisition methodology, the approach with which PROTÉGÉ-II was used to create knowledge bases for T-HELPER also applies to construction of protocol knowledge bases for other clinical domains in which EON also might operate.

The first step in developing knowledge-based systems using the PROTÉGÉ-II methodology is to construct a domain ontology (see Figure 2). The domain ontology defines the general concepts and the relationships among concepts that we ultimately need to describe the individual elements that constitute the application area. Thus, the T-HELPER domain ontology defines abstract concepts such as *protocol*, *prescription*, and *medication*. Concepts in the ontology are related hierarchically; therefore, we can indicate that *clinical-trial protocol* is a kind of *protocol*, and that *antiretroviral drug* is a kind of *medication*. An ontology can be viewed as a formal representation of the domain of discourse for describing an application area.[16, 17] PROTÉGÉ-II takes a domain ontology, such as the one that we developed for T-HELPER, and generates automatically a knowledge-acquisition tool based on that ontology (Figures 6 and 10); developers enter the content knowledge for particular applications into this knowledge acquisition tool, casting their statements about the application area in terms of the corresponding domain ontology. For example, when using the knowledge-acquisition tool generated from the T-HELPER ontology, developers describe AIDS clinical trials using the language of *protocols* and *medications* defined by the ontology. By extension, the domain ontology also defines the domain of discourse with which T-HELPER's *end users* can specify clinical concepts—since the only patient descriptions that are meaningful to EON are those that appear in the protocol knowledge bases, which always are defined in terms of the concepts in the domain ontology.[38]

The PROTÉGÉ-II approach allows developers to generate automatically a custom-tailored knowledge-acquisition tool that is well suited for entry of descriptions of clinical protocols. By filling in the blanks of computer-generated forms (see Figure 10), and by creating flowcharts that depict the procedural elements of protocols by making selections from domain-specific drawing palettes (see Figure 6), developers can enter extremely complex protocol specifications in a straightforward manner. The knowledge-acquisition tools generated by PROTÉGÉ-II convert the users' entries into an internal knowledge representation (specified in CLIPS). In the case of T-HELPER, the knowledge-acquisition tool not only provides a means to create protocol knowledge bases quickly, but also allows clinicians to browse through descriptions of previously entered protocols and to update the knowledge whenever necessary. These changes are then also automatically reflected in the corresponding decision-support tool.

As discussed in Section 2, with the PROTÉGÉ-II methodology, automation of a particular task requires developers to construct an explicit mapping between the domain-specific knowledge (e.g., protocol descriptions) and the data requirements of the *problem-solving method* that provides the control strategy for the task (e.g., EON's ESPR method). Because both ESPR and the temporal-abstraction mechanisms that constitute the RÉSUMÉ system are "standard" reusable PROTÉGÉ-II problem-solving methods, we can create the necessary mapping.[9, 20] PROTÉGÉ-II provides a tool that makes it possible to indicate how each input–output requirement of a problem-solving method might relate to one or more elements of the domain ontology. Thus, we specify that the skeletal plan elements on which the ESPR method operates correspond to *protocols* in the T-HELPER domain ontology; that certain *parameters* on which RÉSUMÉ

