KNOWLEDGE ACQUISITION FOR USER SUPPORT IN HETEREOGENEOUS NETWORKS

Maurice Sharp & Brian R. Gaines

Knowledge Science Institute, University of Calgary
Calgary, Alberta, CANADA T2N 1N4
sharp@cpsc.ucalgary.ca, gaines@cpsc.ucalgary.ca

Abstract: This paper analyzes knowledge representation and acquisition for an intelligent network support system in which an object-oriented knowledge base is used to model a heterogeneous network. The system is designed to be effective with a minimal knowledge base of: the processors and peripherals on the network; software packages and where they can be run; users and their normal activities; and active processes and current loading. The object structure allows this knowledge to be expanded to greater levels of detail in particular areas as required. The system is designed to support users of varying levels of skill and requirements to make more effective use of the network computing resources by advising them on software, processor and processing availability. It is also designed to support the network manager in resource planning, security violation detection and load balancing.

INTRODUCTION

We have entered an era of *system integration* where the dominant theme in information systems development is the integration of diverse technologies into unified systems. In the late 1960s suppliers of computer-based information systems found it necessary to form separate system integration divisions to cope with the complexities of hardware and software interaction and integration. The ad hoc activities of such divisions have grown to become major cost centers, and it is no longer possible to treat system integration as a fairly informal customization stage in system production. This has focused developmental effort on system integration as a major technology in its own right. In the 1980s integration is the primary developmental thrust in all areas, and at all levels, of information systems.

As usual in the development of new technologies, however, the thrust towards integrated systems, while solving old problems also creates new ones:

- First, the overall complexity of the total system after integration can be very high. This is, of course, the objective—to design, develop and implement complex systems by integrating a number of simpler ones. Success in such an approach to system design is critically dependent on decoupling the system layers—only the functionality of the lower levels must be relevant to the upper levels—consideration of their implementation and operation should be irrelevant to the system design. However, this decoupling is not easily achieved—it becomes the major, and most problematic, design objective.
- Second, the problems of designing complex systems are exacerbated if the components include not only hardware and software but also people. The "liveware" component may be designed in part by selection and training but there remains a massive residual uncertainty about its functionality and performance that conventionally requires human "management." The integration of people and the associated management processes into large-scale integrated systems presents major new problems. In particular, the management structures already in place are unlikely to be appropriate to the overall integrated system. Major changes in organizational structure and corporate culture will generally be necessary to the effective implementation and operation of large-scale integrated systems.
- Third, the classic approach to the design of complex systems is based on homogeneous system concepts—that the overall system is designed as a set of related components developed with mutual interaction in mind. Heterogeneity may be introduced in the design of such systems, but its introduction is a design decision made in the light of overall system functioning, not a result of the fairly anarchical evolution of systems not originally intended to be integrated.

Fortunately, as often happens, the advances in technology that have created these problems also bring with them the means of solution (Gaines 1988a). Knowledge-based systems, in particular, provide a technology suited to modeling the diversity and complexity of integrated heterogeneous systems. The model can encompass as much detail as required to capture the important features of the system. The operational form of the model makes it possible to simulate system behavior at various levels of detail. The high-level representation makes it possible to use the model to aid designers and advise users. There are major problems involved in keeping the model up-to-date which place important constraints on the implementation and application of such knowledge-based systems. However, these constraints are intrinsic to the operation of complex integrated heterogeneous systems, and the knowledge-based modeling approach throws light on these fundamental problems.

The project described in this paper is targeted on the development of knowledge-based network support systems in which an object-oriented knowledge base is used to model a heterogeneous system. The support system is designed to be effective with a minimal knowledge base of: the processors and peripherals on the network; software packages and where they can be run; users and their normal activities; and active processes and current loading. The object structure allows this knowledge to be expanded to greater levels of detail in particular areas as required. The system is designed to support users of varying levels of skill and requirements to make more effective use of the network computing resources by advising them on software, processor and processing availability. It is also designed to support the network manager in resource planning, security violation detection and load balancing.

The concepts underlying the intelligent network support system design are: that much can be achieved through the availability of small amount of processible knowledge *about* the network; that it is better to keep a small knowledge base correct than to provide a large knowledge base that is out of date; and that there will be a heterogeneity of knowledge requirements and the capabilities to keep the knowledge base up to date. The initial implementation is in Smalltalk on a network of some 30 workstations under NCS, including Sun, Apollo, Macintosh, Iris, and Symbolics, and access to Vax, Butterfly, Multics and Cyber 205 mainframes. These systems serve a community of research groups in vlsi, ai, graphics, simulation and networking. The overall design is targeted on wider communities without specialist computing knowledge using heterogeneous networks for purposes such as computer-integrated manufacturing, dispersed process instrumentation and control, and collaborative research.

