The Knowledge Engineering Review, No. 4: 3, 1989, 205–233

What is a deep expert system? An analysis of the architectural requirements of second-generation expert systems

E. T. KERAVNOU AND J. WASHBROOK

Department of Computer Science, University College, London, Gower Street, London WC1E 6BT, UK

Abstract

First-generation expert systems have significant limitations, often attributed to their not being sufficiently *deep*. However, a generally accepted answer to "What is a deep expert system?" is still to be given. To answer this question one needs to answer "Why do first-generation systems exhibit the limitations they do?" thus identifying what is missing from first-generation systems and therefore setting the design objectives for second-generation (i.e. deep) systems. Several second-generation architectures have been proposed; inherent in each of these architectures is a definition of deepness. Some of the proposed architectures have been designed with the objective of alleviating a subset, rather than the whole set, of the first-generation limitations. Such approaches are prone to local, non-robust solutions.

In this paper we analyze the limitations (under the categories: human-computer interaction, problem-solving flexibility, and extensibility) of the first-generation expert systems thus setting design goals for second-generation systems. On the basis of this analysis proposed second-generation architectures are reviewed and compared. The paper concludes by presenting requirements for a generic second-generation architecture.

1 Analysis of first-generation limitations

Although first-generation expert systems have reached high levels of performance, in the sense of deriving correct recommendations, they have many limitations. In some fields (e.g. medicine) these limitations are deemed so significant that the acceptability of expert systems by their potential users has been seriously undermined. The limitations, discrepancies between human and artificial expertise, can be divided into the following classes: human-computer interaction, flexibility, and extensibility. For example, human experts are often consultants who enter into dialogues with their clients both to ascertain facts and to convince their clients of the validity of their recommendations; human experts can deal with exceptional cases, not only with the routinely occurring cases, and are in a position to refer clients to other experts when the client's case is outside their own expertise; and human experts continually upgrade (revise) their expertise, both through their own experiences and by following the literature on developments in their field and reports on their colleagues' cases.

Second-generation expert systems must therefore provide adequate human-computer interaction, exhibit flexibility in executing their tasks, and support evolutionary development. The limitations of first-generation expert systems have been attributed to their being shallow. Improvements should therefore accrue by making them *deeper*. Presently there is no general agreement as to what a deep expert system is (Price and Lee, 1988). A necessary condition for accepting a definition for a deep expert system is that it is based on a sound analysis as to why first-generation systems exhibit the limitations they do.

Designing a second-generation architecture with the objective of alleviating a specific first-generation limitation (without viewing this limitation in the context of the other limitations) is prone

to generating local, non-robust solutions (see section 2). An analysis of the entire set of limitations would reveal which limitations are in fact side effects of the same cause. A viable second-generation architecture should be designed from the perspective of the ultimate causes and not their effects; in this way the architecture will provide a global and thus effective solution.

Table 1 First generation limitations

Problem-solving flexibility
• Monolithic, rigidly-applied reasoning: – Inability to dynamically plan its reasoning strategy for a specific case, based on the characteristics of the case. — Orthogonal strategies not supported. • Performance degrades dramatically when dealing with difficult (rare) cases. • Inability to recognize that a problem case is at the periphery or outside of its area of expertise.
Human–computer interaction
• Inadequate user interface: – Information required to be entered in very specific terminologies and formats. – Historic information on a case not maintained. • Inadequate dialogue structure: — System raises incoherent or redundant questions. — User not allowed to volunteer information of focusing guidance. — User not allowed to revoke an answer, or to pursue the effects of an alternative answer (to see "what if..."). • Inadequate explanation structure: — "Explanations" are just rule playbacks, and not meaningful. — Explanations are not user-tailored and do not cover all the explanation needs of the user.
Extensibility (maintainability)
• Difficult to modify the system knowledge, both manually and automatically; consistency checks not facilitated. • Inability of the system to evolve on the basis of its experiences in problem solving.

Below we analyse the first-generation limitations, summarized in Table 1.

Problem-solving flexibility

One meaning of flexible is the ability to adapt to new situations. A flexible problem-solver is one that is able to depart from standard or routine procedures; such diversions are driven by case-specific data. Problem-solving flexibility therefore derives from the ability to apply more than one reasoning method and to switch from one method to another as the particular problem case dictates. A problem-solver that applies the same method on every problem case is not flexible. First-generation systems employ a single monolithic reasoning method, usually that for dealing with routinely occurring problem cases. Attempting to apply such a method on exceptional problem cases will undoubtedly lead to drastic performance degradations. Alternatively trying to solve routine problems by applying the methods for exceptional problems is neither efficient nor effective. One uniform reasoning method and its corresponding knowledge structures is not sufficient. The coexistence of different reasoning methods requires the integration of multiple representations for the domain knowledge and the ability to move from one representation to another. (Such a system exhibits orthogonal or multi-model reasoning.) Instances of routine problems can be dealt with in one way and instances of exceptional problems in another; only then is the knowledge put into *effective* use.

Expertise is often characterized by the ability of the expert to progress a solution in the face of incomplete or missing information. Experts are good at knowing what question to ask next, reasoning with incomplete information, and revising their strategy when answers to previously unanswered questions become known. An expert's reasoning is therefore non-monotonic; new strategic choices are made as the case-specific information grows. A deep expert system must be capable of such non-monotonic reasoning at the strategic level if it is to exhibit the flexibility of a human expert.

A deep expert system must be able to plan its reasoning strategy for a specific case and modify the plan when new case-specific information violates the rationale underlying the application of a particular plan step.

In summary, flexibility is the ability to determine at every choice point during a problem-solving activity the "best" strategy for the particular case, and to change strategy when the currently pursued one ceases to be the best, thus backing out of an unpromising route. Flexibility depends on the problem-solver's grasp of the domain's reasoning knowledge—by this we mean not only a grasp of the totality of the strategies available, but also a grasp of the logical bases underlying the application of these. It is through this knowledge that a decision can be taken as to which is the best strategy in a particular context, and whether a chosen strategy is no longer the best.

A flexible problem-solver therefore identifies and reasons from case characteristics. The division of cases into routine and exceptional, used above, may be naturally extended to include peripheral cases, as peripheral cases may be mistaken for particularly exceptional cases. First-generation systems are incapable of recognizing cases which are at the periphery or outside their expertise. The need to be able to effectively recognize such peripheral cases arises because originally expert systems were considered for very narrow domains, which rendered knowledge overlaps with other closely related domains unavoidable. Thus there was a danger that the user could inadvertently use the wrong system. Consider two patient management systems, one for breast cancer and the other for lung cancer. Suppose that the former system is monitoring a patient with breast cancer who is responding successfully to radiotherapy. Unfortunately at some stage the patient develops lung cancer as well. Although lung cancer is not in the area of expertise of this system it is of paramount importance that the system recognizes, as soon as possible, the existence of a related cancer so that the patient will be managed by the other system as well. Therapy plans will need to be mediated between the two systems.

In the above example the penalty for failing to recognize a peripheral problem could be very serious indeed. In general, failing to recognize peripheral cases can seriously undermine the credibility of the system since in the eyes of its users it cannot see beyond its own, very narrow, domain. This can form a strong argument against building rather specialized, stand-alone, systems. In order to ensure that a system will never be faced with a peripheral case, its domain must span all the (narrow) areas which are closely related. (Such a solution would be more appealing if we are dealing with clusters rather than chains of areas.) Although the credibility of such a compound system will not be undermined on account of failing to recognize peripheral cases (since it will never be faced with peripheral cases), if the system embodies a distributed, cooperating architecture (cf. Mdx, section 2), the individual specialists must still possess peripheral knowledge linking their area of expertise to other specialists in the system. In the sequel the term "peripheral knowledge" is used to refer to that knowledge which enables the recognition of problems which are at the periphery or outside a particular area.

Finally, a problem solver is further enhanced by the support of a truth maintenance system. This is discussed later.

Human-computer interaction

The human-computer interaction limitations are discussed under the headings user interface, dialogue structure, and explanation structure (see Table 1).

Inadequate user interface

The user interface of many first-generation systems is inflexible and "unforgiving" requiring the user to enter information in specific terminologies and formats, otherwise the information is ignored. Other systems use menu-driven interfaces which, whilst saving the user from having to learn these terminologies and formats and ensuring that all the elicited information is meaningful to the system, can be rather tedious and inflexible.

The reason for the lack of flexibility in the user interface is that the system has neither an explicit model for its data nor a model of the user. A data model (see later) would enable the system to comprehend the user information better. Further, it would enable the intelligent abstraction and inferencing from the information given (thus relaxing the need to adhere to a very rigid terminology), as well as guiding the elicitation of additional information in an intelligent and fruitful way. The user model would tailor the questioning for the acquisition of further information to the particular user.

The above limitation is accentuated when historic information on a case is not maintained, requiring the user to enter this anew for every consultation on the same case. First-generation systems are designed as stand-alone systems. The proper management of historic case information not only requires temporal reasoning (which is not supported in first-generation systems) but also the integration with databases holding the historic information. The integration of expert systems with database systems is beginning to emerge (Brachman and Levesque, 1986; Kerschberg, 1986; Murdoch, 1987; Murdoch and Johnson, 1987; Wiederhold, 1984).

Inadequate dialogue structure

First-generation systems can raise sequences of questions which are incoherent to the user. Although the individual questions are perfectly sensible to ask, the context in which they are asked makes them incomprehensible. Incoherent sequences of questions are often the side effects of erratic or rapid changes of focus by the system which in turn are attributable to the fact that the means used by the system to derive its solution are not akin to the means used by the human expert (Keravnou, 1985).

Incoherent questions do not necessarily undermine the credibility of the system but questions whose answer can be derived from the information already known to the system certainly do. Redundant questions occur when the system lacks knowledge of the dependencies between its data, knowledge of taxonomic relationships between its data concepts, etc. Such structures (taxonomies, inferential networks, etc.) should be included in its data model. The intelligent handling of data requires the explicit representation of data models and the integration of various strategies (some of a commonsense nature) operating on the structures in the data models (Keravnou and Johnson, 1987).

Handling user-volunteered information poses problems; Mycin (Shortliffe, 1976) is an (extreme) example of a first-generation system where the user is not even allowed to volunteer information. When the system knowledge is not structured (and reasoned with) in ways comparable to those of human experts, any volunteered information cannot be acted upon, since it cannot be indexed into the system knowledge. It can only be stored and matched against subsequent information requests by the system. For the same reason the user cannot volunteer any focusing guidance; if the system is not reasoning in the way that the expert is, any guidance will not be understood.

Finally, users cannot pursue effects of alternative answers. The system is not in a position to see the significance (if any) of a particular alteration, in the same way that an expert would, due to incompatibilities in their reasoning. For the system to do this it needs to model explicitly the conceptual dependencies between its domain objects and to dynamically register which dependencies have been instantiated during a consultation. Similarly the user cannot revoke answers since the system is not in a position to determine the effects (if any) of the revocation. Systems that do allow users to revoke answers tend to restart the consultation using the new information plus any previously supplied information that still holds (this would not be necessary if the revoked answer refers to a volunteered piece of information which the system has not made use of yet). This is a

consequence of truth maintenance not being supported in first-generation systems. A truth maintenance system is auxiliary to a problem solver and functions to register the derivation paths to some inference. These consist of input information and other inferences. Hence if some input information is revoked then inferences relying on that information can be revised.