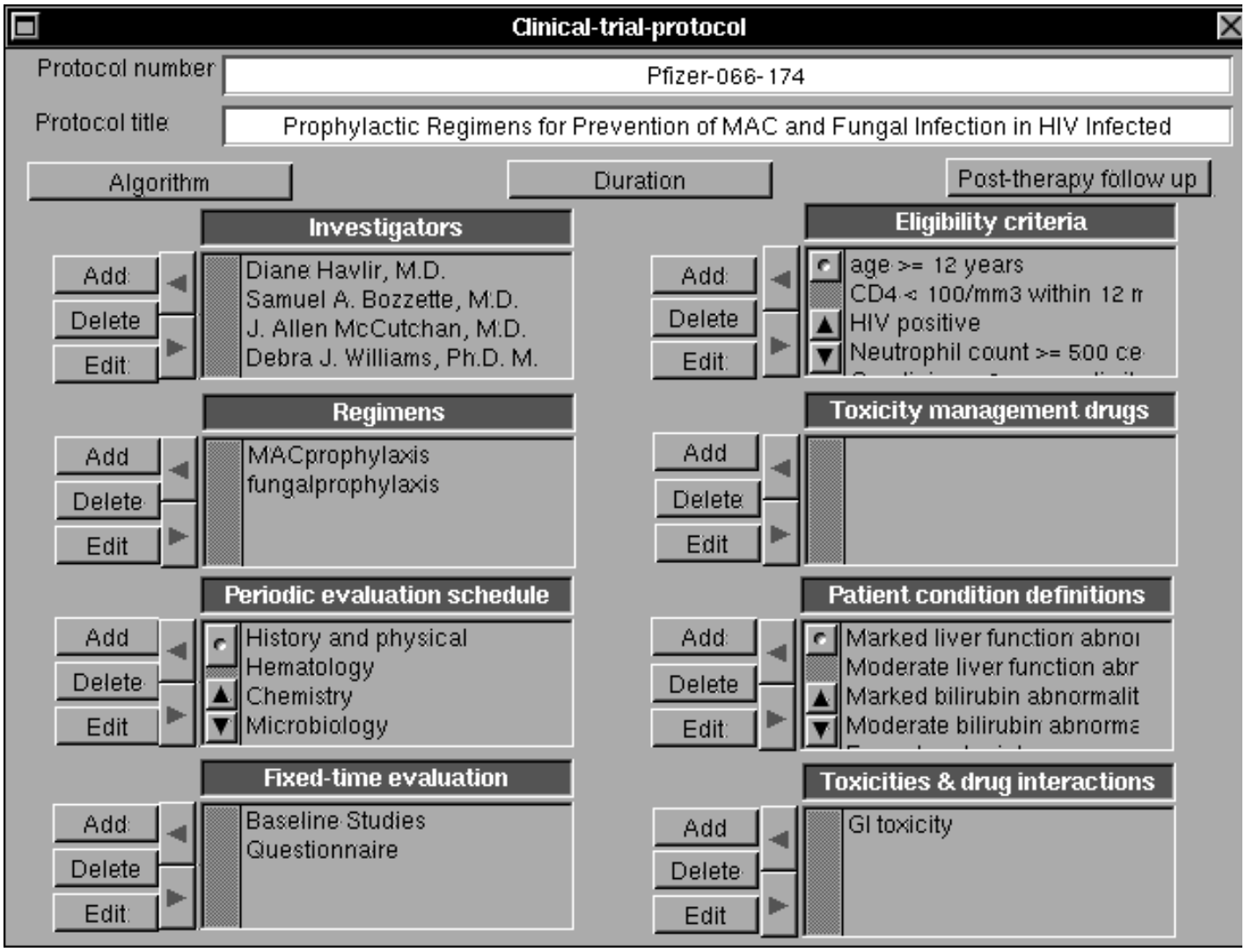

Figure 10: This is a form from the knowledge-acquisition tool generated from the AIDS domain ontology (see Figure 2). Pressing the button labeled *algorithm* causes the graphical environment shown in Figure 6 to appear on the workstation screen.

operates correspond to *laboratory-test results* in the domain ontology; and so on.

The component-based approach that characterizes the PROTÉGÉ-II methodology simplifies adaptation of EON for new clinical application areas. For example, we recently created an experimental version of the T-HELPER system that uses EON to assist protocol-based care in the domain of breast cancer. To construct this prototype system, we modified the T-HELPER AIDS ontology to create a new ontology that captures distinctions relevant to the care of patients who have breast cancer. Figure 11 shows a portion of the breast-cancer ontology as it appears in the PROTÉGÉ-II Ontology Editor. As is the case with the AIDS ontology, the breast-cancer ontology models protocols to include a set of steps and transitions that define a finite-state algorithm. In Figure 11, the leftmost column in the Ontology Editor shows that the

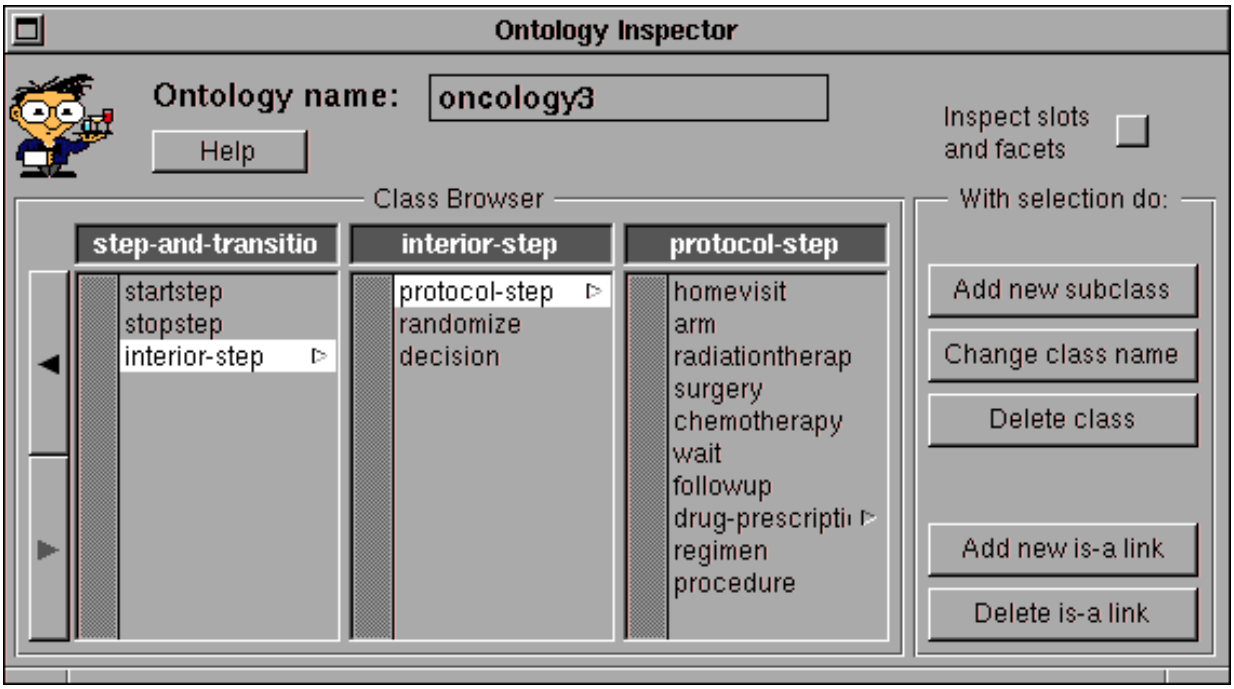

Figure 11: This small portion of the breast-cancer ontology appears in the PROTÉGÉ-II Ontology Editor. Compare the information shown with the AIDS ontology depicted diagrammatically in Figure 2.

