TOPO: IMPLICATIONS OF THE SYSTEM-MODEL-OPERATOR METAPHOR FOR KNOWLEDGE ACQUISITION

William J. Clancey and Monique Barbanson

In *Second Generation Expert Systems*, J. M. David, J. P. Krivine and R. Simmons, Eds. New York: Springer Verlag. 467-476, 1993. Also appeared in *IEEE Expert*, 6(5) 61-65, 1991

This is the original technical report. Please refer to printed copy before quoting for publication.

The systems-model-operator perspective provides a good basis for guiding the knowledge acquisition process. An experiment is presented in which we use this metaphor to develop TOPO, an expert system for configuration of computer networks. Generalizing TOPO, we show that its modeling language and operators can be adapted to other tasks which require relating a physical-organizational structure to a service-supply network. The experiment demonstrates how expert systems can be generalized and more easily related to each other if we express control knowledge in terms of operators for constructing system models.

1. Task-specific architectures

The technique for building expert systems has advanced from tools that provide an "empty knowledge base" (e.g., EMYCIN [vanMelle, 1980]) to tools that presuppose a particular task such as diagnosis or design (Chandrasekaran, 1984; Clancey; 1985; Hayes-Roth, et al., 1988; Marcus, 1988; Musen, 1989). Task-specific tools incorporate a way of organizing knowledge and an inference procedure for applying this knowledge. As researchers collect these tools and attempt to integrate them, we need to understand the relation between specific problems and general ways in which knowledge can be organized and applied. In essence, how should we formalize the part of the knowledge base that gets reused so its capabilities and limits can be related to new problems?

Generalizing from our experience and analysis of related task-specific tools, I believe that task-specific architectures should address the following distinctions (Clancey, in press):

❏ What *system* in the world is being modeled?

❏ For what purpose is the system being modeled (a *task*)?

❏ What subsystems and *subprocesses* are represented in the general model (the knowledge base)?

❏ What subsystems and subprocesses are represented in a *situation-specific model* (a problem solution)?

❏ What *relational networks* are used to represent processes (kinds of hierarchies and transitional graphs)?

❏ What are the *operators* (inference procedures) for constructing situation-specific models?

❏ How are the relational networks and inference procedure implemented in a *programming language* (e.g., frame and rule-based languages)?

The idea of describing knowledge bases in terms of tasks, systems, and models was presented in the heuristic classification analysis (Clancey, 1985). The ideas are extended here to *describe inference as a process of constructing and comparing situation-specific models of processes for some purpose*. Usually, situation-specific models are chained, so a model of one system (e.g, a diagnostic model of physiological processes) feeds into decisions for constructing another system or process (e.g., a therapy process). A key idea is that we can view a situation-specific model as a graph and inference subprocedures as operators for manipulating the nodes and relations in this graph (Clancey, in press).

In HERACLES-DX, a diagnostic shell developed from NEOMYCIN, the inference procedure is represented by *subtasks* and *metarules*. Subtasks are procedures that order and control the application of conditional statements, called metarules. Metarules are stated in a language of relations for representing structures and processes in the system being modeled (e.g., causal and subtype relations). The general model (domain knowledge) is expressed as a set of propositions over these relations. Metarules themselves use variables rather than domain terms (so we say that they are domain-general). A given set of propositions about a particular situation in the world constitutes a situation-specific model. The relations and operators—constituting a language for representing general and situation-specific models, plus an inference procedure—is the reusable knowledge contained in the HERACLES-DX shell. To summarize, according to the system-model-operator view, subtasks are *operators* for manipulating situation-specific *graphs* that represent *structures and processes* in the system being modeled.

The systems-model-operator perspective provides a good basis for developing expert systems. We can view HERACLES as the more general architecture that allows defining subtasks, metarules, or relations, but does not contain any specific operators or relations. Can we use the HERACLES architecture to efficiently write new subtasks and metarules for a different, non-diagnostic task? To illustrate this, I will present an experiment in which we develop TOPO, an expert system for configuration using the HERACLES shell. This experiment demonstrates how expert systems can be generalized and more easily related to each other if we express control or strategic knowledge in terms of operators for constructing system models.