Inadequate explanation structure

In rule-based systems explanations are play-backs of rule activations. Thus the quality of the explanations is dependent on the quality of rules. Since much knowledge is implicit in rules and their justifications are usually missing (Aikins, 1983; Clancey, 1983; Dhar and Pople, 1987; Swartout, 1983, 1985; Wallis and Shortliffe, 1982), rule playbacks cannot constitute adequate explanations. Some first-generation systems can only explain their final solutions (e.g. by summarizing the arguments for and against these) without being able to explain how these solutions were derived, i.e. run time explanations are not supported (cf. Casnet (Weiss, 1974; Weiss, Kulikowski, Amarel and Safir, 1978)). Final solution justifications are very useful to have but certainly they are not sufficient on their own. Other systems dynamically generate and display traces of their activities, displaying intermediate conclusions and their support (for some systems the trace is restricted to just intermediate conclusions). Such explanations are entirely machine-initiated and tend to be given at a rather gross level. For example the justifications of the displayed activities are omitted and often the trace does not cover completely all the reasoning that took place (cf. Internist (Miller, Pople and Myers, 1982)). The above are also true for systems which support user-prompted explanations. For explanations to be adequate the system reasoning must be comparable to human reasoning and must be explicit and properly abstracted in the system. Lastly, the explanations of first-generation systems are not tailored for the specific user. Incorporating user models in a system whose knowledge is not modelled in a comparable way is not feasible. (See (Keravnou and Washbrook, 1989) for a comparison of the explanation structures of a sample of medical expert systems.)

Extensibility (maintainability): knowledge acquisition and learning

The ability to modify the system knowledge simply by adding or changing rules has been quoted as one of the strengths of rule-based systems (Davis and King, 1977). However there is now strong evidence that unstructured rule-bases (especially large ones) do not facilitate changes (Aikins, 1983; Bottacci, 1985; Clancey, 1983; Soloway, Bachant and Jensen, 1987; van de Brug, Bachant and McDermott, 1986). Rules interact implicitly through the sharing of antecedent or consequent clauses. The clauses in rule antecedents must be evaluated in the order given. These orderings often hide reasoning strategies which would be better expressed explicitly. The functionality and justification of rules is not explicit in the system. Thus for large rule-bases it is not easy to see how the rules interplay to implement the system's reasoning. Since rules do not make the encoded knowledge perspicuous, knowledge changes are not facilitated. A knowledge change may have unforeseen and undesirable effects which are not easy to detect.

Knowledge acquisition, whether manual or automatic, is facilitated when the system reasoning and factual knowledge is explicitly represented. The meaning (functionality) of every piece of knowledge should also be explicit, possibly through higher level knowledge in the system that makes use of it.

Another significant discrepancy between human and first-generation system expertise is that the system is not able to evolve on the basis of its experiences in problem solving. Human experts are able to learn from failures. The experience acquired from failing to (completely) solve a problem is used constructively to enhance their knowledge so that the next time they are faced with another instance of the same or similar problem they can perform better than the previous instances. For example such experiences would enable a physician to recognize new disorders and to refine his models of known disorders. Similarly human experts are able to reinforce their successes positively. Learning was not addressed in first-generation systems, but is considered an important issue in

second-generation systems. Recent research in Case-Based Reasoning (CBR) is very relevant to the evolutionary aspect of expert systems (Kolodner, 1982, 1984; Koton, 1988a, 1988b; Riesbeck, 1984; Ross, 1989)

Conclusion

The limitations manifested by first-generation systems are explained by inadequacies in the systems reasoning knowledge, domain factual knowledge and solution progression (see Table 2). These are considered in turn below. Reasoning knowledge, domain factual knowledge and solution progression are intimately related components of an expert system: the reasoning knowledge operates on the domain factual knowledge using case-specific information to progress the solution, so inadequacies in one will be reflected in the others.

The *reasoning knowledge* embodies the dynamics of the system. It is the component that carries out the inferencing. There are two schools of thought regarding the nature of the reasoning in expert systems (Dowie and Elstein, 1988; Schwartz and Griffin, 1988). On the one side are those who support that the reasoning must be based on statistical methods (e.g. de Dombal, 1988; Lauritzen and Spiegelhalter, 1988), the aim being to "outperform" human experts (although it is debatable what are the ideal performance criteria for a particular domain, especially medical domains). On the other side are those who support that the reasoning must emulate that of the corresponding human experts (e.g. Elstein, Shulman and Sprafka, 1978; Feltovich et al., 1984; Fox, 1984; Johnson et al., 1981; Kassirer, Kuipers and Gorry, 1982; Swanson, Feltovich and Johnson, 1977; Whitbeck, 1981). We take the latter stance. From this premise the reasoning knowledge of first-generation systems can be criticized from the following perspectives:

- *The reasoning knowledge is (wholly or partly) not akin to expert reasoning.* For example an expert diagnostic system which reasons entirely deductively is not capturing human diagnostic reasoning accurately since abductive reasoning is a central inference process in this respect.
- *The reasoning knowledge is incomplete.* Although human experts employ a number of alternative

Table 2 Causes of first generation limitations

Reasoning knowledge
• Not akin to expert reasoning.
• Incomplete:
— Lacks orthogonal reasoning.
— Lacks peripheral reasoning.
• Not properly abstracted:
— Generic tasks and strategies implicit.
— Control structure does not reflect the task semantics.
— Domain theories missing.
Domain factual knowledge
• System knowledge structures incompatible with expert's knowledge organization.
• Knowledge implicit in system.
• Knowledge missing from system:
— Explicit Data Models missing.
— Peripheral knowledge missing.
Solution progression
• Integration with databases missing.
• Temporal/qualitative reasoning not supported.
• Truth maintenance not supported.

means (orthogonal strategies) for achieving some goal, the expert system exhibits invariance of problem-solving strategy, there is a lack of negative introspection (the system does not know what it does not know about), and a lack of knowledge about rare cases. Also the reasoning involved in recognizing and dealing with peripheral cases (if any) is missing.

- *The reasoning knowledge is not properly abstrated.* In some systems there is no abstraction at all, for example in rule-based systems reasoning strategies are implicit and hence no abstraction is possible. In other systems (cf. Centaur (Aikens, 1983)) the specific instantiations (in particular contexts) of a generic strategy are explicit in the system but not the generic strategy itself; i.e. the same basic strategy is used in a number of places in the system, but no attempt is made to generalize it, e.g. as a procedure. In some systems (cf, Casnet (Weiss, 1974; Weiss et al., 1978), Internist-I (Miller, Pople and Myers, 1982) the reasoning *is* expressed generically, in terms of parameterized procedures in the particular implementation language (e.g. Fortran, Lisp, etc.). Parameterization is the essence of genericity, but procedures in a programming language do not render knowledge perspicuous; much more preferable are declarative symbolic representations, which can be used to make the knowledge more explicit (Bylander and Mittal, 1986; Fox, 1989; Fox et al., 1988). Obviously, if the generic tasks and strategies are implicit then their semantics cannot be used by the control structure of the system (this is elaborated further in section 3), and neither can first order theories (if any) be explicated in the system since these are built on top of generic tasks.

  In van Harmelen's (1989) classification of meta-level architectures (a meta-level expert system being one where control knowledge exists as an identifiable component in the system), pure meta-level architectures properly abstract the control knowledge which drives the system—it is explicit, extensible, and can form a recognizable part of the explanations which are given. The Socrates framework conforms to this architecture (Corlett et al., 1989).

The domain factual knowledge embodies the domain facts ("measles is a disorder", "measles cause spots", "spots is a symptom") but not the case facts ("the patient has spots"). It is thus the static component of the expert system. As is to be expected the inadequacies of the factual knowledge correspond closely with the inadequacies of the reasoning knowledge. These are:

- *The factual knowledge structure is not comparable to the knowledge organization used by the human experts.* The structure underlying the facts mentioned above could be: disorders cause symptoms; disorders have subtypes (i.e. taxonomic structure). Domain structure is expressed in terms of entity classes (e.g. disorders, symptoms) instead of actual entities (e.g. measles, spots). If there are mismatches between the factual knowledge structure and the expert's knowledge organization then domain knowledge is implicit in the system; e.g. taxonomic information may be implicit in the ordering of antecedent clauses in rules, and data dependencies (world facts) in screening clauses (Johnson, 1985).
- *The factual knowledge is incomplete.* Often "common-sense" background knowledge is necessary to solve a problem efficiently or credibly. For example a system which could not deduce that an infant is not an alcoholic, that a person who underwent a serious operation was administered drugs (anaesthetic being a type of drug), that a person with a temperature of 39°C has a fever, that if a class of findings is eliminated any instance of this class is also eliminated, would not be a very credible consultant. This kind of knowledge is best embodied explicitly in a *Data Model*, rather than implicitly in screening clauses of rules or whatever.

  Data Models are missing from all first-generation systems. Informally, a Data Model (Keravnou and Johnson, 1987; Keravnou et al., 1989) is a model of the input data to the system. It includes both user-volunteered and system-elicited data. More specifically, from the representational point of view, it:

— denotes the set of valid input sentences (which may include temporal aspects)
— defines abstraction, restriction, and inferential relationships between (groups of) such sentences

and from the inferrential point of view, it:

— enables the decidability of whether a sentence (or its negation) follows from a group of sentences
— enables data-abstration—a very important function of a Data Model is to convert low-level numeric data into qualitative generalizations, e.g. white-blood-count <2500 means leukopenia.

Another source of background knowledge is peripheral knowledge (see above), which is also missing from first-generation systems.

The *solution progression* is the register of the development of the solution. Most first-generation systems are able to justify their final solutions (by explaining why these are superior to their alternatives). However they are not able to present partial solutions in an equally meaningful way. What is required is for the progression towards the solution to be understandable as much as the ultimate solution. Again the inadequacies of the reasoning and/or factual knowledge are mainly responsible for the inadequacies of the solution progression. If the reasoning is not akin to human reasoning the solution path (progression) will be different from that projected by the expert, even if the final solutions correspond. If the reasoning is implicit (e.g. not properly abstracted) the system is unable to explain, in acceptable or understandable terms, a particular aspect of the solution progression. However, the solution progression may be hindered because auxiliary support is missing in the system. This support is summarized below:

- *The lack of database support.* This requires the user to enter historic case information anew with the unfortunate consequence that important cues might be missed out or might be inaccurately specified (e.g. the temporal aspect of a historic finding might not be properly updated). In addition this does not facilitate the progression of a solution over a number of temporally distinct sessions with the system. Since in most domains, case information is temporally qualified, what is required is the integration with a well-supported temporal database (Dean and McDermott, 1987).
- *Temporal and qualitative reasoning* (Allen, 1983; Allen, 1984; Kowalski and Sergot, 1986; Kuipers, 1984; Kuipers, 1986; Kuipers, 1987; Shoham, 1987; Williams, 1986) *are not supported.* First-generation systems do not represent time (e.g. about how objects evolve in time) or support reasoning about time (e.g. what values an object may be expected to show at a particular time). Temporal reasoning has a common-sense aspect in that it is not "expert"; it pervades much of human everyday reasoning, and in this respect should be modelled as an auxiliary function in a system. At a high level expertise about how to reason in a particular domain will be supported by these lower levels. Similarly qualitative reasoning, in its simplest form as data abstraction, is largely common-sense. If temporal reasoning and qualitative reasoning are not supported, then at best the system may appear incompetent, and at worst vital information given by the user may be missed by the system with the result that the solution progression is incompatible with the user's conceptions or expectations.
- Lastly, *the user should be able to revoke answers.* This could be for a number of reasons, e.g. information is by nature uncertain or unreliable, or the user simply wants to see the effect on the solution progression by two alternative values for some parameter. A Truth Maintenance System (TMS) (de Kleer, 1986; Doyle, 1979; Inoue, 1988; Smith and Kelleher, 1988) is subservient to the problem solver. The problem solver specifies the dependencies between the elements of its solution space and the TMS instantiates dependencies within the solution progression so that the effects of some revocation can be propagated in the solution progression.

Thus from the premise that an expert system must emulate human expertise, there are two dimensions to knowledge: knowledge type *completeness*, and *explicitness* within a knowledge type, neither of which are satisfied in first-generation expert systems.