STEP-AND-TRANSITION class has subclasses that include STARTSTEP (an identifier for the beginning of a procedure), STOPSTEP (an identifier for the end of a procedure), and INTERIOR-STEP (which models all other states of the algorithm); subclasses of INTERIOR-STEP include PROTOCOL-STEP, which has subclasses that correspond to the modalities of breast-cancer therapy. Among these modalities are new classes of therapeutic interventions—such as *surgery*, *radiotherapy*, and *home visits*—that are not germane to the protocol-based care of patients with HIV-related disease as practiced in the AIDS clinic where T-HELPER is installed. Many of the more abstract concepts in the T-HELPER AIDS ontology could be adapted, however, for use in the new breast-cancer ontology. Development and refinement of the complete breast-cancer ontology required several weeks of work.

We then used this breast-cancer ontology to generate a new knowledge-acquisition tool that was custom-tailored for specification of breast-cancer protocols. Figure 12 shows a portion of this tool, which allows developers to enter specifications for the algorithm of a breast-cancer protocol. Note that the elements of the palette on the left side of the figure correspond to subclasses of the STEP-AND-TRANSITION class modeled in the breast-cancer ontology (see Figure 11). The breast-cancer knowledge-acquisition tool was functionally similar to the tool used to acquire AIDS protocols, but was custom-tailored to include the concepts in the breast-cancer ontology that were relevant for the new class of protocols that we wished to acquire. Given this tool, it was straightforward for us to enter descriptions of complex breast-cancer clinical-trial protocols so that, ultimately, the components in EON could access the protocol knowledge to generate recommendations for protocol-based care in this new domain.

The breast-cancer ontology, knowledge-acquisition tool, and individual protocol knowledge bases were different from their counterparts in the AIDS domain. However, we made no changes to ESPR, RÉSUMÉ, or Chronus to build a version of "T-HELPER" that automated protocol-based care for breast-cancer patients. We simply reused the core components of the EON architecture to construct a workstation that had the same behavior as did T-Helper, but that provided treatment advice regarding protocol-based