INTEGRATED HETEROGENEOUS SYSTEMS

Figure 1 shows the information systems in a typical integrated heterogeneous system, such as a banking system using ATMs or a modern production unit using computer integrated manufacturing techniques. The outer ring comprises eight basic information technologies that together support the organization and its interface to the physical and organizational worlds. The center shows the organization itself with its corporate structure, and the rings surrounding it show the human interfaces and communication protocols necessary for the systems' management and integrated operation. The figure has been simplified to the minimum structure necessary to support an effective integrated system, and the residual complexity shows the need to consider the total context of the information technology involved. Such a system will not necessarily succeed if any one component, or even several, are advanced state-of-the-art technologies. However, it will probably fail if one component is inadequate or the total system is not well-integrated.

The Organization: The central core is the people that manage and operate the system, their organization and value systems. The overt operation of the system is part of the *technical culture* of the organization in Hall's (1959) terms. It can be communicated, documented and partially programmed. It can also be changed fairly readily in response to new requirements and technologies. The management structure of the organization is part of its *formal culture*. Only part of it is overt and documented and it is resistant to change. What has come to be called the "corporate culture" (Deal & Kennedy 1982) is in Hall's terms the *informal culture* of the organization. It pervades the organization, is communicated by example, undocumented, and changes only slowly or under pressures of survival. The managerial structures and social processes of the organization are not normally treated as part of its information technology. However, to be successful the technical systems must be consistent with the organizational structures and processes. This applies particularly to inter-personal communications where computer networks will not be used if they violate the managerial and social norms of the organization, and to command and control paths which have to conform to existing authority and procedures that may not be those of the organization chart.

Person-Computer Interface: Recognition that the managerial structure of the organization dominates its style of operation draws attention to person-computer interaction as a critical technology. Advances in graphics, voice input/output and intelligent terminals have improved the technical repertoire for interaction. Widespread contact with low-cost personal computers has increased computer literacy and user acceptance of the technology. However, much remains to be done to bring the human engineering of computer-based systems to the same level of professionalism as hardware and software engineering (Gaines & Shaw 1986a,b). For command and control systems involving expert systems and simulation, information presentation and ease of communication are of prime importance. A technically powerful system will be wasted unless it is comprehensible and accessible to its users, and gives them confidence in the basis of its operations.

Communication Protocols: The days of single-vendor, special-purpose systems designed against a static, well-defined specification are long gone. Integrated systems involve all aspects of the operation of a organization. Equipment from many vendors, purchased at different times, for different purposes, by different parts of the organization, has to operate together. In this lies the major practical obstacles to the creation of effective integrated systems at present. The technical problems are beginning to be resolved at the lower levels by the adoption of industry standards in communication protocols such as the open systems interconnection (OSI) standard (Day &

Zimmerman 1983), the provision in data-processing and database packages for the import and export of data from "foreign" systems, and the provision of data-communications for parameter-setting and data-collection in control and instrumentation systems. These alone are necessary but not sufficient to allow the widespread creation of integrated systems. They provide a syntactic framework for system integration but no coherent semantics in which to express it.

The Applications Ring: Basic Technologies: The outer ring in Figure 1 shows the main information technologies in integrated systems and their typical applications. Moving counter-clockwise around the ring:

Remote Communications are necessary to interface the organization with the organizational world in which it operations A particular organization is part of an infrastructure each part of which is trying to optimize its own operations.

Computation Systems perform the classical number-crunching tasks under the control of algorithms that with modern software engineering have a high degree of accuracy, conformity to specification and reliability.

Database Systems have become the classic core of an organization's information system, recording all its activities in a coordinated fashion and making the information available on a controlled basis to those systems and people with a legitimate need to know.

Control Systems are now increasingly integrated with information systems so that decisions can be implemented directly in the physical world.

Instrumentation Systems are also increasingly integrated to provide timely information on the state of the physical world and the effectiveness of operations.

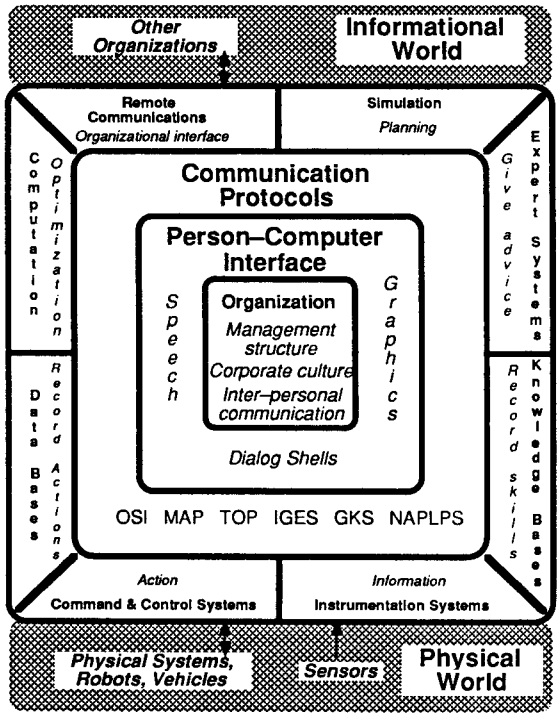

Fig.1 Structure of an integrated heterogeneous system

The Applications Ring: Decision Technologies: Three technologies in the outer ring of Figure 1 are particularly relevant to decision aids in integrated systems.

