Knowledge Systems Laboratory
Working Paper No. KSL 87-32

April 1987

Diagnosis, Teaching, and Learning: An Overview of GUIDON2 Research

William J. Clancey
Institute for Research on Learning
2550 Hanover St.
Palo Alto, CA 94304

In Proceedings of AI-87, Osaka, Japan.

Diagnosis, Teaching, and Learning: An Overview of GUIDON2 Research

William J. Clancey

Computer Science Department
Stanford University

Abstract

We are developing a family of tutoring programs, collectively called GUIDON2, which enable a student to watch, explain, debug, and learn in apprenticeship the *process* of diagnosis. This research aims to develop cognitive models of human problem solving and learning in expert systems, and to use these models as a foundation for a teaching program. A theory of learning, based on the role of metaknowledge in detecting and coping with problem-solving failures, is being tested in a knowledge acquisition program, which will serve as a standard for evaluating and assisting a student.

Two perspectives on the nature of knowledge engineering interact in the design of GUIDON2 instructional programs: We view knowledge engineering as a modeling methodology, and we view knowledge engineers as students, who use domain-general knowledge about representations and problem-solving procedures to critique and improve qualitative models. In GUIDON2, we are investigating whether we can teach students to play an active role in directing their own learning, by analogy with the methods for detecting and coping with failure used by a knowledge engineer.

1. Background

Artificial intelligence (AI) is a branch of computer science that develops programming techniques for modeling processes qualitatively, that is, in terms of explicit representations that describe spatial, temporal, and causal relations among objects, as opposed to just numeric measures. Two basic methods exist for representing processes: as a classification of types of processes (e.g., a taxonomy of recurring faults in equipment) and as a simulation, either as a causal state-transition network, or as a hierarchical, structural-functional model. While AI research originally emphasized the representation of problem-solving processes, application to scientific and engineering problems has shifted concern to modeling systems in the world (Clancey, 1986a, Clancey, 1985a).

AI-based instructional programs, often called *intelligent tutoring systems* (ITS), use these qualitative modeling techniques to represent: 1) processes in the subject domain (e.g., a steam propulsion plant, an electronic circuit), 2) problem-solving processes (e.g., diagnostic strategy, programming methods), and 3) communication processes (e.g., the Socratic method, case-method discourse, and rhetorical principles in explanation) (Clancey, 1986b). Typically, instructional programs may represent only one or two kinds of these processes. When a simulation model of problem-solving processes is incorporated in the program, a basis is provided for evaluating and assisting the student in a very general way. Such programs, which can solve the same problems given to a student, are called *knowledge-based tutors* (Clancey, 1987a).

Research in the ITS field focuses on the three areas described above: qualitative modeling of system processes, problem-solving processes, and instructional discourse processes. Research involves simulating these processes, involving both empirical work to develop the content of models and AI programming to develop techniques for representing

the models. Research generally proceeds incrementally, by cyclical critique and re-representation. Research also proceeds by building models on top of each other, in the way GUIDON was constructed on top of MYCIN (Clancey, 1982). In this way, student modeling, explanation, and instructional discourse research has tended to follow advances in modeling problem-solving.

In the late 1980's, a major opportunity has opened for exploiting problem-solving research of the past decade. In particular, it is now clear that when problem-solving processes are factored from the model of the subject domain, a knowledge acquisition (learning) program is better able to identify the causes of problem-solving failures (Smith, et al., 1985, Mitchell, et al., 1985). Of considerable interest to the ITS community, these models of learning are based on a process of explaining problem-solving, using metaknowledge about strategy and knowledge organization (Mitchell, et al., 1986, DeJong and Mooney, 1986).

Better representations and models of learning make this a particularly productive time for implementing knowledge acquisition (learning) programs as the basis of computer tutors. This research overview describes an approach that makes explicit the relation between: 1) the constraints a problem-solver is seeking to satisfy (the form of a good solution), 2) the language used for representing the real-world system being reasoned about, and 3) a psychological model of the problem-solving process.