2 Second-generation architectures

The terms "shallow" and "deep" are attributed to Hart (1982). Michie (1982) used the terms "high-road" and "low-road". These remarks sparked much research in the early eighties, aimed at

alleviating the limitations of first-generation systems, which is still going strong. The early systems exhibited flat knowledge representations and syntactic control, i.e. the types of knowledge and their semantics were not properly differentiated and explicated. Hence it is not surprising that by and large the work on second-generation architectures focuses on enriching the semantics of the representation and control. The order of presentation below is not significant. The architectures presented were developed (some are still being developed) in parallel, with much less mutual influence than might seem apparent.

The architectures are first reviewed in turn, and then analyzed within the framework set in section 1.

Davis' first principles architecture

Davis (1983a, 1983b, 1983c) advocates that a deep expert system must be able to reason from first principles. His thesis is demonstrated in the field of electronic troubleshooting, where he models a device both structurally (physical organization into chips, boards, etc.) and functionally (interactions between functional modules, e.g. adders, connections, etc.), at multiple levels of abstraction. The function of each generic functional module is described in terms of two classes of rules: simulation rules relating its inputs to its output, and inference rules relating output to inputs. The fault model is expressed in terms of a set of assumptions of decreasing restrictiveness, e.g. "single non-intermittent fault", "multiple non-intermittent faults", "intermittent faults", "sensor faults", etc. The topmost assumption holds until it is shown that it fails to cover the occurring fault. Dropping the topmost assumption relaxes the fault model and this may be accompanied with changes to the device model. The basic device model is that of the device from the perspective of the most constrained fault model. More relaxed fault models provide extensions to this basic device model. The ability to dynamically modify the device model is equivalent to having many predefined orthogonal perspectives of the basic device model and switching from one perspective to a more relaxed perspective as necessary. This is possibly more significant than the ability to reason from structure and function.

Davis argues that expert systems which can reason from first principles should exhibit the flexibility of a human expert. To be able to apply first principles, for electronic circuits, the device is modelled in a way that in fact does not reflect the particular task (of troubleshooting) for which the model was designed in the first place. Thus the same device model could equally be used in the context of a verification task. Task independence is not in the true spirit of an expert system, since it precludes task-specific knowledge, the use of which would, in general, give rise to a more effective solution derivation than the use of task independent knowledge would alone. The architecture therefore precludes the use of experimental knowledge (empirical associations) as an alternative strategy.

Clancey's Neomycin and heurstic classification architectures

First we discuss the architecture of Neomycin (Clancey and Letsinger, 1981), a specific medical diagnostic expert system, and then we discuss the generic architecture underlying the heuristic classification method (Clancey, 1985), which resulted from the Neomycin work.

Neomycin has an explicit taxonomy of etiologies. The ultimate etiologies corresponding to leaf nodes on this taxonomy constitute the solutions, i.e. a diagnostic problem is solved if all the manifestations are covered by instantiated ultimate etiologies. The etiological taxonomy enables the application of a top-to-bottom refinement strategy; this strategy is often instantiated not at the root node but at some other node well into the taxonomy. Such nodes are instantiated via *triggers* (associations between cheap findings and classes of etiologies). The refinement of an etiology is also based on empirical associations which make suggestions about possible refinements. The etiological taxonomy provides one route to the ultimate etiologies. Another route is provided via a causal network linking observations (findings) through intermediate pathophysiological states to etiologies; the relationship modelled here is *caused-by*. The etiological taxonomy offers the primary

reasoning method and the causal network provides a fall-back procedure. When the etiological taxonomy is applicable it should home onto the solution more quickly. But even if the causal network is not used directly for solving a problem it still serves a useful purpose by enabling the generation of more detailed explanations about why the particular diagnosis was made, through the attribution of findings to etiologies.

The provision of alternative diagnostic strategies enhances the flexibility of the system; if one approach fails the other is tried. This is not unique to Neomycin (e.g. (Fink, Lusth and Duran, 1985; Pople, 1982; Sticklen, 1987; Torasso and Console, 1989; Van de Velde, 1986) demonstrate this as well). The real novelty of the architecture, at the time, was the way the domain reasoning knowledge was modelled. This was represented declaratively as a taxonomy of tasks. The refinement of a task is controlled by (meta) rules associating preconditions to the invocation of its subtasks. The particular way in which the meta-rules for a task are interpreted constitutes the control strategy for the task. Declarative representations for the reasoning knowledge have since been used in other second-generation systems (see for example OSM (this section)).

Heuristic classification is an extension to the top-to-bottom refinement strategy exhibited in Neomycin. In heuristic classification there are two taxonomies: a data taxonomy and a solution taxonomy. The essence of the method is the *heuristic-link* linking a node in the data taxonomy to a node in the solution taxonomy. This is the generalization of the trigger link. Very briefly the dynamics of the method are as follows. Case specific observations instantiate nodes in the data taxonomy (usually low nodes). These nodes are suitably abstracted by going up the taxonomy until data nodes involved in heuristic links are reached. The instantiation of heuristic links results in the instantiation of nodes on the solution taxonomy (usually high nodes). From each of these nodes a refinement process commences until terminal nodes on the solution taxonomy are instantiated.

This method emphasizes the handling of the given information in an intelligent way by deriving suitable abstractions from it, and is claimed to have resulted in enhanced human-computer interaction.

Heuristic classification, therefore, embodies two reasoning methods; data abstraction and solution refinement. The two methods are not "competitors" but rather "collaborators" (Neomycin's methods are both "competitors" and "collaborators"; they collaborate when the one method takes over from the other at a node where the latter can not proceed further; such nodes belong to both the etiological taxonomy and causal network). Heuristic classification was only exhibited in a rudimentary way in Neomycin, but has been implemented more fully in the shell Heracles (Clancey, 1986).

Patil's multi-levels of abstraction architecture

In this section we overview the architecture of another specific medical expert system, Abel (Patil, 1981; Patil, Szolovits and Schwartz, 1981; Patil, Szolovits and Schwartz, 1982b). The essence of the architecture is the coexistence of detailed pathophysiological knowledge with less detailed phenomenological knowledge and the ability to reason at both levels. The pathophysiological knowledge provides for a more accurate attribution of findings and the phenomenological knowledge provides a more global view of the search space. Both types of knowledge are modelled through causal networks embodying "causes" links. A causal link is interpreted as a relation between attributes of the cause and effect as well as contextual information. This is a rather novel and rich interpretation. For example the causal association between diarrhea and metabolic acidosis is expressed by the following rule:

IF DIARRHEA WITH
SEVERITY = "severe"
DURATION = "more than 2 days"

```
THEN maybe   METABOLIC ACIDOSIS
             WITH
             ANION-GAP: = "normal"
             SEVERITY:  = "mild" IF RECENT-THERAPY = "bicarb therapy"
                        : = "moderately-severe" IF RECENT-THERAPY
                          = "none"
```

The architecture represents the same knowledge at multiple levels of abstraction, i.e. supports parallel perspectives of the same knowledge in increasing detail. Davis' architecture also supports parallel perspectives; the device model for a particular fault model is represented at different levels of abstraction. The reasoning method applied (in both Davis' and Patil's architectures) is the same at each level of abstraction. Neomycin models orthogonal perspectives of the same knowledge with corresponding orthogonal reasoning methods. Davis' architecture provides orthogonal perspectives but far less evidently.

Chandrasekaran's generic tasks architecture

Chandrasekeran's generic task architecture (Chandrasekaran and Mittal, 1983) is presented in the background of the Mdx/Patrec system (Chandrasekaran, 1986), yet another medical expert system, which gave rise to this architecture.

We start by outlining the thesis (Chandrasekaran and Mittal, 1982) underlying the work on Mdx/Patrec: At one end there is experiential (shallow) knowledge, represented as a database of patterns. At the other end there is deep knowledge which enables the application of first principles. The former is task dependent whilst the latter is task independent. In between the two extremes there is another type of knowledge. This is the knowledge (and associated problem structure) which has been *compiled* by the expert from the deep knowledge from the perspective of particular tasks. Thus this corresponds to knowledge which is directly applicable to those tasks and hence is very useful knowledge. Once the compilation is complete, the compiled knowledge can solve all the problems, in the context of the particular task, that are solvable by the deep knowledge and more importantly it can do so far more efficiently (task dependent knowledge is more useful in problem solving). This is the central theme of Chandrasekaran and Mittal's thesis, which in fact differentiates compiled from shallow knowledge. (Steels, see below, is similarly motivated.) Shallow knowledge can only deal with instances of routinely occurring problems. Thus if a problem case matches with a pattern in the shallow knowledge, its solution will be derived very quickly. However shallow knowledge alone is of little use when no match can be obtained. Compiled knowledge and shallow knowledge are indistinguishable in that they may both be expressed in the same form, e.g. as rules, and used in exactly the same way when making inferences. The difference between them lies in the way they are derived. Compiled knowledge is the summary result of a chain of reasoning based on a deep model; this chain can be reproduced if needed as an explanation or justification of the result. The compilation of knowledge may take place during a consultation; the system, having followed through a chain of reasoning, captures the result for future use. Eventually, when the system has been used on many cases, we may expect that it will have compiled sufficient knowledge to deal with routinely occurring cases quickly and efficiently from its store of compiled knowledge. It is not clear yet how successful this turns out in practice; Van de Velde (1986) reports some progress. Shallow knowledge, on the other hand, is purely heuristic, "rule of thumb" knowledge which has no justification other than that it is useful and "works", and is very limited in the support it gives for explanation. (Clancey has used the term compiled knowledge as well. However he does not make the above claim about the coverage of compiled knowledge. In this respect Clancey's use of the term is nearer to Chandrasekaran and Mittal's use of the term shallow knowledge.) Thus the strong implication of Chandrasekaran and Mittal's thesis is that the deep knowledge (or first principles knowledge) can not enhance the flexibility of the problem solver; however it can be used to generate more detailed explanation, e.g. in a medical diagnostic system it can provide more detailed

explanation of how the derived diagnosis covers for the manifestations. (Chandrasekaran and his team are currently working towards an automatic, incremental, compilation of deep knowledge (Chandrasekaran, Smith and Sticklen, 1989); this constitutes a form of learning.) In the Mdx/Patrec system the knowledge compiled from the deep knowledge is encapsulated in a taxonomy of concepts and the distribution of the reasoning across this taxonomy.

The compound system Mdx/Patrec exhibits the generic method of heuristic classification. Patrec performs data abstraction and Mdx performs solution refinement. More specifically, Patrec (Mittal and Chandrasekaran, 1980; Mittal, Chandrasekaran and Sticklen, 1984) is a sophisticated manager of case-specific information drawing intelligent inferences on that information and assisting Mdx in its diagnostic task. Both systems exhibit the same architecture. This architecture is in the spirit of object-oriented frameworks. The essential structure is a taxonomy of domain concepts or objects (medical data concepts for Patrec and diagnostic concepts for Mdx). The reasoning methods specific to a concept are attached to the concept in the form of procedures or rules. The reasoning is therefore distributed among the domain concepts, which are now treated as dynamic entities and referred to as *specialists*. In order to achieve its task an invoked specialist may need to involve other specialists (cooperating specialists).

Work on Mdx/Patrec has given rise to the Generic Task Methodology (Chandrasekaran, 1986; Chandrasekaran, 1987; Chandrasekaran, 1988), and associated architecture. A generic task is characterized by: the domain knowledge structures it uses; its input and output information types; and its control regime. It is implemented via a tool which encodes the problem-solving methods and knowledge appropriate for the task (Chandrasekaran, 1987). Mdx and Patrec are respectively instantiations of the generic tasks "hierarchical classification" and "knowledge-directed information passing", whose corresponding tools are CSRL (Conceptual Structures Representation Language (Bylander and Mittal, 1986) and IDABLE (Intelligent DAta Base LanguageE) (Sticklen, 1983). The methodology advocates that the building blocks for second-generation systems should be generic tasks as opposed to say backward and forward chaining which are predominant building blocks for first-generation systems. The objective is to "explicitly indicate the real control structure of the system at the task level". Generic tasks are conceptual entities and hence the methodology, in line with other methodologies for second-generation systems (Keravnou and Johnson, 1986; Breuker and Wielinga, 1985; Breuker and Wielinga, 1987a; Breuker and Wielinga, 1987b; Schreiber et al., 1988), focuses on the knowledge, rather than representation, level; "the representation of knowledge should clearly follow its use". A compound generic task is characterized by its component generic tasks and the paths and conditions for information transfer between these.