2. TOPO: BLACKBOARDS AND OPERATORS FOR COMPUTER NETWORK LAYOUT

TOPO is an experiment in the use of HERACLES which explores the problem of writing new subtasks and metarules for logical topology design of computer networks. The problem is relevant towards the development of a front end for the programs developed by Digital Equipment Corporation for computer system configuration. An expert system like TOPO could potentially provide a sales person with a language for modeling a potential client's business and information-processing requirements. TOPO is intended to propose a layout of generic components and connections, suitable for input to sizing and configuration programs, such as XCON [63]. The program described here is a prototype that runs on one example case. Our interest here is not in definitively solving the logical topology problem, but in determining how the model-operator perspective and the availability of the subtask/metarule language helps or hinders the knowledge acquisition process.

The development of TOPO illustrates the process of using the HERACLES subtask/metarule language to define operators for a configuration task. Recall that HERACLES was originally designed with subtasks and metarules for diagnosis in NEOMYCIN. Here we show how the level of abstraction results in domain models, blackboards, and operators that might be reused in a wide variety of related applications. We also discover that the idea of a "physical-organizational model" is a common starting point for many expert systems.

2.1 Designing TOPO in terms of process models

TOPO's reasoning can be described as follows:

1. Construct a model of the **physical and organizational structure of the client's business**. Describe floor locations of workgroups at each site.

2. Determine the **information-processing requirements**, leading to a preliminary sizing (e.g., the number of printers required).

3. Design the **network topology**:
   a. derive the backbone from the physical layout.
   b. represent the components and connections.

Following the lessons learned in our heuristic classification analysis, we first conjectured that TOPO would contain three kinds of models corresponding to these three steps. However, in the final design, we found that it was more convenient to view information-processing requirements as properties of workgroups, rather than as separate objects. In effect, there are relations between elements of the physical and organizational structure (e.g., sites and buildings). Also, there are relationships between elements of the network topology (e.g., segments and backbones). But we chose not to represent relations between information-processing requirements, and therefore do not require a separate model for them.

Conceptually, we view the reasoning of the program in terms of constructing a situation-specific model (SSM) for each connected system or process (e.g., Business-Site-1, Network-at-Site-2), illustrated by Figure 1. In effect, the program first constructs a model of the client's business and then produces a corresponding network design.