2. Project Accomplishments

Early in our research, we identified the importance of representing problem-solving processes in a well-structured procedural language. This enables the explanation and student modeling programs to reason about the problem-solving process, so it can be selectively articulated and identified in student behavior. In a sequence of programs, we demonstrated basic AI techniques for achieving the separation of domain facts from a diagnostic procedure (NEOMYCIN), and the advantages of this separation for explanation (Hasling, et al., 1984) and student modeling (IMAGE, ODYSSEUS). A sequence of articles describe the NEOMYCIN program as a psychological model of diagnosis (Clancey, 1984), an architecture for representing strategic knowledge (Clancey, 1985b), and as a general problem-solving method (called "heuristic classification") (Clancey, 1985a). Other publications generalizing this research and synthesizing related work have appeared in the past year (Clancey, 1987b, Clancey, 1986b, Clancey, 1986c, Clancey, 1987a).

In 1984 we began to develop a family of tutoring programs built upon NEOMYCIN, or more precisely, the general shell out of which it is constructed, HERACLES. Research accomplishments in the past two years include:

- Completion of GUIDON-WATCH, a program that integrates the HERACLES consultation program with a graphics browser and explanation program, refined through a series of student trials (Richer and Clancey, 1985). The most important window is the "situation-specific model" which shows a diagnosis in the form of a proof graph, with abnormal findings explained by successively more specific causal and subtype descriptions of processes towards the top of the graph. Other key windows include: the line of reasoning displayed as a "task stack" and a summary of evidence displayed as a table.

- Demonstration of HERACLES's generality through the development of a knowledge base for sandcasting diagnosis called CASTER (faults in iron cast in sand molds) (Thompson and Clancey, 1986).

- Demonstration of a prototype knowledge-acquisition program, GUIDON-DEBUG, that integrates the GUIDON-WATCH browsing capabilities with the ODYSSEUS modeling program and grapic editing. By this design, an edited consultation typescript is analyzed by ODYSSEUS to determine how the knowledge base would have to be modified to produce the new sequence of data requests, consistent with the diagnostic procedure.

- Extension of ODYSSEUS to incorporate metarules in its model of student reasoning, rather than just sequences of diagnostic tasks. (Thus replacing hand-coded heuristics and enabling the program to read the metarules to make analyses required by GUIDON-DEBUG.)

- Re-implementation of the explanation program with improved bookkeeping of consultation reasoning, enabling "roll back" of GUIDON-WATCH's display to any moment during a previously run consultation (for student exploration) and explanation of reasoning at the level of individual metarule clauses (replacing hand-coded text strings associated with metarules). Menus created dynamically, in the context of particular student inquiries, bring together relevant related information, enabling a student to get detailed information without browsing through the extensive menu system.

- Development of a prototype tutoring system, called GUIDON-MANAGE, in which HERACLES carries out a student's diagnostic tasks (e.g., to test a particular hypothesis or determine the implications of a new piece of problem information). Research has focussed on interpreting the diagnostic process to carry out the tasks (trapping at intermediate levels of detail that we expect a student to detect and complete on his own) and to provide problem-solving assistance (simulating what the program would do next). Research has also considered, but not solved, the problem of providing appropriate feedback so the student understands what each command to the program accomplished.

- Development of a script-based graphics display program, GUIDON-TOURS, a facility for automating subject matter lectures or system documentation. An interactive program allows the script writer to design a sequence of window displays and text, which a student or programmer plays back, pausing and exploring the display at will.

- Re-implementation of HERACLES, repartitioning knowledge files so domain-general components are better separated from specific knowledge bases. This basic system maintenance will allow us to export the program to other sites, as well as to provide a foundation for re-using and extending the procedural language in future research (e.g., solving design problems or adapting the program to use an agenda).

Significant active research, extending the programs described above includes:

- Re-representation of tasks to make explicit the constraint each seeks to accomplish in modifying the situation-specific diagnostic model. Proof of concept demonstration of detecting failure to satisfy a constraint and reading the metarules to conjecture missing domain knowledge (leading towards full implementation of the theory of learning in the GUIDON-DEBUG knowledge acquisition program, and providing the basis for proposed instructional research (see Section 4).

- Explanation during a consultation at the level of metarule clauses, selectively constructing simple paragraphs for "WHY" explanations, using heuristics for leaving out details.