Knowledge Bases are distinguished from databases by containing inference rules rather than facts. The rules may necessarily be heuristic in nature, generating plausible advice rather than hard facts. Whereas databases record an organization's activities, knowledge bases record its skills.

Expert Systems are the means to apply a knowledge base to the solution of a particular problem. They stand in relation to the knowledge base as does conventional computation to the data base. They differ in that their role as advisors and dependence on heuristics makes *accountability* and *comprehensibility* of major importance. An expert system has to be able to justify and explain the basis of its advice.

Simulation has been separated from other forms of computation because it is based on a model of some other system and it is the relation of the simulated model to that of the system modeled that is of paramount importance. The models used in simulation to infer the behavior of a system are generally deeper than the models used in expert systems to infer the proper advice about a system, even though both may be concerned with the same system (Gaines 1986).

KNOWLEDGE STRUCTURES UNDERLYING HETEROGENEOUS SYSTEMS

What are the natural knowledge structures for integrated heterogeneous systems? Figure 1 captures the key problem areas for such systems and these should form the top level knowledge islands, but what about the relations between them? How do the corporate culture, person-computer interface, network protocols, information technologies, physical and informational worlds relate, and how does this affect operations? The only strongly formalized and standardized structures within Figure 1 are the network protocols, and it is useful to examine these in more detail because they involve concepts that are useful in formalizing more of topics encompassed in Figure 1.

The most important concept underlying the OSI Reference Model is that of *virtual circuits*. Designers of a particular level are concerned primarily with its relation with the equivalent level in the system with which it is communicating. They need have little concern with the levels above or below it. Hence they think in terms of messages passing directly between equivalent levels, through a "virtual circuit," rather than through the multiple levels of the actual system. The top level above the OSI defined layers is the "user level virtual circuit," representing the interactions between user programs through the communications network, and hence the interactions between users themselves, and their activities, if the programs are interactive.

This concept of virtual circuits between users, and user activities, is very powerful in enabling Figure 1 to be restructured to provide a more integrated framework for the other sections not directly included in the communications protocol. It can be redrawn in such a way that the relation of managerial and operations considerations to higher-level protocols in the network is apparent.

Figure 2 shows the same conceptual blocks as those in Figure 1, but now in hierarchical form with the person-computer interface, information systems, organizational structure and external worlds shown as higher level "layers" above the network layer. In OSI terms these upper layers may be seen as subsuming and extending the presentation and application layers of the protocol, and give substance to the user level virtual circuit.

The virtual circuits between the human-computer interface layer and the information systems layer in Figure 2 are easy to understand. Users of a computer terminal do not think of themselves as communicating through a network but rather as directly interacting with some data-processing sub-system. This layer may go beyond the OSI specification but it seems a natural extension of it to the technical aspects of the interaction between people and computing systems.

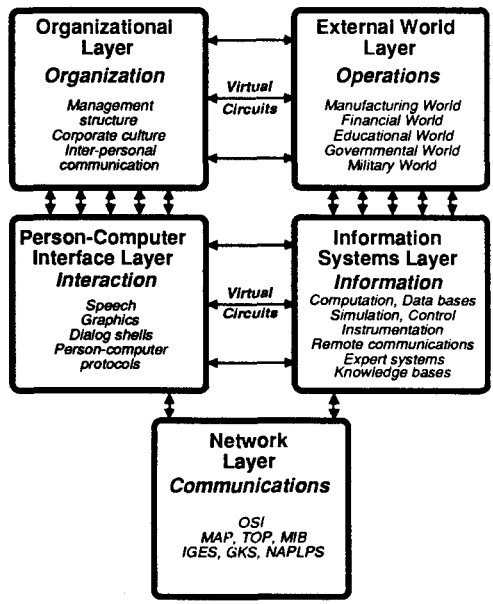

Fig.2 Knowledge structures for heterogeneous systems as a hierarchical protocol

The virtual circuits at the level above, between the organizational layer and the external world layer, extend the OSI concept a step further. The sales manager does not see himself as interacting with a complex administrative and productive system, but rather as communicating directly with a customer organization by supplying it with goods. The chief executive does not see himself as concerned with that level of communication but rather as communicating directly with the board and shareholders through a flow of profits and dividends. The main problem in extending the model to the management level is that people are necessarily involved, and they have different dynamics from technical systems. However, this is also true for the level below where human-computer interaction is involved, and models have been developed at this level that integrate the human and technical components within a hierarchical layered communication model (Gaines 1988c).

FUSION: A KNOWLEDGE-BASED NETWORK COGNITIVE SUPPORT SYSTEM

Figure 2 is a major step towards modeling an integrated heterogeneous network. Each of the boxes represents a major conceptual structure in information and management science. The vertical links between boxes represent a change in levels of abstraction. The horizontal links between boxes represent conceptual information flows, or virtual circuits. The overall framework partitions a heterogeneous systems into relatively well-known domains, and the links between them represent relatively well-defined flows. Note that the partitioning is functional rather than geographic. It is based on the type of activities being undertaken rather than their location in the network or organization. Note that the functionality is abstract rather than specific. It is based on the basic nature of the activities rather than their specific technical or organizational roles. These other dimensions, of the physical and logical organization of the network, and of its domain-specific functionality, are additional to those analyzed through Figure 2. Note also that the framework, although well-structured, encompasses highly heterogeneous systems—it conceptually packages functionality and flows but presupposes nothing about how they are implemented.