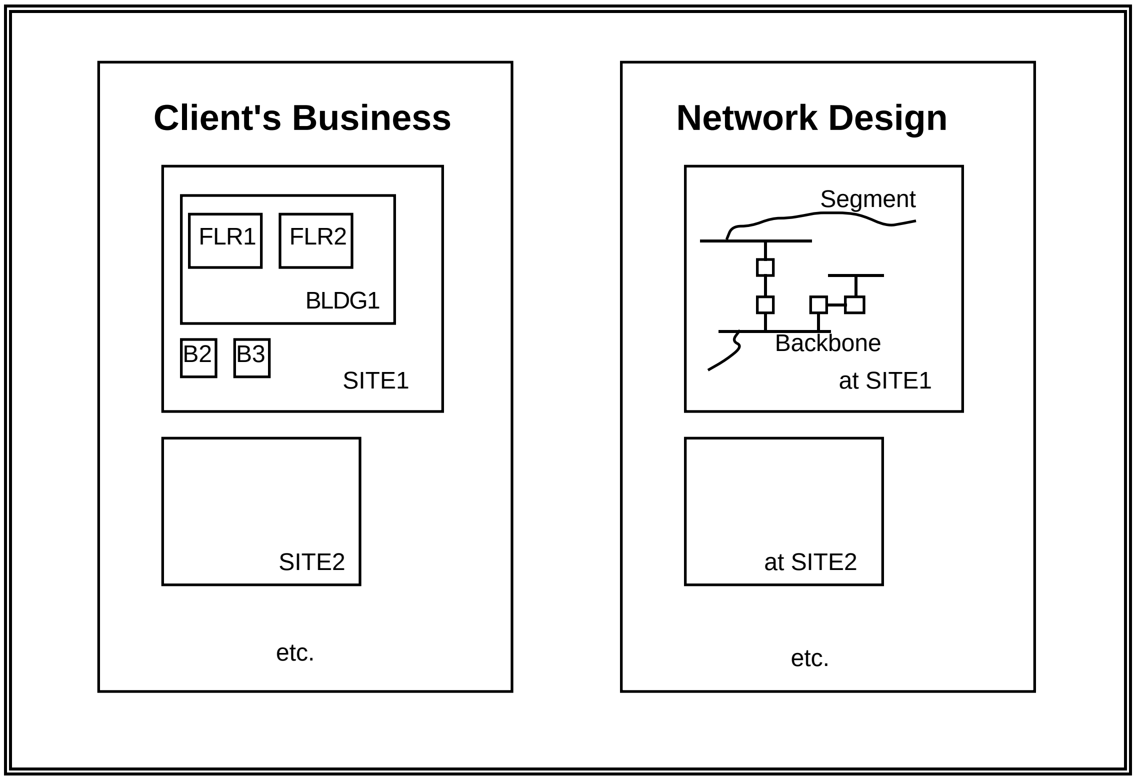


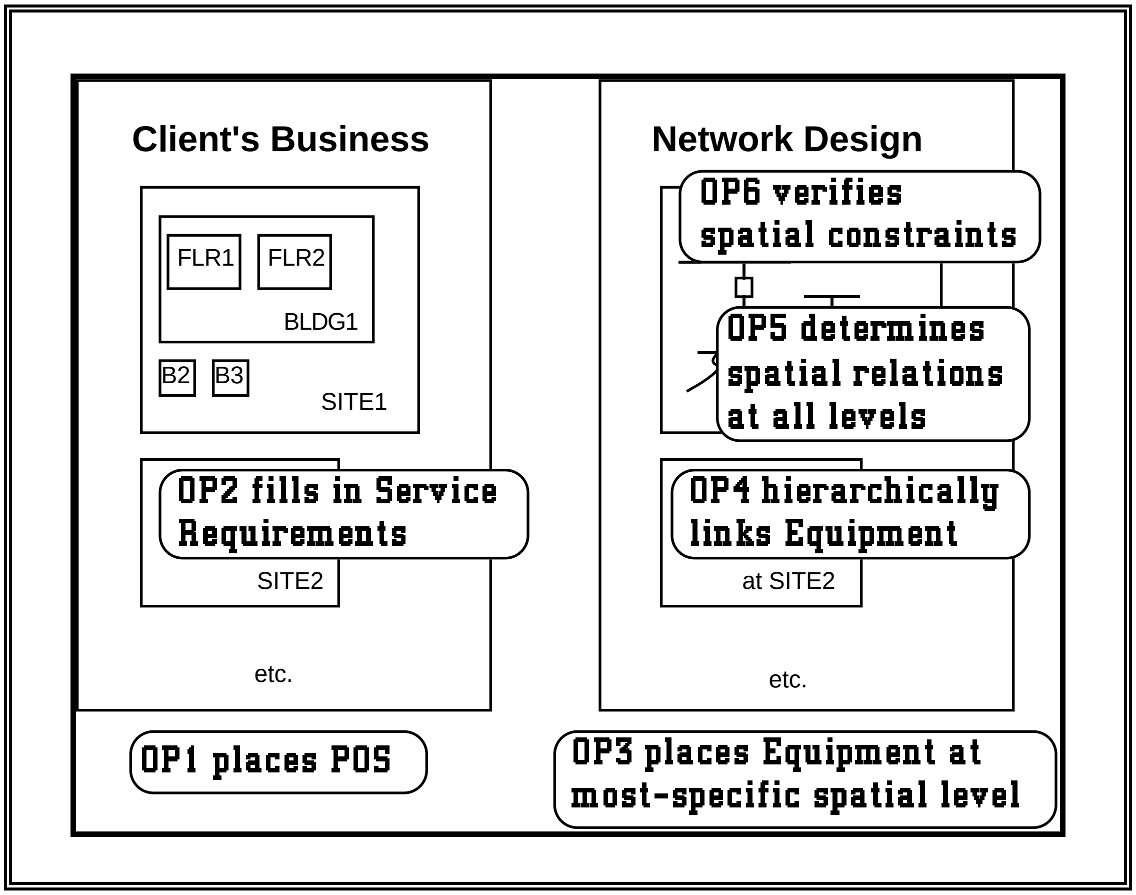
Figure 1. Two blackboards in TOPO (client business and network design) with a panel for each site; levels of each panel correspond to hierarchical connections (e.g., segments on a backbone).