- Extension of the task/metarule (procedural) language, allowing the explanation program to be represented in the same language (so potentially it can reason about itself). Implementation of a compiler to combine already compiled metarules into Lisp functions for each task (enabling the explanation program to run more quickly and avoid stack overflow).

- Extension of GUIDON-MANAGE to indicate changes to the situation-specific model resulting from each student command, and implementation of the explanation

program within GUIDON-MANAGE to allow explanations of what a task did and why a task is being suggested (in response to a student request for help).

- Extension of ODYSSEUS to prune down the greatly increased search space that resulted when modeling heuristics were replaced by a more general program that reasons about the actual HERACLES metarules.

- Exportable version of HERACLES with facility to create a knowledge base using a graphics-based editor.

In addition to these programming projects, student trials are continuing for GUIDON-MANAGE and the explanation program.

3. Research Themes, Projects, and Theories

The general direction of the research is illustrated by Figure 1. Across the top are general research areas. In developing AI-based instructional programs, we are developing principles for the design of expert system shells, cognitive modeling, and knowledge acquisition.

The middle of the diagram indicates active programming projects. We are exploiting graphics for constructing knowledge bases and browsing reasoning. We are developing models of physical systems (e.g., physiology and sandcasting), called knowledge bases. We are developing a language in which the domain knowledge is represented declaratively in first-order predicate calculus and the inference procedure (so-called control knowledge or strategy) is representing procedurally (as tasks and metarules); this enables us to model the factual and strategic components of human reasoning. We are relating knowledge acquisition to student modeling (Wilkins, et al., 1986), showing how explanation and debugging capabilities are integral parts of interpreting behavior and completing an interpretation (learning). A subproject focuses on text generation for lines of reasoning explanations and summaries, extending previous work in natural language generation.

Finally, the lowest section indicates the most general theories we have formulated to support or justify design considerations in our programs. These theories begin as abstractions (generalizations) of existing programs; observed patterns are then restated as principles or rationales that could generate them (Clancey, 1986c). For example, this is the approach we took in reformulating MYCIN's knowledge base into NEOMYCIN and then abstracting the method of *heuristic classification* from this. The general direction is to describe AI programming as a methodology for representing models of processes (Clancey, 1986a, Clancey, 1985a). In the extreme, we relate these models to what people know and how they learn by considering the nature of representation (Clancey, 1987c, Clancey, 1987d).

We believe that the central issue in ITS research is the relation between learning, problem-solving (particularly diagnosis), and explanation. A convergence of ideas has developed as follows (refer to Figure 1):

- Diagnosis is viewed as a process of critiquing a model of a system behaving abnormally. Operators for diagnosis (HERACLES tasks) seek to explain the process occurring in causal terms by constructing an explanation graph.

- Debugging is one form of learning which involves detecting what operators failed to meet problem constraints (e.g., contrasting alternative explanations or refining an explanation to the point of being able to repair the system). Being able to articulate the problem-solving procedure (e.g., diagnostic strategy) allows articulating the failure in terms of missing knowledge.

- Explanation is viewed as a process of debugging a failed attempt to construct a situation-specific model. Thus, the model of inquiry for diagnosis and the