A task-specific approach to problem solving facilitates the job of the knowledge engineer (Chandrasekaran, 1989). Assuming that a problem domain is correctly associated with a generic task, then the problem-solving method(s) and related knowledge structures implementing the task constitute the focus for knowledge acquisition. (This argument is also used by the KADS methodology (Wielinge and Breuker, 1986) with its interpretation models.) Other generic tasks associated with the selected task will guide the knowledge acquisition further. Another benefit of having an explicit representation of tasks is the generation of explanations at the task level (i.e. which task was used to reach a particular decision) in addition to lower-level explanations (i.e. which data contributed towards a decision). Both these types of explanation are needed by the expert when analyzing an erroneous decision so as to attribute the cause of the error either to a task or a piece of factual knowledge. (Detailed justification of pieces of knowledge generated for example through a functional model can further enlighten an erroneous situation.) Such explanations form the basis of corrective-learning or knowledge acquisition in situ (Chandrasekaran, 1989). This is learning in its broadest sense, through human-machine synergy.

Reconstructing generic tasks within SOAR

A Generic Task (GT) architecture is a shell incorporating the inference engine for the generic task together with primitives for acquiring the knowledge required to instantiate the engine for a

particular application. The inference engine encodes, in procedural terms, a single problem-solving method for achieving the generic goal defining the given task. However, serious problems have been encountered using GT architectures. An inference engine incorporating a single problem-solving method is inflexible. If the assumptions underlying the application of the method are not satisfied by a problem then a dead-end is reached. Further, a procedural representation does not render the reasoning of the inference engine perspicuous and hence extending it, e.g. by incorporating an alternative method, is hindered. Another drawback is that although in theory it was considered easy to integrate a number of GT architectures in order to construct a problem-solver for a complex generic goal, in practice it has proven otherwise. These problems have motivated a new approach, through SOAR.

SOAR (Laird, Newell and Rosenbloom, 1987) provides a very general framework for problem solving based on the traditional state-space paradigm. SOAR, however, extends this paradigm in significant ways, firstly by offering a choice of state-spaces and secondly by resorting to using weak methods when problem-specific information does not give rise to a firm choice. Given an initial goal, a state-space and an initial state within this space are chosen. Operators are gradually selected and applied to the current state until the goal state is reached. The attractive feature of the SOAR framework is its very high problem-solving flexibility, a direct consequence of offering alternatives at the various stages of problem solving (choosing a state-space, selecting a state) and using problem-specific information to make the preferred choice from amongst the alternatives. The modular, declarative representation of alternatives in terms of productions facilitates their addition.

Representing a GT architecture within the SOAR framework is straightforward (Johnson, Smith and Chandrasekaran, 1989). Essentially it means representing the problem-solving method (inference engine) declaratively in terms of productions associating operators (for achieving sub-goals) with the conditions governing their selection in particular contexts, and extending these by adding "weak" operators which are selected when the choice of other stronger operators is not possible (these could deal with unanticipated situations). Making the reasoning explicit facilitates the addition of alternative operators (for the same sub-goal), and makes the explanation of the selected method possible.

Steels' model-based architecture

A system which is capable of learning through its own experiences has significant and welcome implications for the task of knowledge acquisition, traditionally considered the bottleneck in building expert systems. This was a main motivation behind Steels original proposal for the components of a second-generation expert system (Steels, 1986). He also cites graceful performance-degradation and richer, more convincing explanations as additional advantages accruing from his proposal. This is that in addition to a heuristic component consisting of a set of heuristic rules, a second-generation system must have a more explicit and detailed model of its domain, e.g. a causal or funtional model; this is the deep component of the system. The coverage of the rule set will normally be restricted to the commonly occurring problems. The deep model has complete coverage, but solving problems through it requires a "blind" search of a much bigger search space, which in some domains may not be feasible. When the rule set fails to solve a problem the deep model is used. The effort in doing this is not wasted as the chain of reasoning is used to extract a new rule and/or refine existing rules for future use. Thus in theory the heuristic component of the system can be empty initially and incrementally constructed from the deep model, directly using the system's own experiences. This would significantly simplify and give a new dimension to knowledge acquisition, especially if the deep model could be derived from publishing material. Van de Velde reports on using such heuristics in (Van de Velde, 1986). This work is analogous to that of Chandresekaran et al. (1989), see above.

More recently Steels has proposed a model-based theory of expertise for second-generation systems (Steels, 1987; Steels, 1988). According to the theory the reasoning knowledge is represented

as a taxonomy of tasks, the *task structure*. The *control structure* is a partial order on the task structure indicating the order in which tasks will be executed.

Tasks are generic and their decompositions into their generic subtasks form *strategies*. Terminal (primitive) tasks are also generic but their execution depends on domain specific knowledge. A primitive generic task is instantiated for a particular domain by associating it with the various domain procedures for achieving it; these procedures are referred to as *problem-solving methods*. These methods operate on a *domain model* which "represents facts about the domain without bias towards their usage". The connection between a method and the domain model is made through *heuristic role annotations*, which can be thought of as notes to the system as to how to make practical use of the domain model when executing various taks. The heuristic role annotations are analogous to the *meta-classes* in the KADS methodology (Wielinga and Breuker, 1986).

The Oxford System of Medicine architecture

The Oxford System of Medicine (OSM) is being developed at the Imperial Cancer Research Fund, London (Glowinski, O'Neil and Fox, 1989; O'Neil, Glowinski and Fox, 1989). The system design is based on the following principles:

- Several different tasks must be supported—the user interface must be uniform for all tasks.
- Flexible decision making should be promoted—reasoning and explanation strategies must be explicit and the user must be able to dynamically change these during problem-solving.
- The system should be open and expandable—the reasoning should be presented declaratively, so that it can be used in explanations and the user should be able to examine intermediate results and volunteer guidance.

The OSM models the expertise and function of a General Practitioner (GP). A GP performs a number of different tasks such as diagnosis, investigation, screening, treatment planning, prescribing and referral (switching freely between these in a non-deterministic fashion), on a number of different domains such as internal medicine, gynaecology and paediatrics. The ability to support different tasks on different domains and to provide a uniform user interface is unique to the OSM. (This follows from the work on a symbolic decision procedure (Fox, 1989; Fox et al., 1988).) Traditionally an expert system performed a single task on a single domain. The novel architecture of the OSM bears application outside the medical field—it constitutes a viable basis for a generic, knowledge-based, decision-support system.

The tasks of diagnosis, investigation, screening, etc. are all instances of the generic decision-making process (Fox, 1989; Fox et al., 1988): candidates (diagnoses, investigations, treatments) are *proposed* and *evaluated* based on *argumentation*; these candidates are *related* into associates and alternatives, and then the alternatives are *ordered*. The generic decision-making methods, propose, argue, evaluate, relate and order are explicitly represented in the system as *meta-theories*. (It should be noted that these mdethods are more generic than Chandrasekaran's generic tasks since they are independent of knowledge structures and hence their instantiations vary from task to task.) Generic decision-making methods are instantiated for a particular task, such as diagnosis, from task-specific factual, strategic and control knowledge. For example the diagnostic task includes the following three strategies for proposing diagnoses:

- Diagnoses may be proposed by looking at the causes of the patient's symptoms.
- Subtypes of a suggested diagnosis are also diagnoses to be considered.
- Causal consequents/antecedents of a suggested diagnosis are also diagnoses to be considered.

Strategies are expressed independently of their justifications or logical bases. For example the above strategic statements do not indicate how a strategic choice is to be made in the context of the particular case information. Such constraints on the invocation of strategies are represented separately as task-specific control knowledge, activated via generic control rules.

Both reasoning knowledge and factual knowledge are represented declaratively. Every piece of

knowledge (task-strategy, medical fact etc.), except generic-methods, is available for inspection by the user. Emphasis is given on the user being able to dynamically modify reasoning knowledge to suit his needs. There is no duplication of knowledge as every medical fact is stored once in the system and can be used by any number of tasks. Medical facts are therefore considered "use independent". In addition to the decision-making tasks, discussed above, the system supports information retrieval tasks, data management tasks, etc. and the design team raises the questions: should the knowledge structuring be optimized for the decision making tasks and how sensitive these are to the structure of the medical facts.

The architecture of the OSM, as with other second-generation architectures discussed here, aims to separate and make explicit the different types of knowledge involved. In addition it aims to represent generic methods or meta-theories explicitly and thus to support, within a uniform framework, a number of different tasks operating on a number of different domains.

Summary

Below we summarize the essence of each of the above architectures; these constitute informal statements of deepness:

Davis
- Ability to reason from first principles.
- Ability to impose and relax assumptions as necessary.

Clancey:

- Separation of reasoning knowledge from domain factual knowledge.
- Orthogonal (cooperative) reasoning methods.
- Ability to abstract from given data to home onto partial solutions.

Patil:

- Causality as the essential relationship (rich interpretation).
- Ability to manipulate multiple representations of the same knowledge at different levels of detail.

Chandrasekaran:

- Knowledge compiled for effective use (knowledge must reflect its usage).
- Control structure explicitly indicated at the task level.
- Compound tasks modelled as a community of cooperating tasks.

Steels:

- Expertise decomposed into domain models, task structures and problem-solving methods.

Oxford System of Medicine:

- Coexistence of a number of different tasks operating on a number of different domains within a uniform framework.
- Explicit representation of meta-theories, generic control strategies, task-specific strategies and control, and factual knowledge.
- Knowledge represented declaratively and non-redundantly; facts are "use independent".

Compared with first-generation architectures we observe that each of the above architectures attempts to enrich knowledge organization and reasoning. Much emphasis is placed on the control structure, and the need to make it conceptual by basing it at the task level.

Discussion

In this section those features of the proposed second-generation architectures overviewed above

which have been realized in working systems are analyzed within the framework of the first-generation limitations and their perceived causes given in section 1. This discussion is summarized in Tables 3 and 4 which respectively correspond to Tables 1 and 2. Referring to Table 3 some of the architectures have overcome more limitations than others but none of them have overcome all the cited limitations.

Table 3 Limitations overcome by architectures

	Problem–Solving Flexibility						*Human–Computer Interaction*					*Extensibility Maintainability*	
	Alternative strategies	*Dynamic strategic plan*	*Graceful perfomance degradation*	*Recognition of peripheral cases*	*Flexible user interface*	*Historic info maintained*	*Questioning coherent & non-redundant*	*User may volunteer info or guidance*	*User may revoke answers*	*Explanation improvements*	*Knowledge acquisition facilitated*	*Evolution supported (learning)*
Davis' troubleshooter	○	○	●							●		
Neomycin	●	○	●				●	●		●	●	○
Heracles					●		●	●			●	
Abel	○		●				●	●		●	●	
Mdx	○			●	●	●	●	●			●	
OSM	●	●	●					●	●	●	●	○

(*Key:* ●, satisfied; ○ the path has been paved for satisfying it)

Davis' troubleshooter

Davis troubleshooter simulates the function of a device in order to identify discrepancies between the expected behaviour and the observed behaviour of the device. This method is used for any problem case. Human trouble-shooters may resort to simulation when faced with unfamiliar problems, but will generally use experience (compiled knowledge) at least in the initial phases of fault tracing. Once the fault has been localized to a relatively small component of the device then the human may mentally simulate its operation. Hence Davis' method, generally known as "reasoning from structure and function", certainly entails deep understanding of the domain but does not fully capture human expertise.

Davis approach does not exhibit orthogonal reasoning in the sense of supporting alternative means for the same ends. However some dynamic planning within the constraints of the given method does take place through the dynamic revision of the fault model. Revisions in the fault model might cause the device model to be revised. As the device is modelled at different levels of abstraction, the system is capable of granular reasoning.