2.2 Defining operators for constructing models

The following is a generalization of how to define inference operators, based on our experience in developing TOPO:

1. Draw separate pictures for each system in the world being modeled (e.g., business and computer service networks).
2. Specify class structure of nodes in each SSM (e.g., buildings, segments) and draw separate SSMs for unconnected systems (e.g., different sites).
3. Draw links showing information mapping between SSMs of different types (e.g., from workgroup to equipment layout).
4. Describe operators for placing nodes in each SSM, linking them, and specifying their spatial and process attributes.

In practice, our descriptions of TOPO's operators (Figure 2) are abstracted from the subtasks/metarules that carry out the necessary computations. A given operator typically corresponds to a single HERACLES subtask (and hence several metarules). Higher-level subtasks control when the operators are performed (just as PROCESS-HYPOTHESIS in NEOMYCIN determines that the operator TEST-HYPOTHESIS is done before REFINE-HYPOTHESIS).



Client's Business
FLR1
FLR2
BLDG1
B2
B3
SITE1
OP2 fills in Service Requirements
SITE2
etc.
Network Design
OP6 verifies spatial constraints
OP5 determines spatial relations at all levels
OP4 hierarchically links Equipment
at SITE2
etc.
OP1 places POS
OP3 places Equipment at most-specific spatial level

Figure 2. The six operators of TOPO's inference procedure place different nodes and links in the situation-specific models. The sizing table (quantifying the "amount of equipment") used by OP3 is computed by another program.

We can describe TOPO's operators in more detail in terms of the nodes and relations they manipulate on the blackboards (Table 1). In effect, the relations of the SSM are specializations of the general domain model. For example, the general domain model indicates that there can be a **USES/CONSUMES** relation between types of organizational-structures and types of services. A client-business SSM indicates that a particular organizational structure uses or consumes a particular service (e.g., "WORKGROUP-3 **USES/CONSUMES** information-storage").

Operators are implemented as HERACLES subtasks; therefore the relations of Table 1 appear in metarule premises. The SSM "nodes" or terms of the relations appear as variables in the metarules.

OPERATOR	**(RELATION Node1 Node2)**

1) "Determine Position"	(**LOCATED-IN/AT** POS <place>)
2) "Determine Service Requirements"	(**USES/CONSUMES** Organizational-Structure Service)
3) "Process Sizing Data"	(**USED-BY** Organizational-Structure Equipment) (**PROVIDES** Equipment Service) (**AMOUNT-SUPPLY** Service Sizing-data-table) (**SIZING.DATA.TABLE** Equipment #)
4) "Derive Logical Topology from POS and Equipment "	(**PROVIDE-SERVICE-FOR** Physical-Structure Logical-Service-Components)
5) "Transfer properties from POS to Service"	(**PROVIDE-SERVICE-FOR** Physical-Structure Logical-Service-Components) (**LINEAR-SPAN** Physical-Structure #)
6) "Verify Skeletal Design"	<Various relations represented in constraint rules>