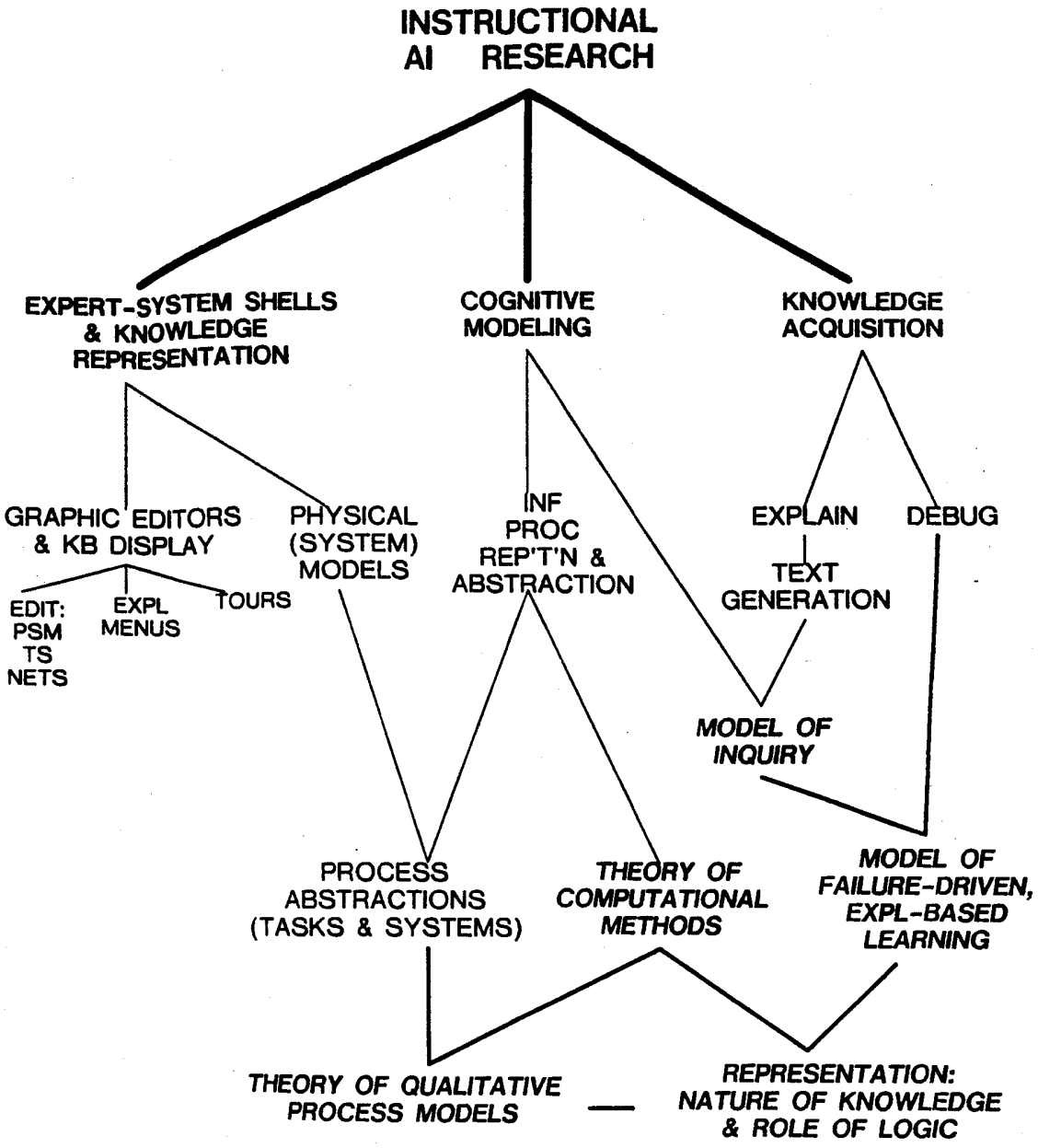

Figure 3-1: Relation of GENERAL RESEARCH AREAS to ACTIVE PROGRAMMING PROJECTS and *THEORIES*

model of debugging (which generates questions) come together in our study of explanation. Can we *explain a student's question* in terms of his failure to understand and "repair" his model by supplying the knowledge he is missing?

With this vision in mind, our research is now focusing on learning and explanation, rooted in the specific knowledge acquisition program we are constructing. GUIDON-DEBUG thus serves as a model of how good questions are generated by the failure to understand some process, both the abnormal real-world process being diagnosed and the program behavior the student is trying to understand. The relation between debugging and explanation is hardly more than a metaphor at this point; we will pursue the connection by showing the limitations and advantages of using GUIDON-DEBUG as an underlying model of learning.

4. GUIDON-DEBUG Details

GUIDON-DEBUG is a knowledge-acquisition program that detects and copes with problem-solving failures by relating problem constraints to the tasks and metarules, conjecturing missing domain knowledge from failed metarules, and reformulating these as questions to ask an expert. This program will be used as a standard for critiquing student behavior and providing assistance. That is, teaching will be oriented toward conveying problem constraints and the problem-solving procedure for achieving them.[1]

In more practical terms, our objective is to teach a student domain facts that will enable him to solve diagnostic problems by heuristic classification. For example, in the language of HERACLES, the student will learn classifications of findings and heuristics to relate them to classifications of solutions. He will learn to recognize and discriminate these prototypes. Using knowledge of the heuristic classification representation and inference procedure, the student will explain his failure to solve problems and direct a teacher to supply him with the facts about the world that he needs to know. A basic assumption is that learning will be more efficient by having the student determine what he needs to know, than having the teacher build a model of his knowledge, present factual lectures, and test him on cases. However, the student might *direct the teacher* to do any of these in the process of actively directing his learning.