Figure 2 is the basis of our development of a generic intelligent network user support system, FUSION. Each box represents an island of knowledge about the system: the communications; the information systems, hardware and software; the user interfaces; the users and their organization; and the world in which they are operating. Within each box there is knowledge about the items and issues relating to its operation. Between boxes there is knowledge about inter-relations, basically the answers to why? and how? questions.

The following discussion outlines the design of FUSION as shown in Figure 3, a network cognitive support system based upon this analysis. The main tasks of FUSION are: for the system side of the model, state representation; and, for the user side, advice giving. The model has two interacting parts, one that models the heterogeneous network (*Internal System Model*), and one that represents FUSION's model of the user (*System View of User*).

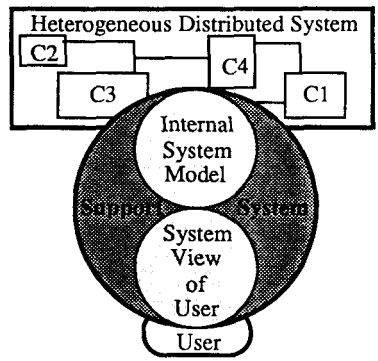

Fig. 3 Basic Model of FUSION

FUSION—Internal System Model

Th internal system model in FUSION is responsible for maintaining a representation of the current state of the network that can be used to aid in completing user tasks. A user task is a step in the fulfillment of an intention. This can range from sending mail, to editing a document, to copying one line, to typing one character (Norman 1984).

A task can be viewed as the execution of a plan, or in computer terms, an algorithm. This algorithm may need a software package, several software packages, or part of a package. Thus from an internal system point of view, task fulfillment is related to the functionality of the software in the system. It is also important to know what software is currently available. This affects what elements are available to construct a plan from. In addition, the functionality of the interface needs to be known. This will allow the system to aid the user in interacting with the software.

Software is procedural in nature. It specifies a set of steps that needs to be accomplished. Humans also have declarative knowledge. The analogous entities in a network are databases. These two sources give the first part of the internal system model. In a user, procedural and declarative knowledge are static quantities. To be of use they must be instantiated in some form. The same is true for computers. For the software to actually accomplish a task, it must be instantiated in the form of a process. The distribution of these processes (or *loading*) gives an indication of how busy elements of the network are. If the users current task is computationally intensive, knowledge of loading will allow the task to be distributed in a reasonable fashion. In terms of Rissland's (1984) knowledge sources, tools includes those available and being used.

Processes are the second source of knowledge. At the lowest level, they are a set of instructions executing on some processor (or processors). A particular piece of software may only run on specific architectures, a package may require advanced graphical functions not available on low end terminals. Thus a knowledge of the type of machines (*platforms*) available is needed. Rissland (1984) refers to this as knowledge of modalities.

It is also useful to know how to access particular platforms. If the software with the required functionality needs a particular type of platform, the system needs to know where this platform exists. The layout will also help to reduce communication costs between elements. Thus a knowledge of the topology of the network is required. The three sources of knowledge allow the internal system model to produce a complete model of the state of the system.

FUSION—System View of User

The main goal of the system is to aid a user in task completion. This is done in two main ways, support of the users cognitive model (*cognitive support*), and an advice paradigm. The cognitive model is supported at the user interaction level through the presentation of the resources. This uses a well designed and consistent interface, but does not require knowledge of a specific user to be effective (Rissland 1984, Greenberg 1988).

Literature on advice falls into two categories, expert systems and tutoring systems. The difference between the two types is the initiator of the advice. In an expert system, advice is given in answer to specific queries, and is thus user initiated. A tutoring system may give this type of advice, or the system may take the initiative.

The advice paradigm does require knowledge of the user on several levels. At the highest level, the advice should be relevant to the user. It is pointless to give a message about memory exceptions and core dumps when a user is attempting to mail a letter. Meaningful advice requires knowledge of the users intentions (Norman 1984, Carroll and McKendree 1987). This forms the first part of the system user view.

After forming an intention, a user needs to fulfil it. This requires a planning process which draws on the users knowledge of the system, as well as their strategic knowledge on how to plan. In the actual planning action, the user will be influenced by their knowledge of the world. Thus the systems user view needs access to all three types of knowledge forming the second part.

After the intention has been broken down into general tasks, and the tasks have been matched to resources, the user will need to interact with each resource. This interaction may differ from resource to resource. The user may also be at different levels with different resources. Thus some knowledge of a user role specific to a resource is needed. Knowledge of these is useful since users tend to reuse sets of resources in particular ways (Greenberg 1988). This knowledge of user role forms the final part of the system user view.

These three components give a model of the user that are used to assist users in completing intentions. Combining these three components with those from the internal system model, and the basic architecture yields a more complete picture of the system as shown in Figure 4.