Table 1. Domain relations used by TOPO's operators.
(# = a number; POS = Physical-Organizational-Structure such as buildings or workgroups; Sizing = capacity of a service or supply)

2.3 Generalizing TOPO's models and operators

To recapitulate, building TOPO in HERACLES was an incremental process in which we shifted attention back and forth between the details of writing metarules and the high-level definition of blackboards and operators. Just as we were able to build CASTER very quickly using NEOMYCIN's relations and operators, we now conjecture that expert systems similar to TOPO can be constructed more quickly that TOPO itself. Towards this end, we want to describe TOPO more generally, so we can talk about its reasoning and representations at a level more general than computer network layout.

Generalizing, we can describe TOPO's blackboards and operators as being suitable for "social services network configuration." This characterizes a wide variety of potential expert systems that map between a model of social structures (e.g., a university) to a network of services or suppliers (Figure 3). In each of these programs, we will follow similar steps: determine the Physical-Organizational-Structure; determine service and local resource sizing requirements, and finally place and connect servers and suppliers. Our claim is that the relations and operators of Table 1 provide a useful level of abstraction for building such expert systems. Put another way, we don't simply give the knowledge engineer the subtasks and metarules used in TOPO; we give the design of the blackboards in terms of the domain relations of Table 1 and a description of what the operators do.

Notice also that we are building up a language for describing kinds of *systems*. We distinguish between a POS and a service network. We observe that the service network involves a distribution of resources, not a single roving supplier. For example, TOPO's blackboards and operators are probably inadequate for solving the traveling salesman problem. We observe that other expert systems that provide "service protection" also start by constructing a POS model. For example, an expert system to configure an earthquake insurance policy for a univeristy would develop a model of the university's physical and organizational structure (e.g., a description of buildings and distribution of workgroups). Notice that such a POS model will contain new distinctions (relations) that are useful for configuring service protection, but not useful for configuring a service network. This suggests that an ultimate library of system models and operators will be roughly hierarchical, with sharing and specialization of descriptions for different tasks.