In contrast with the model developed in GUIDON, we are not merely presenting information to the student, who must read facts and store them away. Instead, we focus on learning that occurs and is motivated by problem-solving failures. In contrast with GUIDON's original design, this is not a strategy for "filling in a knowledge base of the student." But again, the student might request an *orientation* at particular times, just as the knowledge engineer applies these methods early in the knowledge acquisition process.

Just as for a knowledge engineer, the student's learning is failure-driven, based on knowledge of what he is trying to do (the form of an adequate solution) and what failures occurred. Specifically the learning procedure is:

- Know what you are trying to do: constraints to satisfy (the form of a solution) and how to satisfy them (model-manipulation tasks).
- Detect possible failures (unsatisfied constraints) in the inferred, situation-specific model:
    - unable to test or refine a hypothesis;
    - unable to explain finding;
    - finding explained by two or more hypotheses;

---

[1] This section is excerpted from a working paper (Clancey, 1987d), which describes the theory of learning and its relation to teaching in detail.

- two or more hypotheses explain exactly the same findings and evidence doesn't discriminate between them, or they explain findings uniquely;
- situation-specific model hypotheses are not specific enough to select or construct action plans.

- Reason backwards to say what task, if it had succeeded, would have prevented this failure, and what facts, if true or proved false, would allow the metarule to succeed (the hypothesized gaps in the domain knowledge).
- Prune alternative explanations using knowledge of what beliefs typically could be wrong or might be true, but which were not explicitly learned before.
- Ask the teacher questions to gain missing knowledge or validate hypothesized facts.

For example, referring to Figure 2, we consider the diagnostic constraint that every abnormal finding must be explained by the most likely hypothesis. We observe that seizures is not explained. Relating this constraint to subtasks, we see that the HERACLES subtasks Test-hypothesis (applied to acute-bacterial-meningitis) and Process-Finding (applied to seizures) have associated inference rules (metarules) that would have satisfied this constraint if they had succeeded. Examining these metarules, we find that a domain rule linking seizures to acute-bacterial-meningitis (among others) would enable one or more of the metarules to succeed. Stating this hypothesized fact as a question, the student would ask, "Could acute-bacterial-meningitis cause seizures?"

Some of the limitations and complications of this model of learning are:

- It is based on a fixed knowledge representation language and inference procedure;
- It requires making explicit the constraints of a solution and how they relate to diagnostic tasks and metarules;
- It requires a search procedure to work causally backwards from failed constraints;
- It might require domain knowledge or domain-general knowledge about disorder processes in order to focus the search for plausible facts, if many possibilities are generated; and
- It may be necessary to relax the constraints imposed on the situation-specific model, given the pragmatic requirements of how the model will be used (e.g., the action plans it must discriminate between) and the inability to confirm hypothesized facts (e.g., lack of scientific understanding of causal mechanisms).

Resolving these uncertainties and filling in details are good reasons for implementing the model as a simulation program. GUIDON-DEBUG builds on the work we have already done in representing an inference procedure so it can be reasoned about, but requires annotations that make the constraints and design that lies behind the tasks and metarules more explicit.

5. Future Research

To this point in our research, we have developed an extensive sequence of programs from one root, the original MYCIN program. We now believe that it is time to branch out beyond the medical domain and beyond the heuristic classification problem-solving method. This process will take us into the next decade of research, and must proceed incrementally to properly build on our programming investment and to systematically follow the track of