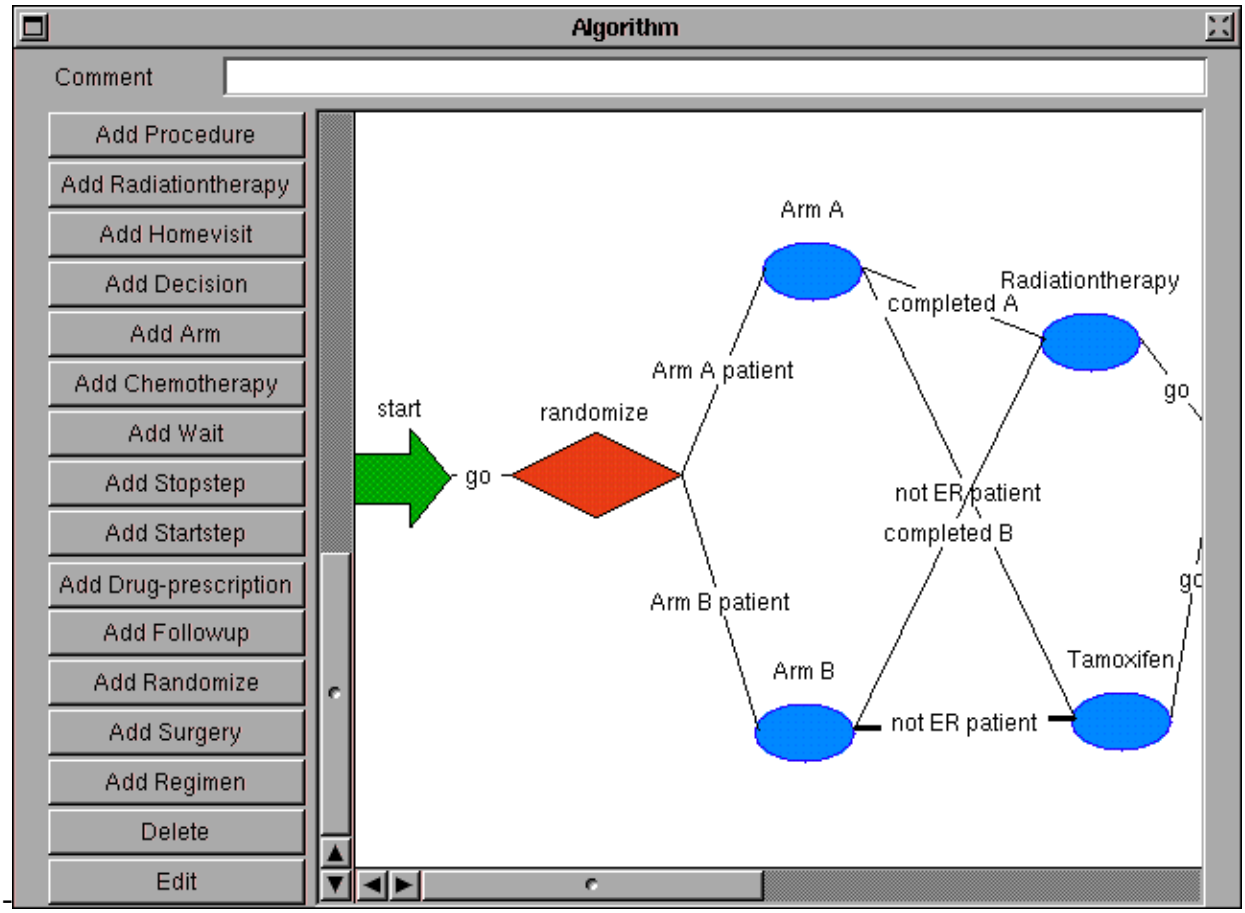

Figure 12: This portion of a knowledge-acquisition tool for entry of breast-cancer protocols was generated automatically from the ontology shown in Figure 11. The protocol shown in the graph randomizes patients either to Arm A or Arm B (chemotherapies entered via yet other graphs that a user accesses by double-clicking the corresponding oval). Those patients who complete their respective chemotherapies receive additional radiation therapy; patients who do not meet certain criteria regarding their estrogen-receptor status go on to receive the drug tamoxifen.

care of breast cancer, rather than of AIDS. Thus, the core components of the EON architecture could be reused directly; the only new programming that we did was to make a some simple domain-specific modifications to the T-HELPER user interface—a matter of a few days' work. As expected, the only significant knowledge-acquisition activity was related to development of the breast-cancer ontology and entry of the breast-cancer protocols into the new knowledge-acquisition tool. Thus, reuse of the EON components allowed us to construct, in only a few weeks, a new clinical information system for support of protocol-based care in breast cancer. Nearly all our time was spent refining the breast-cancer ontology; little time was needed for programming.

We have not installed the breast-cancer system in a clinical setting or evaluated formally the system's treatment recommendations. Our goal has been only to demonstrate reuse of the EON components in this new application area, and to confirm the ability of the components to operate on a different body of domain knowledge.

7 Discussion

The EON architecture comprises a number of components: (1) problem solvers, such as the ESPR-based therapy planner; (2) the RÉSUMÉ system for performing knowledge-based temporal abstraction; (3) the Chronus historical database query system, which supports the TLSQL temporal query language; and (4) the particular knowledge bases of protocols and guidelines that developers create using the knowledge-acquisition tools generated by PROTÉGÉ-II. Although each of these components has been described in the literature previously, it is useful to see how they interrelate to provide a task-specific architecture for the automation of protocol-based therapy.

The decomposition of EON into the components described in this paper emphasizes our view that there are recurring subtasks within the domain of protocol-based care, each of which can be addressed profitably by particular components. Because the interfaces between these components are well defined, developers can easily modify individual components to address changing software requirements, and reinsert the new components into the overall architecture. For example, one might want to alter the *temporal-pattern–matching* mechanism of the RÉSUMÉ system to enhance EON's ability to recognize diurnal changes in patient data; reprogramming would be constrained to a single component and would not affect other modules in the system (although extensions to the protocol knowledge base might be required). Developers also can reuse the EON components to address application tasks that have nothing to do with protocol-based care—such as incorporation of Chronus within other software systems that require support for temporal data queries. As clinical software systems address application tasks of burgeoning complexity, we see increasing advantages for systems that cleanly separate computational functionality into well-described modules, and that consolidate all application-specific knowledge into a discrete knowledge base that can be created and edited with easy-to-use tools.

7.1 Addressing the Complexity of Protocol-Based Care

The EON system clearly has more computational machinery than it would if we encoded protocols using more familiar approaches, such as the Arden syntax.[39] All systems that support protocol-based care, however, must be able to abstract generalizations about the patient from point data, must support archival storage and retrieval of time-related patient data, and must provide a means by which developers can review the knowledge of existing protocols and can encode the knowledge of new treatment specifications. Computational approaches to guideline-directed therapy that do not support these functions have significant limitations. To represent substantial protocols in the Arden syntax, for example, developers currently must write complex programs for large numbers of medical-logic modules that may interact with one another in ways that are unexpected and difficult to debug.

The use of medical-logic modules in the Arden syntax is similar to the use of production rules in first-generation expert systems. The aggregations of medical-logic modules required to encode knowledge of clinical protocols do not allow for an explicit representation of abstract protocol concepts, and modules must interact by posting intermediate conclusions to a global database.[40] The problems of rule-based programming are now well understood by the knowledge-based–systems community,[12, 13] and have led to the development of second-generation expert systems[14] in which knowledge is defined in principled ways. In second-generation systems, cleaner separation between *content knowledge* (e.g., the knowledge entered into a PROTÉGÉ-II–generated knowledge-acquisition tool) and *control knowledge* (e.g., the solution strategies encoded in problem-solving methods such as ESPR) leads to systems that are easier to build and maintain. Modularization of control knowledge in terms of reusable problem-solving methods also leads to software solutions in which system behavior can be better understood, and in which substitution of alternative problem-solving methods facilitates adaptation of systems to new problem-solving requirements.[8] Although such modern architectures for knowledge-based systems are only now

becoming prevalent, the potential complexity of the task of performing protocol-directed therapy demands a computational approach in which the control knowledge is represented in an explicit and modular manner. The composition of problem-solving methods and mechanisms in the EON architecture provides precisely this kind of explicitness and modularity.