Davis' aim was to achieve high levels of performance and to overcome rapid performance degradations due to exceptional or peripheral cases. In this respect, where the application of his architecture is feasible, he has probably achieved his aim. However he did not necessarily aim to capture human problem-solving flexibility, nor to capture conversational adequacy, nor to support

extensibility. Some aspect of human problem-solving flexibility is captured though in the dynamic revision of the fault model. Also since the architecture makes explicit the structure and function of generic device components, extensibility (adding or modifying components) is facilitated.

Table 4 How were the limitations overcome

	Reasoning knowledge								*Domain factual knowledge*				*Solution progression*	
	Reasoning akin to expert reasoning	*Orthogonal reasoning*	*Peripheral reasoning*	*Explicit generic task & strategies*	*Semantic control structure*	*Explicit domain theories*	*Knowledge structure meaningful*	*No implicit knowledge in structures*	*Explicit data control*	*Peripheral knowledge*	*Database integration*	*Temporal/qualitative reasoning*	*Truth maintainance*	
Davis' troubleshooter	○	○						●				○		
Neomycin	●	●		●	●		●	●						
Heracles	●			●	●		●	●	●					
Abel	●	○					●	●						
Mdx	●		●				●	●	●	●	●	●		
OSM	●	●		●	●	●	●	●		●			●	

(*Key:* ●, achieved; ○, the path has been paved for achieving it)

Neomycin

Clancey's aim in building Neomycin was specifically to capture human diagnostic skills so that the system could form an efficient basis for teaching these skills (the tutoring system, Guidon2 (Clancey, 1979b; Clancey, 1983a) using Neomycin was significantly more successful than its predecessor system, Guidon (Clancey, 1979a) which used Mycin).

Neomycin exhibits orthogonal reasoning (ultimate etiologies are derived either by reasoning from classes of etiologies to their subtypes or by reasoning from observations to their ultimate causes (etiologies) through intermediate causes). The causal model captures the progression of the disease processes and thus provides for a more accurate attribution of the findings than the etiological taxonomy. Hence the causal model can aid the diagnosis of problems which are not captured by the etiological taxonomy thus giving rise to smoother performance degradations than would have been possible in the absence of the causal model. Neomycin's and Abel's causal models capture malfunctioning behaviour and hence although they serve a similar role to Davis' "structure and function" model they are not direct analogues. Kuiper's qualitative models essentially capture proper functioning (Kuipers, 1984; Kuipers, 1986: Kuipers, 1987) and hence they are analogous to Davis' models. However, the structure and function of electronic components is well-defined and thus much easier to capture than that of human body systems. Although Neomycin supports orthogonal reasoning it is not clear whether it is capable of more effective revisions of its current strategy than the: "try this approach and if it fails try the other approach". (This is exactly how the fault model in Davis' system is revised: if it is shown, by an exhaustive search, that the currently adopted

model fails to cover the particular fault then the fault model is revised to a less-constrained model.) It seems that this is the approach currently adopted by the designers of systems which employ orthogonal (or multi-model) reasoning with a view to investigating (and modelling) knowledge-directed plan revisions.

Neomycin not only improves problem-solving flexibility over its predecessor system Mycin but it also achieves improvements in human-computer interaction and extensibility. Neomycin's reasoning is akin to human reasoning (this was its premise) and this reasoning is abstracted in terms of tasks and meta-rules. Neomycin can give run-time strategic explanations (Clancey, 1983b; Hasling, 1983; Hasling, Clancey and Rennels, 1984). Further, its question sequences are coherent and the user can volunteer information and focusing guidance (the latter in terms of hypotheses to be added in the differential). However Neomycin does not include a Data Model, although it does captures some aspects of one. Lastly, since knowledge in Neomycin is differentiated and explicated through symbolic declarative representations, knowledge updates are facilitated. In addition Neomycin's strategic and domain knowledge is used by a learning system (Wilkins, Buchanan and Clancey, 1984) which learns by watching a human expert solve problems. For every question raised by the expert the learning system formulates hypotheses about the possible reasons behind the question. These hypotheses are based on Neomycin's reasoning capabilities. The generated hypotheses are presented to the expert to confirm that the system's reasoning is correct or to correct this reasoning. Knowledge updates are necessary when the system cannot reach a hypothesis as to why the expert raised a particular question.

Heracles

The Heracles shell has been successfully used in engineering and medical applications. Heracles employs a single method for deriving a solution namely top-to-bottom refinement. The flexibility of a particular Heracles application depends not so much on the accuracy of the heuristic links (which after all can, be definition, be fallible) but on the quality of the knowledge associated with the nodes in the solution taxonomy. If each node has the knowledge to quickly divert attention to a node on an alternative solution path when the case information indicates so, then the problem-solver is flexible and the wasted processing due to backtracking is minimal (this in a sense would also capture peripheral reasoning—see discussion on Mdx below).

However unlike Neomycin, Heracles has an explicit Data Model which supports the auxiliary reasoning method of data abstraction. The user can volunteer information (the Data Model allows for a flexible interface) and the system questioning is non-redundant. Lastly Heracles knowledge structures are explicit and thus knowledge updates are facilitated.

Abel

Like Davis' troubleshooter, Abel does not exhibit orthogonal reasoning at the global level, i.e. it does not have different global means for deriving a solution. Also like Davis' troubleshooter it applies granular reasoning. The presence of the most abstract description level enables the system to focus on the significant features of a case and thus to avoid the possibility of being bogged down in (possibly insignificant) detail (which would be likely if there were a single, rather detailed, description level). This leads to flexibility since the higher level indicates where in the lower level attention should be focused and when the focus should be switched to another part of the more detailed space. Thus in Abel control switches freely between the different levels of abstraction. (This is not so in Davis' troubleshooter, where the reasoning proceeds from top-to-bottom and going back to a higher level is considered backtracking and thus wastage in processing.) In Abel the most detailed description level enables the handling of exceptional cases, thus yielding graceful performance degradations.

Abel does exhibit orthogonal reasoning in information acquisition, information acquisition being a significant aspect of diagnostic reasoning. Abel carefully constructs and executes a plan for

information acquisition based on its current differential (Patil, Szolovits and Schwartz, 1982a). This plan is used to justify the system's information requests which are coherent and non-redundant. In general Abel tries to emulate human reasoning, e.g. its information acquisition strategies are intuitive strategies observed to be used by human experts (Elstein, Shulman and Sprafka, 1978). Thus Abel does address human-computer interaction, enabling the user to query the system's questioning (giving meaningful explanations (Swartout, 1981)), and to volunteer both information and focusing guidance. The domain factual knowledge is declarative and explicit and revisions are facilitated but the reasoning is hardwired in terms of Lisp procedures.

Mdx

None of the systems analyzed so far in this section directly addresses the need to recognize problem cases which are peripheral to their area of expertise. The original literature on Mdx (Chandrasekaran et al., 1979) does address this problem. The Mdx architecture aimed to emulate the way a community of specialists cooperate to solve a problem. The system designers based their communication model on the way real-life medical specialists refer a case to other specialists, although the real-life scene has been considerably constrained in Mdx: specialists in Mdx are taxonomically related, communication is along these hierarchical links (no lateral calls are allowed) and possibly more importantly, these specialists are extremely narrow. The knowledge of a specialist is essentially concerned with establishing itself (i.e. establishing that the case is an instance of its specialty) and in the case of non-primitive specialists with refining itself. However specialists invoked by superspecialists (as part of the latter's refinement process) may recognize that their superspecialist made a mistake in calling them and may reply back with a firm suggestion as to which of the other subspecialists in the system ought to be invoked instead. It is not clear though whether Mdx can recognize if a case is peripheral to its entire "expertise".

Mdx does not exhibit orthogonal reasoning at the global level. At the global level Mdx is seen to have a taxonomy of specialists (each with its specific knowledge) and with top-to-bottom refinement as the overall reasoning method (this method and associated knowledge structure constitutes a single generic task in the Generic Tasks Architecture). The refinement solution progresses from specialist to (sub)specialist and a particular specialist may have recourse to alternative means to achieve its refinement. Hence whilst at the global level orthogonal reasoning is not apparent, at the level of individual specialists orthogonal reasoning is possible.

Mdx has the services of Patrec. Patrec embodies an explicit Data Model about medical concepts and is integrated with a database holding patient records. Patrec ensures that the given information is handled in an intelligent way, that the user interface is more flexible than in other systems and that Mdx does not raise redundant questions. The user may volunteer information and focusing guidance, the latter in terms of specialists to be invoked.

Knowledge in Mdx is very modular and the interactions between pieces of knowledge (specialists) are constrained and explicit. Hence knowledge updates are facilitated.[1]

OSM

OSM is the only system out of these considered that gives the leading hand to the user. The system is user-driven, with the user deciding when to give information and which task (diagnosis, treatment plan, etc.) should be considered next. The user can also request the system to justify its current decisions or beliefs (diagnoses, treatments, etc.) or to display relevant knowledge (reasoning or factual) at any point during the consultation. The system is truly subservient to the user. Thus the issue of the system raising coherent, non-redundant questions is not relevant here.

[1] Rule-bases are modular but modularity on its own is not sufficient to guarantee maintainability (Bottacci, 1985; Soloway, Bachant and Jensen, 1987; Van de Brug, Bachant and McDermott, 1986)—interactions between rules are implicit which makes the management of a large rule-base unwieldy.

OSM exhibits orthogonal reasoning (e.g. there are different means for generating diagnoses, either through causality relations or through taxonomic relations, etc.) and is capable of revising its strategic plan dynamically, by switching attention to a new strategy, when incoming information satisfies the criteria controlling the activation of this strategy. OSM generally exhibits much flexibility in switching from one task to another or from one strategy to another. The multiplicity of strategies (for the same goal) and the corresponding richness in the factual knowledge structure increases the chances that if one approach fails or does not apply to some case, another approach will, resulting in a more graceful performance degradation. OSM's knowledge base spans large areas of medicine, e.g. internal medicine, paediatrics and gynaecology. However the system should address the problem of recognizing cases that need specialist attention and to suggest the appropriate referrals. At the moment peripheral reasoning is not addressed in OSM (although referral is one of its tasks).

In addition OSM does not have Data Models, it does not support temporal or qualitative reasoning and it does not integrate with databases. However it does support truth maintenance (this is provided automatically by the implementation environment) and hence the user can freely revoke information.

Lastly, every piece of knowledge, both factual and reasoning, is represented symbolically and declaratively. This enhances its explicitness and also facilitates knowledge acquisition. In theory the user is able to modify every piece of knowledge (except domain theories) thus quickly generating systems that yield different behaviours for different users. This is an aspect of evolutionary development, but not in the sense of the system learning on its own through its experiences.

3 Generic second-generation architecture: requirements

A relative definition of Deepness

By definition second-generation systems are *deeper* than first-generation systems. If therefore we knew what a deep expert system was we could design a generic architecture for second-generation systems. Similarly if we had a generic architecture for a second-generation system we could define deep expert systems. However, given the nature of expert systems, classifying expert systems into deep and not-deep may not be possible. Deepness is neither a binary nor a multi-valued (fuzzy) relation but rather a multi-variable relation (see below). This is why a relative definition of deepness, enabling a multi-dimensional comparison between systems, is more appealing. A relative definition does not have the ambitious objective of delineating second-generation systems as its extension includes every pair of systems, but it can pave the way to an absolute definition.

A relative definition of deepness can be derived empirically by a comparative analysis of a sample of existing expert systems. The sample can be entirely drawn from first-generation systems; first-generation is a rather crude qualification since systems in this broad category exhibit highly diverse architectures (the boundaries between first and second-generation systems are blurred). We have carried out such an analysis using a representative sample of medical expert systems (Keravnou and Washbrook, 1989). The aim was to analyze each of six medical expert systems from the perspectives of their reasoning knowledge, factual knowledge and solution progressions, thus specifying models of expertise for these systems. The explanation, and dialogue, potentials of these systems were also analyzed and matched against their models of expertise . The relative definition of deepness which resulted from this work is given below. (This definition is a generalization and extension of the definition proposed by Klein and Finin (1987): a model M is deeper than a model M′ if M represents knowledge or is able to infer knowledge that is implicit in M′.)