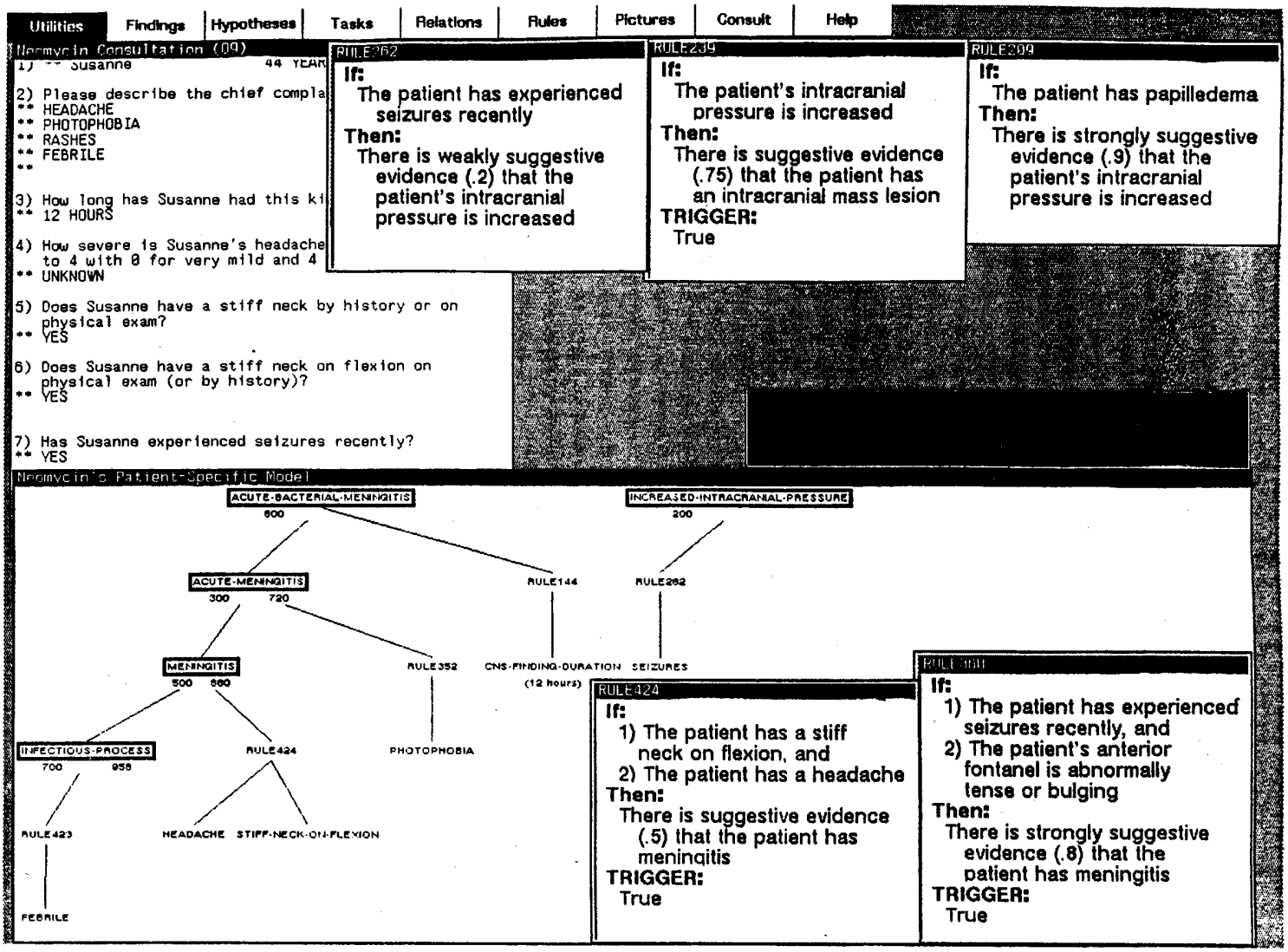

Figure 4-1: A diagnosis expressed as a patient-specific model in the form of a graph.[2]

---

[2] *In this case, data-directed reasoning from the chief complaints (questions 2-6) led to the application of rule424, putting meningitis into the patient-specific model (called triggering). Meningitis is supported by evidence for its more general category, infectious-process. The meningitis explanation is further refined by acute-meningitis and acute-bacterial-meningitis, each explaining other specific complaints. Further support for meningitis was sought via rule060, leading the program to ask question 7. Data-directed reasoning, through rule262, then put increased-intracranial-pressure in the patient-specific model. The program attempts to reason forward to explain this hypothesis, but before applying rule239, it seeks further support for increased-intracranial-pressure, through rule209. This leads to a question about papilledema, which is not shown.*

research ideas. Incremental changes to HERACLES will occur in several dimensions, most likely in parallel:

1. *Implementation*: It is now necessary to bring up-to-date the foundation of the program, originally designed in 1972, by replacing the primitive frame system of EMYCIN by an object-oriented design.