As a second-generation expert-system architecture, EON derives a significant advantage from its relationship with PROTÉGÉ-II. PROTÉGÉ-II provides both a methodology and a set of tools for configuring problem-solving methods such as ESPR, and for constructing declarative ontologies of clinical application areas. In the case of the T-HELPER system, we have used PROTÉGÉ-II to construct a knowledge-acquisition tool tailored to the requirements of HIV-related protocols, and have used this tool to enter specifications for more than a dozen clinical trials. Because the knowledge-acquisition tool presents the contents of each protocol knowledge base in domain-specific terms, nonprogrammers can enter new information into the tool, and can browse through existing protocol specifications. At the same time, the modular configuration of problem-solving building blocks used to construct both the ESPR method and the RÉSUMÉ system makes the computational elements of the architecture easier for system developers to maintain and enhance.

We view the integration of EON with PROTÉGÉ-II, and the attendant advantages for knowledge acquisition and maintenance, as an important element of our architecture. Previous approaches to providing decision support for protocol-directed therapy have tended to ignore the significant problems of building large knowledge bases of protocol descriptions and of maintaining those knowledge bases over time. As we demonstrated with the OPAL knowledge-acquisition tool[3] for the ONCOCIN system,[2] it is extremely advantageous for health-care workers to be able to view and edit protocol knowledge that is presented to them in an intuitive fashion. Domain-specific knowledge-acquisition tools, such as those generated by PROTÉGÉ-II, have the potential to allow collaborating clinicians to take the responsibility for managing their own libraries of computer-encoded protocols. Given that protocols and guidelines may be highly specific to particular health-care organizations, and that guidelines often change over time, any scalable technology for assisting protocol-based therapy should provide this capability.

7.2 Reusing Standard Components

The design philosophy that we have adopted in EON presumes that protocol-based care is a frequent task in medicine, and that, within the domain of protocol-based care, there are subtasks that also must be addressed on a recurring basis. We seek to develop not only a computational framework that can be reused for tasks such as protocol-based therapy planning and protocol-eligibility determination, but also a set of subcomponents (such as RÉSUMÉ and Chronus) that we similarly predict will be reusable in addressing a number of clinical tasks—some of which are not even in the domain of protocol-based care. A major assumption in the construction of all component-based architectures is that developers will be able to predict which software elements are likely to be reused frequently, and that the relatively high cost of building a component with reuse in mind can be amortized over all subsequent applications of that component. For example, it will always be easier to write programs that perform temporal abstraction in limited, ad hoc ways than it will be to construct general-purpose shells such as RÉSUMÉ. The construction of a generic shell requires a developer to consider potential operations that may never be relevant for the task at hand. The need for reusability also places significant constraints on the developer to ensure that there are no domain-dependent assumptions built into the component. As a consequence, components such as RÉSUMÉ and Chronus can be complex to build and to use. We believe that the attendant complexity of the EON components is more than compensated by their potential for reuse. Our experience to date in reusing the integrated components, however, is limited to the AIDS and breast-cancer domains. (RÉSUMÉ by itself has been reused to solve a wide variety of tasks, such as monitoring of patients with diabetes,[6] assessing the growth of children,[32] and even determining problems in urban traffic flow[41]; Chronus has been reused not only within the T-HELPER application, but also within the

DoseChecker program for screening physicians' use of drugs in patients with renal impairment, developed by Kahn's group at Washington University in St. Louis.[42])

We do not claim that the components that we have currently assembled to create EON represent an optimal collection of modules. We merely observe that the requirements of decision support for protocol-based care include considerations that the EON components address in a direct manner, and that the EON components are thus far highly reusable—both to automate new protocol-related tasks (such as in the case of the breast-cancer domain) and to facilitate other tasks outside of protocol-based care (such as the use of RÉSUMÉ to interpret pediatric growth charts[32]). We currently are developing new problem solvers for protocol-based care in which the EON components interact in new ways. For example, we probably can make the ESPR therapy planner more efficient if we couple RÉSUMÉ and Chronus more tightly. We therefore are developing a new version of EON in which RÉSUMÉ and Chronus are integrated into an aggregate module that both queries the patient database and performs temporal abstractions as required by ESPR. This new component thus can serve as the primary conduit between the therapy planner and the patient database, and can perform necessary abstractions—and cache those abstractions and other intermediate results for later use—in a proficient, transparent manner. Ultimately, the particular assembly of components selected to automate any task, and the control-flow and data-flow relationships among the components, must be optimized both for their overall run-time efficiency and for the clarity of the aggregate software architecture.

The fact that there may not be one "best" assembly of components for automating a particular task does not diminish the utility of taking a modular approach to the construction of a software architecture. Although reuse of any piece of software presents a difficult problem—particularly when that software has been created to be maximally general and abstract—there are obvious advantages when developers can avoid programming de novo to solve new tasks.[1] The challenge, of course, is to demonstrate that individual software components have sufficient value that developers will take the time to learn how to apply them in new situations. Otherwise, software engineers will always retain the perception that it is more expedient to reprogram every solution from scratch.

In the case of EON, we have developed each of the software components not only so that the overall EON architecture can be reusable for protocol-based care in novel medical domains, but also so that the individual components (namely, ESPR, RÉSUMÉ, and Chronus) will be reusable to automate tasks outside of those related to protocol-based care. This design decision undoubtedly adds to the complexity of the overall architecture, but will allow us to continue to experiment with the EON components in building new decision-support systems. At the same time, the general way in which the components are implemented (C-based programs that assume nothing more than the existence of data stored in relational tables) will allow us ultimately to embed the EON components in a variety of information systems for addressing the task of protocol-based care, which inspired the development of those components in the first place. Although the existing version of EON was developed before the emergence of standard interfaces for interoperabiliy among software components, we are working on a new version of EON in which all elements of the architecture will use CORBA to communicate with one another.