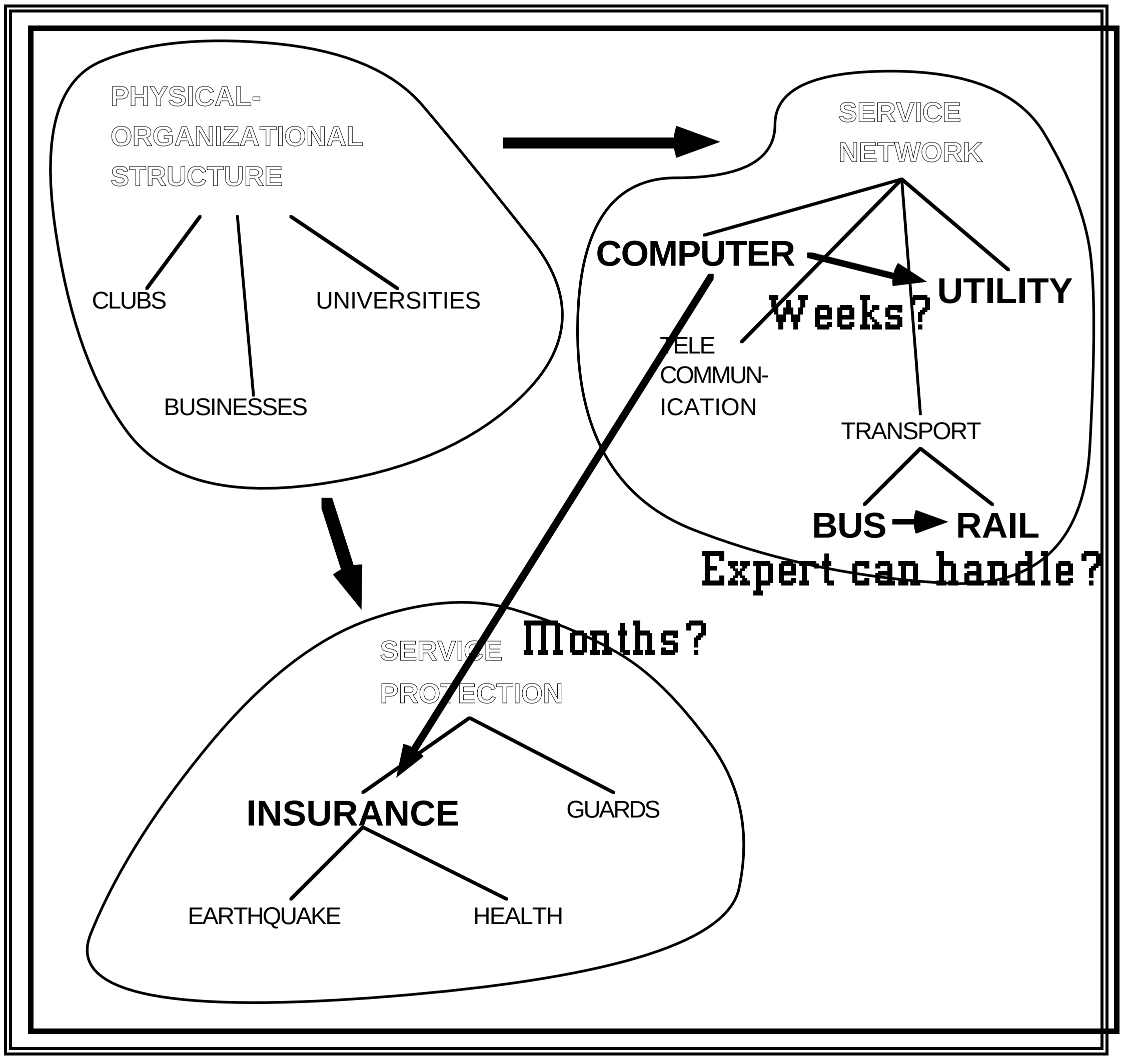


Figure 3. Generalization of TOPO's inference structure: a physical-organizational-structure (POS, e.g., a business description) is related to a service network (e.g., telecommunication network). Existing POS models could be reused in expert systems that design a different kind of system or process (e.g., insurance policy configuration).

Figure 3 shows how a classification of system types can be used to describe how specific expert systems are related, and hence the knowledge engineering effort required (in boldface) to generalize and specialize existing knowledge bases. As always, we use the term "system" very generally here; an earthquake insurance policy is a kind of system. We conjecture that it takes much more time to shift between systems that are of different abstract types (e.g., shifting from service networks to service protection). But it should be much easier to reuse blackboards and operators for systems that are of a similar type (e.g., converting operators developed for bus route configuration to rail configuration). The point of this diagram is to illustrate the research task of developing future knowledge acquisition tools. It shows just a small part of the space of potential expert systems that we might develop, but illustrates the kinds of conversions and adaptations that we expect to find throughout our enterprise. The blackboard/operator metaphor suggests that classifying our tools in terms of models of systems (e.g., telecommunication) will facilitate sharing and reuse of representations and inference procedures.

Other researchers have advocated the analysis and decomposition of knowledge bases in terms of general or "generic" components. How do these analyses compare? We can summarize the relation between the inference operators described here, Chandrasekaran's generic tasks, and McDermott's role-limiting methods as follows:

❏ A *(role-limiting) problem-solving method* is a procedure composed of several *inference operators* that--through their controlled interaction—form a situation-specific model that satisfies task constraints. For example, NEOMYCIN's subtasks together implement a variant of MOLE's COVER-AND-DIFFERENTIATE method.