2. *Abstraction*: When HERACLES is available to domain specialists, such as engineering students, we expect to collect a number of knowledge bases for further study. Preliminary analysis indicates that modifications to the inference procedure will be common in building new programs, but these modifications can be expressed as specializations of common programming constructs (loops, filters, collectors, etc.). This project will involve reformulating the tasks of the inference procedure as instantiations of explicitly-represented primitive programming constructs. Similarly, the causal networks of the domain knowledge bases can be abstracted in terms of recurring processes (e.g., narrowed flow, insufficient feed). We believe that collecting such abstractions into a *knowledge base library* will be a major undertaking that will greatly improve knowledge engineering efficiency. Initially, we must identify examples of these abstractions and establish principles for their representation.

3. *Extension*: To better model student reasoning and to move beyond problems solved by heuristic classification, we believe that HERACLES must be reformulated to use an agenda. This involves another layer of interpretation for ordering and selecting tasks posed by the metarules (rather than executing them directly as they are generated). Agenda-based control of reasoning is important for incrementally piecing together system descriptions from primitives, rather than selecting them whole (e.g., from a fault process hierarchy) (Hayes-Roth, 1986).

At this time we cannot indicate precisely what other problem domains we will consider in developing a knowledge base library or in extending the HERACLES shell. An obvious initial engineering project is to continue development of CASTER. We expect to establish a collaboration at Stanford with one or more engineering departments to develop instructional programs. Some of the criteria we are considering for identifying good engineering problems are:

- The problem must strongly involve the *problem solving of individual people*, so that modelling cognitive processes is important.

- To be realistic, problem solving must involve *complex, open systems*, rather than narrowly viewed devices. For example, we are interested in computer systems diagnosis, but not electronic circuit diagnosis.

- There should be *at least partial numeric models* for solving problems, so we can relate qualitative modeling techniques to traditional engineering methodology. A problem area that makes use of traditional numeric simulation packages would be highly desirable.

- The problem area must well-enough understood, so that *system behavior can be modeled in terms of structure and function*, so there will be interesting opportunities for constructive problem solving. (In contrast, medicine rarely fits this criterion.)

- Nevertheless, there should be a number of interesting *diagnostic problems* in the domain which are solved by heuristic classification, so we can become familiar with the engineering problems within the framework of the existing tool.

- Subject matter in the problem area should involve some kind of *inference*

procedure which a student must learn (e.g., analogous to diagnosis and therapy in medicine, but not necessarily diagnosis), and domain facts should be sufficiently complex so that misconceptions occur. Ideally, there should be an established curriculum that would benefit from reconsideration in light of cognitive science research over the past decade.

- The problem domain should be *inherently interesting to laymen*, so that other researchers and contract supporters are naturally interested in learning about the project. For example, automating medical diagnosis fit this criteria. There must be widespread agreement that the problem is exciting, innovative, and important to society.

Opportunities at Stanford for such collaborative research exist in several departments, including Civil Engineering, SIMA: The Stanford Institute for Manufacturing Automation (within Mechanical Engineering), and STARS: The Space Telescope and Radar Science group (predominantly Electrical Engineering).

6. Conclusion

In the *knowledge-based tutoring* paradigm, we build a teaching program on a simulation program that can carry out the tasks we present to a student. Thus, a medical diagnosis program serves as a basis for evaluating and assisting a student as he diagnoses a patient. As we extend our research, we move beyond the problem-solving model to include other aspects of student activity in using the teaching program. For example, we might consider what goals would drive a student using the GUIDON-WATCH knowledge-base browsing program. In considering a student trying to understand the NEOMYCIN model, we have focused more narrowly on the issue of what a good diagnosis is and how deficiencies in domain knowledge can be detected by analyzing a partial solution. To simulate this analysis and test out the model, we are developing a knowledge acquisition program, GUIDON-DEBUG. In an important sense, this program integrates the analysis at every step in solving a diagnostic problem (what improvements to the model of the specific patient are required?) with the analysis of explaining why a final diagnosis is not better (what improvements to the general model of medical disorders are required?). The next step is to integrate this view of failure-driven learning with the explanation program that responds to a student's difficulty in understanding NEOMYCIN.