7.3 Related Work

Most workers in medical informatics have explored rule-based approaches when attempting to automate protocol-based care; several groups also have investigated alternative software architectures. The M-HTP system,[43] for example, incorporates reasoning strategies that perform many of the temporal-abstraction functions achieved by RÉSUMÉ. M-HTP monitors patients who are undergoing heart transplantation, but it is not designed to be readily reusable in new domains. In particular, the system does not have a well-defined temporal-abstraction ontology that establishes the general properties of clinical parameters that might be abstracted in different contexts. Nevertheless, the analysis of the heart-transplantation domain

required to build M-HTP confirms the importance of the classes of temporal abstractions performed by the RÉSUMÉ system.

Unlike EON, M-HTP lacks a temporal query component that could allow the program to access patient data in an external archival database. Whereas M-HTP was built as a standalone system, the EON architecture was designed to be embeddable within more complex clinical information systems. EON's Chronus component has the designated function of mediating temporal queries between an external relational database and the other components within EON. A limitation, however, is that Chronus assumes that the data in the relational database are stored in a particular format—one that includes start times and end times for each tuple (see Figure 9). We are developing a more general approach, in which a temporal database *mediator* will allow EON to issue Chronus-style TLSQL queries on data stored in legacy databases that may not adopt the particular data schema that Chronus now requires.[44]

Recently, a consortium of workers in Europe has developed a *general protocol and guideline model* as part of the DILEMMA project.[45] Many aspects of the DILEMMA model correspond with elements of the domain ontology for protocols constructed with PROTÉGÉ-II. Many concepts in the broad DILEMMA model, however, are not captured by the ontologies that we have created with PROTÉGÉ-II for use by EON (e.g., organizational aspects of protocol administration). Whereas the PROTÉGÉ-II philosophy provides for the possibility of *multiple* protocol ontologies—each one specialized for a particular area of medicine—the DILEMMA consortium has developed an overarching model of protocol-based care that might be applied in a wide range of clinical settings. A consequence of this considerable generality is that the DILEMMA model is of necessity less specific than the more narrowly focused domain ontologies for protocol-based care that we have created using PROTÉGÉ-II. The DILEMMA model has inspired the development of small-scale systems in which much of the protocol-specific knowledge has been hand-coded in PROLOG. Use of the model to guide the acquisition of protocol knowledge directly has not yet been described in the literature.

The EON approach should be contrasted with those of a number of investigators who are experimenting with improved access to protocol information stored as text or as multimedia. GEODE-CM,[27] for example, is a standalone system that allows health-care workers to follow a state-transition diagram that expresses the logic of clinical algorithms. The system also permits users to view hypermedia documents that relate to the general clinical situation and to the potential decisions that must be made in association with each state. Other approaches provide hypertext browsing of guidelines via the World Wide Web, and several even use embedded rule-based systems to provide situation-specific recommendations based on the user's manual entry of clinical information into computer-generated forms.[46, 47] None of these systems currently allows data stored in an electronic patient record to invoke appropriate protocol logic in a transparent fashion. At the same time, encoding of protocols as elementary state-transition tables or as situation–action rules does not permit representation of protocol logic in a manner that allows either the complexity or the clarity of the representations used for EON. For example, if the protocols in T-HELPER were represented only as state-transition tables, then transitions from one state to another would have to be predicated on only primary patient data, because there would not be a direct mechanism to invoke a system such as RÉSUMÉ to generate the appropriate temporal abstractions. Furthermore, the complexity of the transition network would increase exponentially with each additional clinical contingency that could cause modification of the standard treatment plan. In the ESPR therapy planner, the use of plan-revision rules that update the standard plan in light of newly identified problems provides an extremely compact representation for much of the protocol knowledge.

7.4 Application of EON

We have tested the EON architecture within the context of both the T-HELPER system—which has been deployed at a county-operated AIDS clinic near Stanford University—and our experimental system for breast-cancer care. Our experience demonstrates that EON's ability to reason about protocol-based care is

sufficient to provide therapy recommendations for a wide range of AIDS-related clinical trials, including antiretroviral studies and trials of antibiotics for primary and secondary prophylaxis of opportunistic infections.[9] Ongoing work in our laboratory explores the extension and further integration of the components in EON, as well as our application of the approach to a wider range of clinical protocols.

To date, most of our experience with EON has concerned reasoning about therapy within clinical trials. Although most clinical-trial protocols are far more complex than are most practice guidelines, clinical trials represent an unusual situation in medicine: Patients with significant comorbidity generally are excluded from clinical trials, and therapy specifications in clinical trials are as precise as possible. Clinical trials represent controlled experiments, and the corresponding protocols must be as reproducible as possible. Practice guidelines, on the other hand, tend to be far less prescriptive, and may allow latitude—both in defining the clinical situations that trigger application of a guideline, and in administering the interventions that are mandated by the particular clinical situation. Such leeway is appropriate in situations where there is no normatively preferred treatment, and where it is impossible to enumerate the myriad contingencies that may influence the provider's decision making.