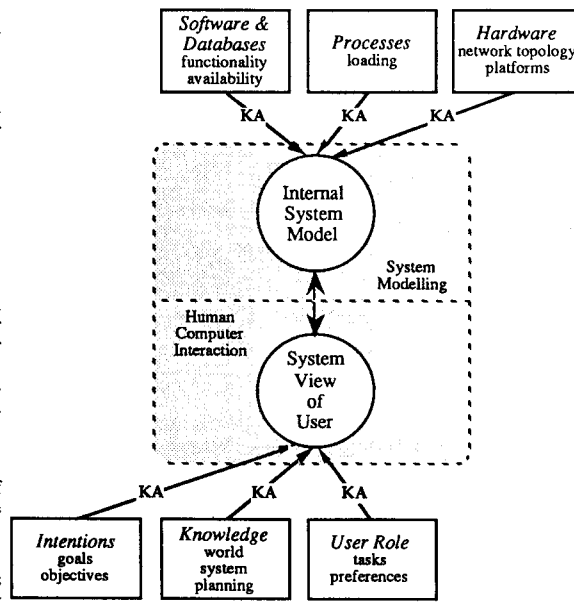

Fig.4 The FUSION System

KNOWLEDGE SOURCES AND ACQUISITION IN FUSION

The components of the expanded view in Figure 4 correspond to major types of knowledge needed about both sides of the communication between user and heterogeneous system. The system side contains information on the available resources :

Software & Databases provide the functionality for completion of user tasks. This includes the functionality of the user interface. Knowledge of where applications and databases reside is also needed. These are passive entities, they can do no work themselves.

Processes are the instantiation of software and databases. These are the active entities on the network. Knowledge of how busy machines are is important. It can be used to determine where best to run a particular process.

Hardware has two aspects. The topology of the network provides information about how to access software and databases. It also figures into time costs. Knowledge of platforms can be used in determining what type of interface to present to a user. A text terminal would not be able to support general graphics.

The user side has information about the individual user :

Intentions represent the goals and objectives of the user. These may change over time, and there may be sub–levels. For instance, a user may start by writing a letter to a friend, but digress into reading mail.

The *Knowledge* of the user ranges from the very general to the specific. The world knowledge affects the way the user thinks about the system. Knowledge of the system allows a user to plan the completion of an intention. Planning is the strategic knowledge about how to plan. All of this knowledge can change over time.

User Roles represent a user trying to fulfil a specific goal. This may be editing a document, reading mail, or building an application. Each role has a set of preferences and tasks associated with it. The preferences are analogous to a workbench, and the tasks are the tools that can be used (Greenberg 1988).

The rest of this section will discuss the sources of the knowledge in each of the components. It will also discuss some of the acquisition techniques used.

Software & Databases: This knowledge is mainly used to match user tasks to available resources. It could also be used to explain why a resource no longer works when it is upgraded. Information about where to find an old version is included in this knowledge. This information can be maintained by use of versioning tools.

The functionality of the resource is needed for task matching. The knowledge of functionality resides in three places, the manual, experienced users' heads, and the design specification:

The current role of text analysis in knowledge acquisition is either to provide a knowledge engineer with a basic view of the domain (Linster 1988), or to provide a list of useful concepts for an automatic acquisition process (Woodward 1988). Neither of these methods will give specific information on functionality. A solution is to assign the task of acquiring the functionality knowledge in the manual to a human. This solution can be expensive, especially since human time tends to be more valuable than computer time.

Automatic acquisition from an expert is possible, but is also not a good solution. The time of an expert is valuable. This is important since any time the resource is updated, the knowledge will need to be updated. It also may not be the same expert giving the knowledge.

A better source is the design specification. To use this source requires making the knowledge of functionality explicit. This is done by using computer aided software engineering (CASE) tools during the design of the resource. The process of acquiring knowledge about functionality now becomes one of using the knowledge from CASE tools. This is reasonable for large software packages, since the industry tends to use these tools.

Processes and Hardware: Though these two components have different levels of knowledge, they are both covered by network management techniques. Pasquale (1988) gives a set of characteristics that an expert distributed resource manager should exhibit : ability to deal with uncertainty, ability to react to changing situations, interpretive and inferential capabilities, ability to collaborate and coordinate, and high performance. These criteria can be reduced since some are equivalent. The ability to deal with uncertainty stems from the interpretive and inferential capabilities. The ability to react is related to high performance.

Knowledge based systems fit the requirement for dealing with uncertainty, and distributed problem solving is also possible (Durfee 1988). The problem will be the performance time. This requirement stems from management needs at a very low level of network operations. Such elements as packet routing and network error detection/correction, are not as critical to user support. Thus the time factor is not as important in FUSION.

At the hardware level there are two types of knowledge required. The topology information is generally available in configuration files, such as those found in TCP/IP. System administrators are the source of this knowledge.

The knowledge of platforms is functional and covers two levels. One is the nature of the devices present. This is information the characteristics of input and output devices, such as the type of screen, and the presence of a mouse. Most systems contain ways to access this information. A simple example is the TERMCAP database in UNIX systems.