In summary, two new perspectives on the nature of knowledge engineering interact in the design of GUIDON2 instructional programs: We view knowledge engineering as a modeling methodology, and we view knowledge engineers as students, who use domain-general knowledge about representations and problem-solving procedures to critique and improve qualitative models.

References

Clancey, W. J. GUIDON. In Barr and Feigenbaum (editors), *The Handbook of Artificial Intelligence*, chapter Applications-oriented AI research: Education, pages 267-278. William Kaufmann, Inc., Los Altos, 1982.

Clancey, W. J. *Acquiring, representing, and evaluating a competence model of diagnosis.* HPP Memo 84-2, Stanford University, February 1984. (To appear in M. Chi, R. Glaser, and M. Farr (Eds.), *The Nature of Expertise*, in preparation.).

Clancey, W. J. Heuristic Classification. *Artificial Intelligence*, December 1985, *27*, 289-350.

Clancey, W. J. Representing control knowledge as abstract tasks and metarules. (To appear in *Computer Expert Systems*, eds. M. J. Coombs and L. Bolc, Springer-Verlag, in preparation).

Clancey, W.J. *Viewing knowledge bases as qualitative models.* KSL Report 86-27, Stanford University, 1986.

Clancey, W.J. Qualitative student models. In Traub, J.F. (editor), *Annual Review of Computer Science*, pages 381-450. Annual Reviews, Inc., Palo Alto, 1986.

Clancey, W.J. *From Guidon to Neomycin and Heracles in twenty short lessons (ONR Final Report 1979-1985).* KSL Tech Rep 86-11, Stanford University, February 1986. To appear in the AI Magazine, August 1986.

Clancey, W. J. *Knowledge-Based Tutoring: The Guidon Program.* Cambridge, MA: MIT Press 1987.

Clancey, W. J. Intelligent tutoring systems: A tutorial survey. In van Lamsweerde (editor), *International Professorship Series: 1985*, . Academic Press, London, 1987.

Clancey, W.J. Review of Winograd and Flores's "Understanding Computers and Cognition". *Journal of Artificial Intelligence*, February 1987, *31(2)*, 232-250.

Clancey, W. J. The knowledge engineer as student: Metacognitive bases for asking good questions. In A. Lesgold and H. Mandl (editors), *Learning Issues for Intelligent Tutoring Systems*, . Springer-Verlag, New York, 1987.

DeJong, G. and Mooney, R. Explanation-based learning: An alternative view. *Machine Learning*, 1986, *1(2)*, 145-176.

Hasling, D. W., Clancey, W. J., Rennels, G. R. Strategic explanations in consultation. *The International Journal of Man-Machine Studies*, 1984, *20(1)*, 3-19. Republished in Development in Expert Systems, ed. M. J. Coombs, Academic Press, London.

Hayes-Roth, B. *A layered environment for reasoning about actions.* HPP Memo 86-38, Stanford University, November 1986.

Mitchell, T.M., Mahadevan, S., Steinberg, L.I. *LEAP: A learning apprentice for VLSI design*, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 573-580, Los Angeles, August, 1985.

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based generalization: A unifying view. *Machine Learning*, 1986, *1(1)*, 47-80.

Richer, M.H. and Clancey, W.J. GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. *IEEE Computer Graphics and Applications*, November 1985, *5(11)*, 51-64.

Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. G. *Representation and use of explicit justifications for knowledge base refinement*, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 673-680, 1985.

Thompson, T. and Clancey, W. J. A qualitative modeling shell for process diagnosis. *IEEE Software*, March 1986, *3(2)*, 6-15.

Wilkins, D.C., Clancey, W.J., and Buchanan, B.G. An overview of the Odysseus learning apprentice. In T.M. Mitchell, J.G. Carbonell, and R.S. Michalski (editors), *Machine Learning: a Guide to Current Research*, . Academic Press, New York, 1986.