The gross ambiguity of many guidelines precludes our ability to model them as deterministic therapy plans on which EON can operate. Although the knowledge-based temporal-abstraction method in RÉSUMÉ does allow identification of clinical situations that may not be fully specified, the planning subtasks of the ESPR method currently assume that treatment recommendations should be unique, and not permit multiple competing alternatives. Nevertheless, in situations where a practice guideline is ambiguous regarding appropriate therapy, it may be more helpful to offer a *critique* of a provider's intended treatment[48] than to suggest a specific therapy recommendation. Our group consequently plans to develop new problem-solving methods that can compare current therapy to that suggested by applicable guidelines, and that can identify significant discrepancies. One promising line of research involves exploiting both domain-specific knowledge about goals and policies of particular protocols, and domain-independent knowledge about possible guideline-revision strategies.[49] This approach would allow a problem solver to reason about whether a health-care worker's actions are consistent with a guideline's intended actions, even if the clinician does not follow the algorithm specified by the guideline precisely. Such a critiquing system would require us to create novel problem-solving methods that can compare a provider's actions with those intended by the relevant guideline. In building this system, we still should be able to reuse RÉSUMÉ to solve the temporal-abstraction task required by the new critiquing methods, and to reuse Chronus to retrieve interval-based data from the electronic patient record.

Acknowledgments

This work has been supported in part by grants LM05708 and LM05304 from the National Library of Medicine, by grant HS06330 from the Agency for Health Care Policy and Research, by grant CA65426 from the National Cancer Institute, and by contract N66001-94-D-6052 supported by the Defense Advanced Research Projects Agency. Dr. Musen is the recipient of National Science Foundation Young Investigator Award IRI-9257578.

We are grateful to Doug Fridsma, John Gennari, and Peter Johnson for ongoing discussions related to the automation of protocol-based care, to Zaki Hasan for software development, and to John Egar for his art work. Diane Oliver and Tom Rindfleisch provided valuable comments on a previous draft of this paper. The comments of the anonymous reviewers were extremely helpful in our revisions of the manuscript.

Additional information about EON and related projects is available via World-Wide Web URL http://www-smi.stanford.edu/projects/eon/index.html.

References

1. Krueger CW. Software reuse. *ACM Computing Surveys*. 1992; **24**:131–183.

2. Tu SW, Kahn MG, Musen MA, Ferguson, JC, Shortliffe, EH, Fagan, LM. Episodic skeletal-plan refinement on temporal data. *Communications of the ACM*. 1989; **32**:1439–1455.

3. Musen MA, Fagan LM, Combs DM, Shortliffe, EH. Use of a domain model to drive an interactive knowledge-editing tool. *International Journal of Man–Machine Studies*. 1987; **26**:105–121.

4. Musen MA. Automated support for building and extending expert models. *Machine Learning*. 1989; **4**:349–377.

5. Musen MA, Carlson CW, Fagan LM, Deresinski, SC, Shortliffe, EH. T-HELPER: Automated support for community-based clinical research. In: Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care, Baltimore, MD; 1992:719–723.

6. Shahar Y, Musen MA. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*. 1996; **8**:267-298.

7. Musen MA. Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research*. 1992; **25**:435–467.

8. Musen MA, Schreiber AT. Architectures for intelligent systems based on reusable components. *Artificial Intelligence in Medicine*. 1995; **7**:189–199.

9. Tu SW, Eriksson H, Gennari JH, Shahar, Y, and Musen, MA. Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: Application of PROTÉGÉ-II to protocol-based decision support. *Artificial Intelligence in Medicine*. 1995; **7**:257–289.

10. Das AK, Tu SW, Purcell G, Musen, MA. An extended SQL for temporal data management in clinical decision-support systems. In: Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care, Baltimore, MD; 1992:128–132.

11. Das AK, Musen MA. A temporal query system for protocol-directed decision support. *Methods of Information in Medicine*. 1994; **33**:358–370.

12. Heckerman DE, Horvitz EJ. *The myth of modularity in rule-based systems for reasoning with uncertainty*. In: Lemmer JF, Kanal LN, eds. *Uncertainty in Artificial Intelligence.* Amsterdam: North Holland, 1988; **2**:23–34.

13. Clancey WJ. The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence*. 1983; **20**:215–251.

14. David J-M, Krivine J-P, Simmons R., eds. *Second Generation Expert Systems*. Berlin: Springer-Verlag; 1993.

15. Musen MA, Gennari JH, Eriksson H, Tu, SW, Puerta, AR. PROTÉGÉ-II: Computer support for development of intelligent systems from libraries of components. In: Proceedings of MEDINFO ’95, Eighth World Congress on Medical Informatics, Vancouver BC; 1995:766–770.

16. Regoczei S, Plantinga EPO. Creating the domain of discourse: Ontology and inventory. *International Journal of Man-Machine Studies*. 1987; **27**:235–250.

17. Guarino N. Concepts, attributes, and arbitrary relations: Some linguistic and ontological criteria for structuring knowledge bases. *Data and Knowledge Engineering*. 1992; **8**:249–261.

18. van Heijst G, Falasconi A, Abu-Hanna G, et al. A case study in ontology library construction. *Artificial Intelligence in Medicine*. 1995; **7**:227–255.

19. Marques D, Dallemagne G, Klinker G, McDermott, J, Tung, D. Easy programming: empowering people to build their own applications. *IEEE Expert*. 1992; **7**:16–29.

20. Gennari JH, Tu SW, Rothenfluh TE, Musen, MA. Mapping domains to methods in support of reuse. *International Journal of Human–Computer Studies*. 1994; **41**:399–424.