A model (of expertise) M can be deeper than a model M′ from the following perspectives:

- the *factual knowledge* in M is more explicit than the factual knowledge in M′:

— M represents knowledge types which are implicit in M′

— M gives a richer interpretation to a knowledge type than M′ does
— M represents types or aspects of knowledge that are absent in M′ even though it would be meaningful to specify such knowledge types and aspects in the context of M′

- the *reasoning knowledge* in M is more explicit than the reasoning knowledge in M′:

— M represents reasoning strategies which are implicit in M′
— M abstracts its reasoning knowledge more than M′ does
— the semantics of the reasoning knowledge in M are more explicit than the semantics of the reasoning knowledge in M′

- the *solution progression* in M, at intermediate stages, is more understandable than the solution progression in M′ (see discussion in section 1)

The above definition enables a comparison between systems from a number of deepness perspectives, e.g. from the perspective of reasoning knowledge or the perspective of factual knowledge. As such, a system may be considered deeper than another system from one perspective and vice versa from a different perspective. This is also true for the Klein and Finin definition. Deepness can certainly be viewed from many perspectives in the context of knowledge-based systems which exhibit complex organizational structures. The above definition enables a richer multi-dimensional comparison between systems.

Proposed architecture

Since the terms *generic* and *task* will figure much in the following discussion we start by clarifying the use of these terms in the present context. An object, O, is generic relative to some type, T, if it applies to every instance of T (type instances are themselves objects): T constitutes the application domain for O. O is still considered to be generic if it applies to most instances of T. Types include higher order types (e.g. type classifications, types which have internal structure, type aggregates) so that an object may be generic to a set of types. A task has ends (goals) and means for achieving its ends. The means employed by the task define the functionality of the task. In second-generation expert systems we talk about generic tasks (see below).

The architecture presented here is generic to objects of type "second-generation expert system". This is a higher order type as it includes "second-generation diagnostic systems", "second-generation design systems", etc. The architecture is based on the analysis of the first-generation limitations given in section 1 and the relative definition of deepness given above.

In second-generation systems the reasoning knowledge must be conceptual, separate, explicit, properly abstracted, with clear application contexts, and with explicit semantics. These requirements are closely interrelated. Reasoning knowledge is distinctly different from factual knowledge and these two types of knowledge must be separated and made explicit in the system; reasoning knowledge must not be implicit in the organization of factual knowledge. Reasoning knowledge must be expressed in terms of conceptual reasoning tasks and strategies and these expressions must be given at the highest possible levels of abstraction, i.e. the level of generality of some reasoning task must be made explicit. This relates to the requirement of clear application contexts. On the basis of generality of application the following classes of reasoning tasks have been identified:

- *Generic tasks.* A generic task applies to all domain entities of the relevant type. In addition the means for attaining the task (strategies) are the same in each case, i.e. the strategies of generic tasks are also generic. For example, in a diagnostic system, **Differentiate** is a generic task relative to type *differential* (a differential is a set of objects (of the same type) which constitute alternatives). The following are strategies for **Differentiate**: "if there are too many alternatives try to **Rule-out** some of these"; "if there is one alternative whose likelihood is sufficiently high and well above the likelihoods of the other alternatives then **Pursue** that alternative"; "if the alternatives can be grouped then **Group-and-Differentiate** (the set of resultant groups is also a differential)". These strategies are meaningful for any differential and hence they are themselves generic. In the example

Rule-out, **Pursue** and **Group-and-Differentiate** are subtasks of **Differentiate.** Differential is a type aggregate where the aggregates are instances of type *candidate.* **Rule-out** and **Pursue** are generic relative to type candidate, and their strategies operate on the internal structures of candidates.

- *Nearly generic tasks.* A nearly generic task is a generic task for a large subset of the domain entities of the relevant type. For the remaining domain entities the particular task is either *conceptually generic* (see below) or it does not apply at all. For example a task **Pursue** would be nearly-generic relative to type *candidate,* if candidates are generally pursued in one way except for a few candidates, each of which is persued in its specific way (for these few candidates the task is conceptually generic).
- *Conceptually generic tasks.* A conceptually generic task is one whose function (in the abstract) is applicable to all relevant domain entities, but the means for achieving it (strategies) are not the same for each domain entity. In the worst case the means for achieving the task are different for each domain entity; in this case the function of the task must be explicit in the abstract whilst the different means are distributed among the relevant domain entities. If there are very few different means then the task can be mapped into a generic task with generic subtasks corresponding to the various distinct means. For example in the OSM (see section 2), the decision-making methods propose, argue, evaluate, relate and order are conceptually generic relative to the set of objects (of type task): diagnose, investigate, treat, refer, prescribe.
- *Non-generic tasks.* A non-generic task applies to a single (or a very small number of) domain entity. Such tasks must be associated with their respective domain entities.

For a particular application one would expect that the majority of tasks will belong to the first three categories above. A task is properly abstracted in the system if its category is made explicit (e.g. the types of the factual knowledge it applied to are made explicit).

The means for achieving a task constitute the strategies for the task. A strategy may involve calling other tasks. For example a common strategy is to decompose the task into subtasks each of which needs to be achieved, possibly in a strict order. Or there may be different strategies for achieving the task, each strategy invoking a different subtask (see above example of task **Differentiate**). The semantics of a task, comprizing the conditions underlying the invocation of its strategies and the interactions between these strategies, must be explicit. The actions of the strategies are not necessarily "competitive" since the corresponding subtasks may interact, enabling the dynamic switching from one strategy to another.

Tasks may have more than one invocation context and are therefore related (via their strategies) into a network. Usually there are distinct taxonomies of tasks, the root of each taxonomy corresponding to some global task (goal) which is refined into its subtasks. (For example in Clancey's heuristic classification there are two conceptually distinct functions, data abstraction and solution refinement, corresponding to two distinct task taxonomies.) Task taxonomies are distinct in the sense that all the tasks in the taxonomy are involved in performing a certain specific global function. Separate task taxonomies (in the same system) are related. The function represented in one task taxonomy may constitute an auxiliary function for some task in another taxonomy. Such associations are lateral as opposed to the refinement associations within a task taxonomy.

Tasks which constitute leaf nodes in task taxonomies are *primitive* tasks. Primitive tasks manipulate the factual knowledge and progress the solution derivation.

The semantics of reasoning tasks explain how the tasks should be achieved in every context that they are likely to be invoked, e.g. which is the best strategy for achieving the given task in the particular context and when this might cease to be the best choice requiring a switch to an alternative strategy. If the strategies for a task cooperate the semantics of the task must explain how this cooperation is carried out. The structure of the semantics of a task form the model for that task (the model of a non-primitive task entails models for the task's strategies). If tasks conform to the same model then we have *generic task-models* (i.e. task schemata). The set of (generic) task-models in a system collectively define the *control structure* of the system. The control structure determines, in

context, the reasoning step to be performed next, and thus embodies the dynamics of the system. The control structure of second-generation systems is semantic as it is based on the semantics of tasks. Control structures in first-generation systems are largely syntactic.

The conclusion of section 1 was that from the premise that an expert system must emulate human expertise, the architecture of a second-generation expert system must satisfy the two knowledge criteria:

- knowledge type *completeness*
- *explicitness* within a knowledge type

Knowledge type completeness

From the perspective of this criterion a second-generation system, in addition to the specialist knowledge comprizing its expertise, must also possess a model for its data (encompassing background domain knowledge) as well as knowledge about other related domains (see Figure 1). Each of these classes of knowledge is divided into reasoning and factual knowledge. The problem-solving strategies are orthogonal and hence the corresponding specialist factual knowledge is multi-model. Truth maintenance and temporal reasoning support (where necessary) must be provided and the system must be integrated with a (temporal) database.

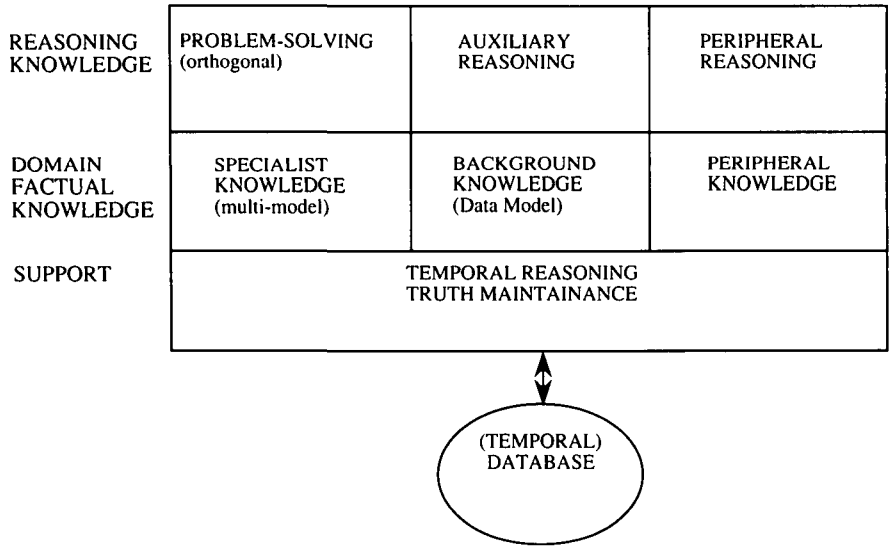

Figure 1 Generic architecture (knowledge type completeness)

Knowledge explicitness

This criterion is illustrated in Figure 2. A second-generation expert system must have a layered architecture, where the domain factual knowledge is at the bottom layer, the reasoning knowledge at the middle and the control structure at the top layer. Both the factual and reasoning knowledge have underlying structure that needs to be explicated in the system in order to make the links between the three layers explicit.

The reasoning knowledge must be expressed as much as possible in terms of generic tasks and their strategies. Generic tasks by definition apply to every factual entity of the relevant type (or class). Hence generic tasks are defined in terms of entity classes instead of individual entities, i.e. in terms of factual knowledge structure.