❏ Furthermore, a subtree in NEOMYCIN's subtask hierarchy (e.g., EXPLORE-AND-REFINE), which appears in multiple problem-solving methods, is packaged and distributed separately in what Chandrasekaran calls a *Generic Task*.

In conclusion, the model-construction operator perspective reveals that different researchers have pursued different levels of generality in formalizing control knowledge: single SSM operators correspond to a HERACLES-DX subtask; subprocedures of one or more operators correspond to a Generic Task; a complete inference procedure corresponds to a role-limiting method.

3. CONCLUSIONS

A central claim of this paper is that it is productive to view knowledge engineering as a modeling methodology. Systems are modeled qualitatively in terms of causal, temporal, and spatial relations. From this perspective, control knowledge consists of the procedures for constructing situation-specific models. Different representations, problem-solving architectures, knowledge acquisition tools, and specific expert systems can then be systematically related by this model-construction perspective, in terms of types of relational networks, process models, inference operators, system-domains, and modeling purposes (tasks).

The HERACLES representation reveals that each time we write a new procedure for interpreting a representation, we define new relations that classify its constructs. For example, we find that classifications and procedures are defined in terms of each other. The representation that results from stating control knowledge abstractly, using variables in place of primitive terms, is not domain-independent, but *domain-general*, in the sense that the language or relations can be made more general than any one system being modeled by using spatial, temporal, causal, and subtype distinctions (a perspective by which all systems can be described).

The system-model-operator view of knowledge engineering is very general. It helps us develop new programs, like TOPO, but its strength is especially in helping us to relate different research terminology. We can relate blackboards to metarules, NEOMYCIN to MOLE, heuristic classification to qualitative process simulation, and Generic Tasks to Role-limiting methods. Put another way, we can now relate representational constructs, expert systems in a given domain, inference methods, knowledge acquisition programs, and reasoning strategies, all from a system-model-operator perspective.

Given any expert system, we can ask, "What are the systems being modeled? What are the structure and process characteristics of this system? What kind of relational network is used to represent these structures and processes? What is the inference procedure for constructing a situation-specific model? How is this model employed by later reasoning phases, evaluated, or conveyed to the user?" Rather than simply asking about a new problem domain, "Is there real-world knowledge that allows classification?" we might ask, "Must the system be modeled as open in its interactions with its environment? Is there a known etiological hierarchy? Are there stages or developmental descriptions involving trends and frequency of behaviors? What experience have people had with this system in rebuilding, modifying, assembling it in different situations?" Thus, knowledge engineering is a form of systems analysis that emphasizes qualitative modeling of processes.

ACKNOWLEDGEMENTS

TOPO was designed and implemented with Monique Barbanson at the Institute for Research on Learning and David Marques at Digital Equipment Corporation, Marlborough, Massachusetts. Partial support for this research was provided by gifts from the Digital Equipment Corporation and the Xerox Corporation.

REFERENCES

Chandrasekaran, B. *Expert systems: Matching techniques to tasks, AI Applications for Business*, Reitman ed., Ablex Publishing Corp, Norwood, 1984.

Clancey, W. J. Heuristic classification, *Artificial Intelligence* 27: 289-350 (1985).

Clancey, W. J. Model Construction Operators, to appear in *Artificial Intelligence*, in press.

Hayes-Roth, B., Hewett, M., Vaughan Johnson, M., and Garvey, A., *ACCORD: A framework for a class of design tasks*, KSL Technical Report, Stanford University, 1988.

Marcus, S., *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, Boston, 1988.

McDermott, J., Preliminary steps toward a taxonomy of problem-solving methods, In *Automating Knowledge Acquisition for Expert Systems*, S. Marcus ed., Kluwer Academic Publishers, Boston, 1988.

Musen, M., Automated support for building and extending expert models, *Machine Learning*, **4**:(3/4) 347-377 (1989).

van Melle, W., *A domain-independent system that aids in constructing knowledge-based consultation programs*, Ph.D Dissertation, Computer Science Department, Stanford University, 1980.