The other level of functionality is the communication protocol. There is a definite move toward standardization of some levels of this interface (for example ISO/OSI). The major difference occurs at the operating system level. There is a great deal of difference between IBM-JCL and UNIX. The simplest way is to have information on the type of operating system present in the topographical knowledge.

Automatic acquisition of this knowledge can be done in a simplistic way. Try to talk with the particular platform using all the dialects that are known until success is achieved, or all the dialects are exhausted. This is an exhaustive search, and not very efficient. It is possible to place constraints on the search by examining the hardware of the platform. This should give some clues as to possible operating systems.

Intentions: The idea of acquiring user intentions seems to be very difficult. However, the problem can be heavily constrained. The only intentions of interest are those relevant to task completion. The fact that the user wants to mail Fred Blogs is important, the fact that he wants to convince Fred to go fishing for the weekend is not directly relevant.

The first level of constraint can take place in the interface design. It can be structured to acquire general intentions by initial user actions. An example would be presenting the user with the screen in Figure 5. In this example, the user is required to specify what general activity they wished to perform before proceeding. This can be as general as the intention to mail, but it could also be more specific. Each node in the hierarchy of intentions contains information on what the intention requires. Since the user can add nodes, they could create a node called 'mail fred' that contained all the resources used for that operation.

The next level of intentions can be acquired by observing that FUSION is a cooperative advisory system. Kass & Finin (1988a,b) present a set of user model acquisition rules that can be used to build a model of the user sufficient for tailoring explanations. This is sufficient for helping novice users gain familiarity and for aiding advanced users in their tasks (Kass & Finin 1988a).

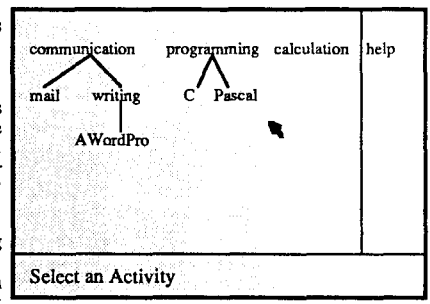

Figure 5 : A Screen for High Level Intentions

Another important observation is about the length of time the knowledge is needed. Since a user comes to a computer with an intention, the knowledge associated with it need only exist for the duration of the relevant session. When intention *A* is completed, the knowledge for it is no longer needed. Thus, unless specifically requested, the system need not remember a large amount of data. In concrete terms, an intention *A*, corresponds to a temporary node *A* in the hierarchy of Fgure 5. This node is a workspace (Greenberg 1988) associated with intention *A*. Unless the user wishes to keep the workspace, the system can dispense with the node once the action is completed.

Thus intentions can be ascertained in two ways. The high level intentions can be found by constraining the user to specify them. Finer details can be gained using both interview and observation.

User Knowledge: According to Norman (1984), a user cycles through four stages of activities, forming and intention (intention), selecting an action (selection), executing the action (execution), and evaluating the income (evaluation). The user intentions component in FUSION, contains high level intentions, the user knowledge component is used to break this high level down into actions. This corresponds to Norman's selection stage :

> "Support for the selection stage comes principally from memory aids ... that allow the user to determine the possible commands and their modes of operation, prerequisites and implications. Selections can be enhanced by "workbenches" that collect together relevant files and software support into one convenient location." (p. 368)

FUSION uses memory aids in two ways. The interface is set up as a hierarchical graph of activities (high level intentions), and the advice system can give information on commands and resources. Norman also suggests four sources for knowledge of commands: retrieval from memory; reminded by another person, the system, or a manual; construct or derive possibilities; teaching by another person, the system, or a manual.

For memory retrieval, the user already knows what commands are needed. Reminding is a task for the advisory system, with teaching adding a tutorial element to this. Construction is a candidate for knowledge acquisition, since it involves the users ability to plan, and their knowledge of the system.

This type of activity is a synthesis task, and specifically a planning task. It is possible to use an explanation based learning approach to knowledge acquisition to assist the user in planning to satisfy the intentions (Levi et. al. 1988). In this case, the planner would not be autonomous, instead it would interact with the user in an interview style to refine the plan. This is also close to a tutorial style, thus as the plan is constructed, the user learns more about the resources of the system and how they can be combined.

User Role: A user role refers to the set of characteristics and preferences associated with a user performing a particular task. The characteristics may differ between applications, and may not reflect the level of sophistication of the user.

> "It also inclines more experienced users to stagnate, for when situations occur that could be handled by new procedures, these users are likely to stick with the procedures they already know, regardless of there efficacy." (Carroll and McKendree 87, p. 14)

This indicates that a knowledge of how a user interacts with a resource is useful in two ways. First, the user is likely to be consistent in the way they perform particular operations. The system can then suggest and perform actions sequences. Secondly, it gives a basis from which the system can tutor the user on other functionality provided by the resource.