Similarly the control structure must be as abstract as possible by expressing it in terms of generic task-schemata. Associating each task with its specific control regime is very clumsy, especially when every task conforms to the same control schema. We illustrate what we mean by generic task-schema through an example schema used in the Neocrib diagnostic system (Johnson and Keravnou, 1986;

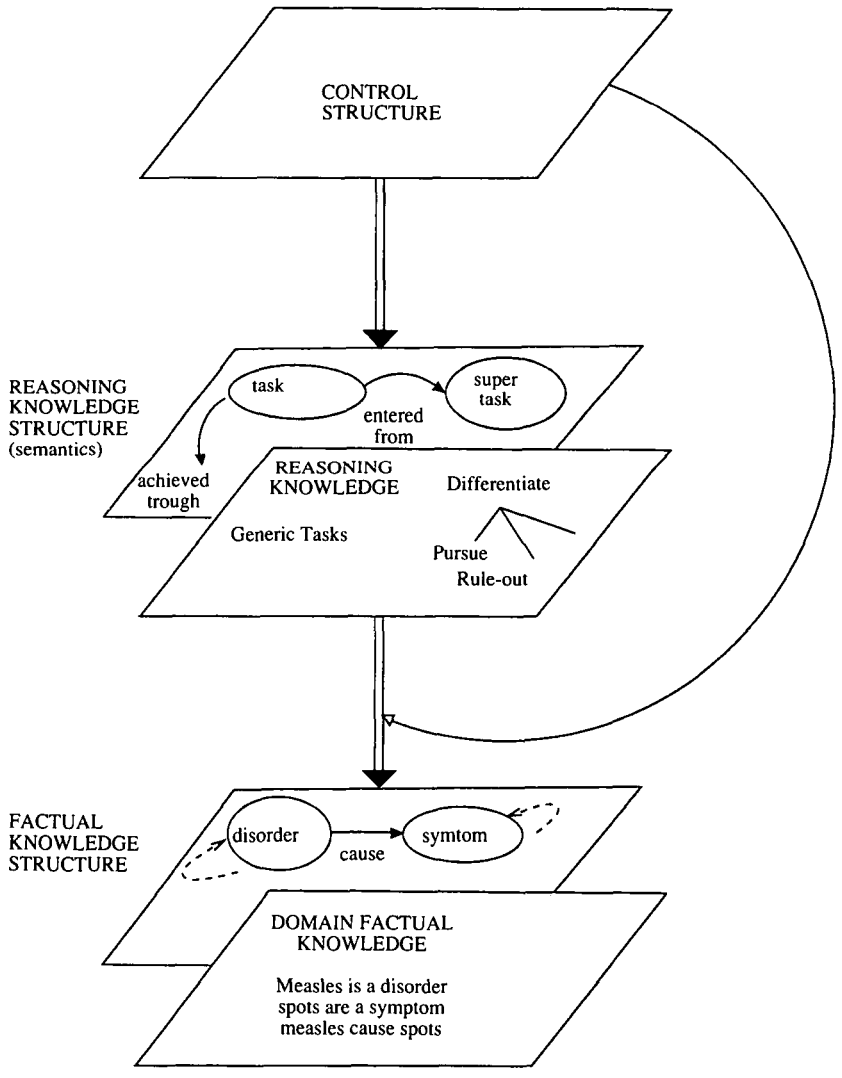

Figure 2 Generic architecture (knowledge explicitness)

Keravnou and Johnson, 1988). Referring to Figure 3 a task is decomposed into a set of subtasks. To achieve a task, instantiations of the specific subtasks are repeatedly selected and executed until a termination condition for the task is satisfied. The selection of a subtask instantiation is geared by case-specific information on which the selection conditions for the subtasks are applied. The selection conditions for a subtask form its logical basis in the context of the particular task. A logical basis for a subtask (always in the context of some task) consists of the conditions that must be true for the subtask to be selected (*enabling condition*) assuming that another set of conditions are not true (*disabling condition*). In addition disabling conditions can be violated through a set of *relaxation conditions.*

Hence the control structure instantiates and executes tasks based on their underlying semantics and thus drives the whole system by causing tasks to manipulate factual knowledge and hence progress the solution.

4 Conclusion

First-generation expert systems have serious limitations attributed to their being shallow. Second-generation systems must therefore be deep which begs the question "What is a deep expert system?". Numerous definitions for a deep expert system have been proposed such as:

- a system that reasons from domain first principles

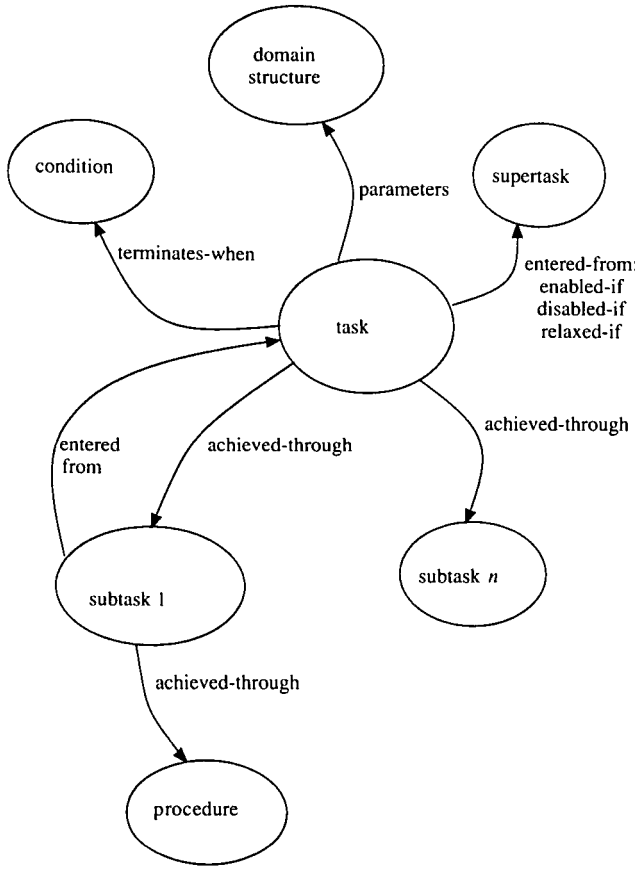

Figure 3 Example generic task-schema

- a system that reasons from structure and function
- a system that reasons qualitatively and temporally
- a system that reasons causally

Definitions like the above define goals for the architecture of second-generation systems. From the user's viewpoint, however, a second-generation system must:

- exhibit more flexibility in executing its task
- provide more adequate human–computer interaction
- support evolutionary development

The user requirements set acceptability goals. A definition for a deep expert system is therefore viable if the resulting architecture meets the user requirements. Thus viable architectural goals should be founded on a thorough analysis of the entire set of first-generation limitations. Our analysis of these limitations reveals the following as their collective causes:

- The domain reasoning knowledge is not akin to expert reasoning, it is incomplete, and it is not adequately abstracted.
- The domain factual knowledge is not structured in the way human experts model their knowledge, it may be implicit and incomplete.
- The solution progression is hindered by the fact that databases, temporal reasoning and truth maintenance are not supported.

The second-generation architectures discussed have suceeded in overcoming many, though not all, first-generation limitations and some of the multi-model approaches are very promising.

On the basis of our analysis of first-generation limitations we have set the requirements for a generic second-generation architecture from the perspectives of knowledge completeness and

knowledge explicitness. A second-generation system satisfying these requirements should alleviate the first-generation limitations and could justifiably be called "deep".

Acknowledgements

The structure of the paper has been greatly influenced by comments from John Fox, John Campbell and Janet Sinclair on an earlier draft of the paper. We would also like to thank Professor Chandrasekaran and another referee for their constructive criticism.

References

Aikins, J, 1983. "Prototypical knowledge for expert systems" *Artificial Intelligence* **20** 163–210.

Allen, JF, 1983. "Maintaining knowledge about temporal intervals" *Communications of the ACM* **26** 832–843.

Allen, JF, 1984. "Towards a general theory of action and time" *Artificial Intelligence* **23** 123–154.

Bottacci, L, 1985. *Modifiability of rule-based expert systems*, PhD Thesis, Division of Cybernetics, Brunel University, UK.

Brachman, RJ and Levesque, HJ, 1986. "What makes a knowledge base knowledgeable? A view of databases from the knowledge level" In: Kerschberg, L, ed, 1986. *Expert Database Sytems: Proc. 1st International Workshop*, pp. 69–78. London: Benjamin-Cummings.

Breuker, J and Wielinga, B, 1985. "KADS: Structural knowledge acquisition for expert systems" *Proc. 5th International Workshop in Expert Systems and their Applications*, Avignon, 887–900.

Breuker, JA and Wielinga, BJ, 1987a. "Use of models in the interpretation of verbal data" In: Kidd, AL, ed *Knowledge acquisition for expert systems: a practical handbook*, New York, NY: Plenum, pp 14–44.

Breuker, JA and Wielinga, BJ, 1987b. "Knowledge acquisition as modeling expertise: the KADS methodology" *Proceedings of the 1st European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading University.

Bylander, T and Mittal, S, 1986. "CSRL: a language for classification problem solving and uncertainty handling" *The AI Magazine* August 1986, 66–76.

Chandrasekaran, B, 1986. "Generic tasks in knowledge-based reasoning: high level building blocks for expert system design" *IEEE Expert* Fall 1986, 23–30.

Chandrasekaran, B, 1987. 'Towards a functional architecture for intelligence based on generic information processing tasks" *Proc. IJCAI87*, pp 1183–1192.

Chandrasekaran, B, 1988. "Generic tasks as building blocks for knowledge-based systems: the diagnosis and routine design examples" *The Knowledge Engineering Review* **3**(3) 183–210.

Chandrasekaran, B, 1989. "Task structures, knowledge acquisition and learning", to appear in *Int. J. of Machine Learning.*

Chandrasekaran, B and Mittal, S, 1982. "Deep versus compiled knowledge approaches to diagnostic problem solving" *Proc. AAAI-82* 349–354.

Chandrasekaran, B and Mittal, S, 1983. "Conceptual representation of medical knowledge for diagnosis by computer: MDX and related systems" *Advances in Computers* **22** 217–293.

Chandrasekeran, B, Gomez, F, Mittal, S and Smith, J, 1979. "An approach to medical diagnosis based on conceptual structures" *Proc. IJCAI-79* 134–142.

Chandrasekaran, B, Smith, JW and Sticklen, J, 1989. "Deep models and their relation to diagnosis" *Int. J. of Artificial Intelligence in Medicine* **1** 29–40.

Clancey, WJ, 1979a. "Dialogue management for rule-based tutorials" *Proc. IJCAI-79* 155–161.

Clancey, WJ, 1979b. "Tutoring rules for guiding a case method dialogue" *Int. J. Man–Machine Studies* **1** 25–49.

Clancey, WJ, 1983. "Methodology for building an intelligent tutoring system" In: Kintsch, Polson and Miller (eds), *Methods and Tactics in Cognitive Science*, Lawrence Erlbaum Publishers.

Clancey, WJ, 1983b. "The advantages of abstract control knowledge in expert system design" *Proc. AAAI-83* 74–78.

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

Clancey, WJ, 1985. "Heuristic classification" *Artificial Intelligence* **27** 289–350.

Clancey, WJ, 1986. "From Guidon to Neomycin to Heracles in twenty short lessons: ORN final 1979–1985" *The AI Magazine* August 1986, 40–60.

Clancey, WJ and Letsinger, R, 1981. "Neomycin: Reconfiguring a rule-based expert system for application to teaching" *Proc. IJCAI-81*, pp 829–836.

Corlett, R, Davies, N, Khan, R, Reichgelt, H and van Harmelen, F, 1989. "The architecture of Socrates" In: Jackson, P, Reichgelt, H and van Harmelen, F, *Logic-based knowledge representation*, Massachutts: The MIT Press, pp 37–64.

Davis, R, 1983. "Expert systems: where are we and where are we going?", *The AI Magazine* Winter 1988.

Davis, R, 1983b. "Reasoning from first principles in electronic troubleshooting" *Int. J. Man–Machine Studies* **19**, 403–423.

Davis, R, 1983c. "Diagnosis via causal reasoning: paths of interaction and the locality principle" *Proc. AAAI-83* 88–94.

Davis, R and King, J, 1977. "An overview of production systems" *Machine Intelligence* **8** 300–332.

de Dombal, FT, 1988. "Computer-aided diagnosis of acute abdominal pain: the British experience" In: Dowie, J and Elstein, A (eds.), *Professional judgement: A reader in clinical decision making* Cambridge: Cambridge University Press, pp 190–199.

de Kleer, J, 1986. "An assumption-based TMS" *Artificial Intelligence* **28** 127–224.

Dean, TL and McDermott, DV, 1987. "Temporal data base mangement" *Artificial Intelligence* **32** 1–55.

Dhar, V and Pople, HE, 1987. "Rule-based versus structure-based models for explaining and generating expert behaviour" *Communiations of the ACM* **30** 542–555.

Dowie, J. and Elstein, A, eds., 1988. *Professional Judgment: A reader in clinical decision making* Cambridge: Cambridge University Press.

Doyle, JA, 1979. "A truth maintenance system" *Artificial Intelligence* **12** 231–272.

Elstein, LD, Shulman, LA and Sprafka, SA, 1978. *Medical problem solving: An analysis of clinical reasoning* Massachusetts: Harvard University Press.