21. Tu SW, Kemper CA, Lane NM, Carlson, RW, Musen, MA. A methodology for determining patients' eligibility for clinical trials. *Methods of Information in Medicine*. 1993; **32**:317–325.

22. Shahar Y, Musen MA. RÉSUMÉ: A temporal-abstraction system for patient monitoring. *Computers and Biomedical Research*. 1993; **26**:255–273.

23. Musen MA, Tu SW, Shahar Y. A problem-solving model for protocol-based care: From e-ONCOCIN to EON. In: Proceedings of MEDINFO '92, Seventh World Congress on Medical Informatics, Amsterdam: North-Holland; 1992:519–525.

24. Friedland PE, Iwasaki Y. The concept and implementation of skeletal plans. *Journal of Automated Reasoning*. 1985; **1**:161–208.

25. Silverman HA. A digitalis therapy advisor. Technical Report MAC/TR-143, Massachusetts Institute of Technology, Cambridge, MA; 1975.

26. Chandrasekaran B. Design problem solving: a task analysis*AI Magazine*. 1990; **11**(4):59–71.

27. Stoufflet PE, Deibel SA, Traum JH, Greenes, RA. A state-transition method of modeling clinical encounters. In: Proceedings of the AMIA Spring Congress, Boston, MA; 1995:81 (abstract).

28. Eriksson H, Shahar Y, Tu, SW, Puerta AR, Musen, MA. Task modeling with reusable problem-solving methods. *Artificial Intelligence*. 1995; **79**:293–326.

29. Kahn MG. Modeling time in medical decision-support programs. *Medical Decision Making*. 1991; **11**:249–264.

30. Shahar Y. A knowledge-based method for temporal abstraction of clinical data. Program in Medical Information Sciences, Ph.D. dissertation, Stanford University, Stanford, CA; 1994.

31. Shoham Y. Temporal logics in AI: semantical and ontological considerations. *Artificial Intelligence*. 1987; **33**:89–104.

32. Kuilboer MM, Shahar Y, Wilson DM, Musen, MA. Knowledge reuse: Temporal-abstraction mechanisms for the assessment of children's growth. In: Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care, Washington, DC; 1993:449–453.

33. McKenzie LE, Snodgrass RT. Evaluation of relational algebra incorporating the time dimension on databases. *ACM Computing Surveys*. 1991; **23**:501–543.

34. Tansel AU, Clifford J, Gadia S. *Temporal Databases: Theory, Design, and Implementation*. Redwood City, CA:Benjamin/Cummings; 1993.

35. Gennari JH, Altman RB, Musen MA. Reuse with PROTÉGÉ-II: From elevators to ribosomes. In: Proceedings of SSR '95: ACM SIGSOFT Symposium on Software Reusability, Seattle, WA; 1995:72–80.

36. Rothenfluh TE, Gennari JH, Eriksson H, Puerta, AR, Tu, SW, Musen, MA. Reusable ontologies, knowledge-acquisition tools, and performance systems: PROTÉGÉ-II solutions to Sisyphus-2. *International Journal of Human–Computer Studies*. 1996; **44**:303–332.

37. Musen MA, Gennari JH, Wong WW. A rational reconstruction on INTERNIST-I using PROTÉGÉ-II. In: Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, New Orleans, LA; 1995:289–293.

38. Musen MA, Wieckert KE, Miller ET, Campbell, KE, Fagan, LM. Development of a controlled medical terminology: Knowledge acquisition and knowledge representation. *Methods of Information in Medicine*. 1995; **34**:85–95.

39. Hripcsak G, Ludemann P, Pryor TA. Rationale for the Arden syntax. *Computers and Biomedical Research*. 1994; **27**:291–324.

40. Sherman EH, Hripcsak G, Starren J, et al. Using intermediate states to improve the ability of the Arden syntax to implement care plans and reuse knowledge. In: Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, New Orleans, LA; 1995:238–242.

41. Molina M, Shahar Y. Problem-solving method reuse and assembly: from clinical monitoring to traffic control. In: Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada; 1996.

42. Abrams CA, Kahn MG, Steib SA, et al. An expert system for dosing renally excreted drugs. In: Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, New Orleans, LA; 1995:960 (abstract).

43. Larizza C, Moglia A, Stephanelli M. M-HTP: A system for monitoring heart-transplant patients. *Artificial Intelligence in Medicine*. 1992; **4**:111–126.

44. Das AK, Shahar Y, Tu SW, Musen, MA. A temporal-abstraction mediator for protocol-based decision support. In: Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC; 1994:320–324.

45. Herbert SI. Informatics for care protocols and guidelines: Toward European knowledge model. In: Gordon C, Christensen JP, eds. *Health Telematics for Clinical Guidelines and Protocols*. Amsterdam: IOS Press; 1994:27–42.

46. Barnes M, Barnett GO. An architecture for a distributed guideline server. In: Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, New Orleans, LA; 1995:233–237.

47. Liem EB, Obeid JS, Shareck PE, Sato, L, and Greenes, RA. Representation of clinical practice guidelines through an interactive world-wide-web interface. In: Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, New Orleans, LA; 1995:223-227.

48. van der Lei J, Musen MA. A model for critiquing based on automated medical records. *Computers and Biomedical Research*. 1991; **24**:344–378.

49. Shahar Y, Musen MA. Plan recognition and revision in support of guideline-based care. In: Working notes of the AAAI Spring Symposium on Representing Mental States and Mechanisms, Stanford, CA; 1995:118–126.