This type of knowledge is performance based. It consists of sequences of activities, and can be acquired by observing user actions. Since FUSION is designed to aid at the resource usage level, it is not concerned with interaction inside the resource. Thus it does not need to know about sequences of commands in a text editor, merely that editor *Y* is used to fulfil task *X*.

The system should also be able to present a user with an environment that is suitable to the current task. It should acquire knowledge of window setups, command aliases, and preferred applications. This knowledge can also be acquired by observing user actions.

SOME TECHNOLOGIES FOR KNOWLEDGE ACQUISITION IN FUSION

To satisfy the goal of aiding a user in the completions of tasks in a large scale heterogeneous network requires a diverse range of knowledge. As such, the knowledge acquisition performed by the system is also varied. Some methods fall outside the current literature on knowledge acquisition, but do exist in other literature. Thus the system needed to satisfy the goal must integrate different technologies. The FUSION system sets out to do this.

Figure 6 gives the sources of knowledge used and some of the techniques used to acquire the knowledge. Current knowledge acquisition methods cover most of the system view of the user. Acquiring intentions is outside this scope, but is covered above. Since part of the aim of this paper is to address knowledge acquisition techniques that are outside the norm, the rest of this section will concentrate on the internal system model.

Most of the techniques for the internal system model are not strictly recognized as knowledge acquisition. However, they do provide solutions to some problems that are recognized as requiring knowledge based solutions. These include (taken from Boose 1988, page 3–1):

Source	Technique(s)
Software & Databases functionality availability	versioning tools CASE
Processes loading	scripts / stats reporting daemons
Hardware network topology platforms	reporting daemons
Intentions goals objectives	interface structure cooperative advisory dialog rules
Knowledge world system planning	interact with user in a problem-solving mode
User Role tasks preferences	interact with user using interview for refinement observe user behaviour

Fig.6 Techniques of Acquisition in FUSION

Analysis tasks: Diagnosis—inferring system malfunctions from observables.

Synthesis tasks: Configuration—configuring collections of objects under constraints in relatively small search spaces; Design—configuring collections of objects under constraints in relatively large search spaces.

Tasks combining analysis and synthesis: Monitoring—comparing observables to expected outcomes.

Diagnosis requires the acquisition of the observables. Configuration and design require acquisition of the collections of objects, and of the constraints. Monitoring requires acquisition of the observables. The observables in FUSION are the hardware platforms, process distributions, and network topology. Software and databases are the collections of objects to be constrained by both their functionality and the user's needs.

Software & Databases : CASE

There are two pieces of knowledge required about the software and databases, what they do (*functionality*) and where they are (*availability*). Both types of knowledge can exist in computer aided software engineering (CASE) environments. Houghton (1987) surveys both the characteristics of tools and the state of the art in CASE. Part of the software life cycle is to define its functionality. Using a tool to specify the functionality makes the knowledge explicit..

Another phase of the life cycle is operations and maintenance. During this phase, tools are used to track changes functionality. These tools also track configuration releases. This knowledge is useful when a user encounters a problem with a new release of a package. An example is code that compiled under the old version but not under the new. The configuration and release knowledge (versioning) allows tells the system that a new compiler is in use.

This versioning capability acquires only part of the knowledge of availability. In addition to knowing what the current version is, the system must know where to locate the software (or where old version are). A system administrator could inform FUSION when new software is installed or removed. However this is both tedious for the administrator and error prone.

A better solution is to have FUSION automatically check for new software. At the very least, the modification dates of executables can be checked. If they have changed, then the software has changed. Removal of software is indicated by absence of executables. It is still difficult to detect the version of the software, though this can often be done by invoking the package with a particular argument.

More recently, vendors of workstations have been providing software release and installation control. A good example is the Apollo RAI system. This consists of a set of tools for reading software from media and installing software onto workstations. Software is initially installed in a specified location (called the Authorized Area). From there, it can be installed on any other workstations on the network. The authorized area contains information on what software is available to be installed, and what the most recent update is. It would be easy to add some code that kept information and where and when software was installed.

Processes

Finding out the processes that are running on a single workstation is a well established technology. The UNIX 'ps' command is a simple example. This can provide information on the name of the process, the time it has been running, and the current state. Most networks have methods to find the processes on remote machines. Again from UNIX, the 'rsh' command allows execution on a remote machine. Since the output is directed to the current machine, it can be used to execute a 'ps' on the remote machine.

Loading measurement is also well established. The usual method is to have a continually running process (daemon) that broadcasts the current load of the machine at specified time intervals. An example is the TCP/IP 'rwhod' daemon that updates files used by the 'ruptime' command. This command gives information about the state of available machines on the network, including the approximate loading factor of the machine, though it is not as good predictive expert system. It will be considerably simpler.

Hardware

This is one the most difficult aspects to determine. It includes the processors (CPU, floating point, ...) and peripherals (disks, screen, ...) of a machine. The simplest solution is to manually enter the information in some predetermined place, which is an unreliable method at best.

Automatic detection is currently difficult. Most machines have a method to determine the current architecture, but the method tends to vary widely. For example, the Sun machines have a command called 'arch' that tells you about the current architecture. The same information is accessible on an Apollo, but requires the creation of a program to interact with system calls.