Feltovich, PJ, Johnson, PE, Moller, JH and Swanson, DB (1984). "LCS: the role and development of medical knowledge in diagnostic expertise" In Clancey, WJ and Shortliffe, EH (eds.), *Readings in Medical Artificial Intelligence: The first decade*, New York: Addision-Wesley, pp 275–319.

Fink, PK, Lusth, JC and Duran, JW, 1985. "A general expert system design for diagnostic problem solving" *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-7** 553–559.

Fox, J, 1984. "Formal and knowledge-based methods in decision technology" *Acta Psychologica* **56** 303–31; and In: Kerschberg, L, ed, 1986. *Expert Database Systems: Proc. 1st International Workshop*, pp. 226–252. London: Benjamin Cummings.

Fox, J, 1989. "Symbolic decision procedures for knowledge based systems" to appear in Adeli (ed.), *Handbook of Knowledge Engineering* New York: John Wiley.

Fox, J, Glowinski, AJ, O'Neil, M and Clark, DA, 1988. "Decision making as a logical process" *Proc. of Expert Systems '88* Cambridge: Cambridge University Press.

Glowinski, A, O'Neil, M and Fox, J, 1989. "Design of a generic information system and its application to Primary Care" *AIME 89* pp 221–233.

Hart, PE, 1982. "Direction for AI in the eighties" *SIGART Newsletter* No. 79, January 1982.

Hasling, DW, 1983. "Abstract explanations of strategy in a diagnostic consultation system" *Proc. AAAI-83* 157–161.

Hasling, DW, Clancey, WJ and Rennels, G, 1984. "Strategic explanations for a diagnostic consultation system" *Int. J. Man-Machine Studies* **20** 3–19.

Inoue, K, 1988. "Pruning search trees in assumption-based reasoning" *Proceedings of the 8th Int. Workshop on Expert Systems & their Applications* pp 133–151.

Johnson, L. 1985. "The need for competence models in the design of expert consultant systems" *Int. J. Systems Research and Information Science* **1** 23–36.

Johnson, L and Keravnou, ET, 1986. "Analysing, representing and interpreting expert strategic knowledge" *Cybernetics and Systems '86* 743–750.

Johnson, PE, Duran, AS, Hassebrock, F, Moller, J, Prietula, M, Feltovich, PJ and Swanson, DB, 1981. "Expertise and error in diagnostic reasoning" *Cognitive Science* **5**, 235–283.

Johnson, TR, Smith, Jr JW and Chandreskaran, B, 1989. "Generic tasks and SOAR" Laboratory for AI Research, Department of Computer and Information Sciences, The Ohio State University, Columbus, Ohio 43210.

Kassirer, JP, Kuipers, BJ and Gorry, GA, 1982. "Towards a theory of clinical expertise" *The American J. of Medicine* **73** 251–259.

Keravnou, ET, 1985. "Building expert systems that model competence: a case study in fault diagnosis" *PhD Thesis*, Division of Cybernetics, Brunel University, UK.

Keravnou, ET and Johnson, L, 1986. *Competent expert systems: A case study in fault diagnosis*, London: Kogan-Page.

Keravnou, ET and Johnson, L, 1987. "Intelligent handling of data by integration of commonsense reasoning" *Knowledge-Based Systems Journal* **1** 32–42.

Keravnou, ET and Johnson, L, 1988. 'Neocrib" In: Keravnou and Johnson (eds.), *Expert systems architectures* London: Kogan-Page, pp 168–182.

Keravnou, ET and Washbrook, J, 1989. "Deep and shallow models in medical expert systems" *Int. of Artificial Intelligence in Medicine* **1** 11–28.

Keravnou, ET, Washbrook, J, Dawood, RM, Hall, CM and Shaw, D, 1989. "A model-based diagnostic expert system for skeletal dysplasias", *AIME 89* pp 47–56.

Kerschberg, L, ed., 1986. *Expert Database Systems, Proc. 1st Int. Workshop*, London: Benjamin-Cummings Co.

Klein, D and Finin, T, 1987. "What's in a deep model: a characterization of knowledge depth in intelligent safety systems" *Proc. IJCAI-87* 559–562.

Kolodner, JL, 1982. "The role of experience in development of expertise" *Proc. AAAI-82* 273–277.

Kolodner, JL, 1984. "Towards an understanding of the role of experience in the evolution from novice to expert" In: Coombs MJ (ed.), *Developments in expert systems* London: Academic Press.

Koton, P, 1988a. "A medical reasoning program that improves with experience" *Proceedings of the 12th Annual Symposium on Computer Applications in Medical Care.*

Koton, P, 1988b. "Reasoning about evidence in causal explanations" *Proc. AAAI-88* 256–261, Saint Paul, Minnesota.

Kowalski, R and Sergot, M, 1986. "A logic-based calculus of events" *New Generation Computing* **4** 67–95.

Kuipers, B, 1984. "Commonsense reasoning about causality: deriving behaviour from structure" *Artificial Intelligence* **24** 169–203.

Kuipers, B, 1986. "Qualitative Simulation" *Artificial Intelligence* **29** 289–338.

Kuipers, B, 1987. "Qualitative simulation as causal explanation" *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-17** 432–444.

Laird, JE, Newell, A and Rosenbloom, PS, 1987. "SOAR: an architecture for general intelligence" *Artificial Intelligence* **33** 1–64.

Lauritzen, SL and Spiegelhalter, DJ, 1988. "Local computations with probabilities on graphical structures and their application to expert systems" *J. R. Statist. Soc.* **50**.

Mitchie, D, 1982. "High-road and low-road programs" *AI Magazine* **3** 21–22.

Miller, RA, Pople, HE and Myers, JD, 1982. "Internist-I: an experimental computer-based diagnostic consultant in general internal medicine" *New England J. of Medicine* **307** 468–476.

Mittal, S. and Chandrasekaran, B, 1980. "Conceptual representation of patient databases" *J. Med. Syst.* **4** 169–185.

Mittal, S, Chandrasekaran, B and Sticklen, J, 1984. "Patrec: a knowledge-directed database for a diagnostic expert system" *IEEE computer*, September 1984, 51–58.

Murdoch, STR, 1987. "A review of the intersection of expert systems and database systems: Expert/Database systems" *Int. J. Systems Research and Information Science* **4** 111–119.

Murdoch, STR and Johnson, L, 1987. *Intelligent data handling by active databases* London: Kogan-Page.

O'Neil, M, Glowinski, A and Fox, J, 1989. "A symbolic theory of decision-making applied to several medical tasks" *AIME 89* pp 62–71.

Patil, RS, 1981. "Causal representation of patient illness for electrolyte and acid-base diagnosis" *MIT/LCS/TR-267.*

Patil, RS, Szolovits, P and Schwartz, WB, 1981. "Causal understanding of patient illness in medical diagnosis" *Proc. IJCAI-81* 893–899.

Patin, RS, Szolovits, P and Schwartz, WB, 1982a. "Information acquisition in diagnosis" *Proc. AAAI-82* 345–348.

Patil, RS, Szolovits, P and Schwartz, WB, 1982b. "Modelling knowledge of the patient in acid-base and electrolyte disorders" In: Szolovits, P (ed.), *Artificial intelligence in medicine*, AAAS Selected Symposium Series, Boulder, Co: West View Press, pp 191–226.

Pople, HE, 1982. "Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnosis" In: Szolovits, P (ed.), *Artificial Intelligence in Medicine*, AAAS Selected Symposium Series, Boulder, CO: West View Press, pp 119–185.

Price, CJ and Lee, M, 1988. "Deep knowledge tutorial and bibliography" *Alvey Report IKBS3/26/048.*

Riesbeck, CK, 1984. "Knowledge reorganization and reasoning style" In: Coombs, MJ (ed.), *Developments in expert systems* London: Academic Press, pp 159–176.

Ross, SP, 1989. "Case-based reasoning: position paper", submitted to Workshop on Medical AI: current issues, in medical decision support, *2nd Scandinavian Conference on Artificial Intelligence*, Tampere, Finland.

Schreiber, G, Breuker, J, Bredeweg, B and Wielinga, B, 1988. "Modelling in KBS development" *Proc. 8th Int. Workshop on Expert Systems & their Applications*, pp. 283–296.

Schwartz, S and Griffin, T, 1986. *Medical Thinking: the psychology of medical judgment and decision making*, Berlin: Springer-Verlag.

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

Shortliffe, EH, 1976. *Computer-based medical consultations: Mycin*, New York: Elsevier.

Smith, B and Kelleher, G, ed., 1988. *Reason maintenance systems and their applications*, Chichester: Ellis Horwood.

Steels, L, 1986. "Second-generation expert systems" In: *Research and development in expert systems III*, Bramer MA (ed.), Cambridge: Cambridge University Press.

Steels, L., 1987. "The deepening of expert systems" *AI Memo 87–16*, Artificial Intelligence Laboratory, Vrije Universiteit Brussel.

Steels, L, 1988. "Components of expertise" *AI Memo 88–16*, Artificial Intelligence Laboratory, Vrije Universiteit Brussel.

Sticklen, J, 1983. "Manual for IDABLE", Technical Report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, Columbus.

Sticklen, J, 1987. "MDX2: An integrated diagnostic system", *PhD Dissertation.* Department of Computer & Information Science, The Ohio State University.

Soloway, E, Bachant, J and Jensen, K, 1987. "Assessing the maintainability of XCON-in-RIME: coping with the problems of a VERY large rule-base" *Proc. AAAI-87* 824–829.

Swanson, DB, Feltovich, PJ and Johnson, PE, 1977. "Psychological analysis of physician expertise: implications for design of decision support systems" *MEDINFO77* 161–164.

Swartout, WR, 1981. "Producing explanations and justifications of expert consulting programs" MIT Laboratory for Computer Science, Technical Report *MIT/LCS/TR-251.*

Swartout, WR, 1983. "XPLAIN: a system for creating and explaining expert consulting programs" *Artificial Intelligence* **21**, 285–325.

Swartout, WR, 1985. "Knowledge needed for expert systems explanation" *AFIPS Conference Proceedings* **54** 95–98.

Torasso, P and Console, L, 1989. *Diagnostic problem solving: Combining heuristic, approximate and causal reasoning*, North Oxford Academic.

van de Brug, A, Bachant, J and McDermott, J, 1986. "The taming of R1" *IEEE Expert*, Fall 1986, 33–38.

Van de Velde, W, 1986. "Learning heuristics in second-generation expert systems" *Proc. Sixth Int. Workshop on Expert Systems and their Applications*, Avignon 1986.

Van Harmelen, F, 1989. "A classification of meta-level architectures" In: *Logic-based knowledge representation*, Jackson, P, Reichgelt, H and van Harmelen, F (eds.), Massachusetts: MIT Press.

Wallis, JW, and Shortliffe, EH, 1982. "Explanatory power for medical expert systems: studies in the representation of causal relationships for clinical consultations" *Meth. Info. Med.* **21** 127–136.

Weiss, SM, 1974. "A system for model-based computer-aided diagnosis and therapy" *Ph.D Thesis*, Computers in Biomedicine, Department of Computer Science, Rutgers University, CBM-TR-27-Thesis.

Weiss, SM, Kulikowski, CA, Amarel, S and Safir, A, 1978. "A model-based method of computer-aided medical decision-making" *Artifical Intelligence* **11** 145–172.

Whitbeck, C, 1981. "What is diagnosis? a preface to the investigation of clinical reasoning" *Metamedicine* **2** 319–329.

Wiederhold, G, 1984. "Knowledge and database management" *IEEE Software* January 1984, 63–73.

Wielinga, BJ and Breuker, J, 1986. "Models of expertise" *Proc. ECAI-86* 306–318.

Wilkins, DC, Buchanan, BG and Clancey, WJ, 1984. "Inferring an expert's reasoning by watching" *Heuristic Programming Project Report No HPP-84-29*, Department of Computer Science, Stanford University.

Williams, BC, 1986. "Doing time: putting qualitative reasoning on firmer ground" *Proc. AAAI-86* 105–112.