Another slightly more reliable method is the UNIX 'TERM' environment variable. This is usually set to the type of terminal (or machine) that the user is on. It requires some work by a system administrator, though it is a de-facto standard. The value of the variable serves as a key to the 'termcap' database. This contains information on the characteristics of the current terminal, including cursor manipulation, key bindings, and screen type.

CONCLUSIONS

Intelligent network support systems modeling a heterogeneous system to provide administrative and user support are feasible at the current state of the art in object-oriented systems. They require the development of overall organizational models that encompass corporate goals and management systems as well as the technological infrastructure supporting these. This paper has developed such an organizational model and discussed its application to the development of an intelligent network support system for integrated heterogeneous systems.

FUSION is an integration of five areas : knowledge based systems, advice giving, knowledge acquisition, object oriented systems, and user modeling. As such it is also a heterogeneous mixture of technologies. The object oriented basis of the system provides a structured protocol for communication between the different types of knowledge. The knowledge facilitates both the giving of advice and the modeling of the user. The acquisition techniques (both traditional and new) are used to gain the knowledge. The integration of advice systems, knowledge based systems, and new acquisition techniques provides a powerful tool that can help users, both novice and experienced, in a large heterogeneous network environment.

ACKNOWLEDGEMENTS: Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

Boose, J. H. (1988). A Survey of Knowledge Acquisition Techniques and Tools. **Proceedings of the 3rd Knowledge Acquisition for Knowledge–Based Systems Workshop**, Banff, Canada, November.

Carroll, J. M. and McKendree, J. (1987). Interface design issues for advice-giving expert systems. **Communications of the ACM** 30(3), 14-31 (January).

Day, J.D. & Zimmerman, H. (1983). The OSI reference model. **Proceedings IEEE**, 71(12), 1334-1340 (December).

Deal, T.E. & Kennedy, A.A. (1982). **Corporate Cultures: The Rites and Rituals of Corporate Life.** Reading, Massachusetts: Addison-Wesley.

Durfee, E. H. (1988). **Coordination of Distributed Problem Solvers.** Norwell, Massachusetts: Kluwer.

Gaines, B.R. (1986). Expert systems and simulation in the design of an FMS advisory system. Hurrion, R.D., Ed. **Simulation in Manufacturing.** pp.311-324. Bedford, UK: IFS Conferences.

Gaines, B.R. (1988a). Structure, development and applications of expert systems in integrated manufacturing. Kusiak, A., Ed. **Artificial Intelligence Implications for CIM.** pp.117-161. Bedford, UK: IFS Conferences.

Gaines, B.R. (1988b). A conceptual framework for person-computer interaction in distributed systems. **IEEE Transactions on Systems, Man & Cybernetics SMC-18**(4), 532-541 (July).

Gaines, B.R. & Shaw, M.L.G. (1986a). From timesharing to the sixth generation: the development of human-computer interaction part I. **International Journal of Man-Machine Studies**, *1*(1), 1-27 (January).

Gaines, B.R. & Shaw, M.L.G. (1986b). Foundations of dialog engineering: the development of human-computer interaction part II. **International Journal of Man-Machine Studies**, 1(2), 101-123 (February).

Greenberg S. (1988). **Tool User, Reuse, and Organization in Command Driven Interfaces.** PhD. Thesis, Department of Computer Science, University of Calgary.

Hall, E.T. (1959). **The Silent Language.** New York: Doubleday.

Houghton, R. C. (1987) Characteristics and Functions of Software Engineering Environments: An Overview. **ACM Sigsoft Software Engineering Notes**, Volume 13, Number 1, January, 64-84.

Kass, R. and Finin, T. (1988a). The Need for User Models in Generating Expert System Explanations. **International Journal of Expert Systems** 1(4), 345-375.

Kass, R. and Finn, T. (1988b). A general user modelling facility. **Proceedings of the ACM SIGCHI '83 Human Factors in Computing Systems**, pages 54-57, Boston, December 12-15.

Levi, K., Perschbacher, D., Shalin, V. (1988). Learning Plans and Information Requirements for Pilot Aiding. **Proceedings of the 3rd Knowledge Acquisition for Knowledge–Based Systems Workshop**, Banff, Canada, November.

Linster, M. (1988). A Critical Look at KRITON. **Proceedings of the 3rd Knowledge Acquisition for Knowledge–Based Systems Workshop**, Banff, Canada, November.

Norman, D. A. (1984). Stages and levels in human-machine interaction. **International Journal of Man-Machine Studies** 21, 365-375.

Pasquale, J. (1988). Using Expert Systems to Manage Distributed Computer Systems. **IEEE Network** (Special issue on expert systems in network operations and management), September, pages 22-28.

Rissland, Edwina L. (1984). Ingredients of intelligent user interfaces. **International Journal of Man–Machine Studies** 21(4), 377-388 (October).

Woodward, B. (1988). Knowledge Acquisition at the Front End: Defining the Domain. **Proceedings of the 3rd Knowledge Acquisition for Knowledge–Based Systems Workshop**, Banff, Canada, November.