(30) Zefirov, N. S.; Tratch, S. S. Symbolic Equations and Their Applications to Reaction Design. *Anal. Chim. Acta* **1990**, *235*, 115-134.
(31) Tratch, S. S.; Devdariani, R. O.; Zefirov, N. S. Combinatorial Models and Algorithms in Chemistry. Configuration-topological Analogs of Wiener Index. *Zh. Org. Khim.* **1990**, *26*, 921-932 (in Russian).
(32) Zefirov, N. S.; Kozhushkov, S. I.; Kuznetsova, T. S.; Kokoreva, O. V.; Lukin, K. A.; Ugrak, B. I.; Tratch, S. S. Triangulanes: Stereoisomerism and General Method of Synthesis. *J. Am. Chem. Soc.* **1990**, *112*, 7702-7707.
(33) Zefirov, N. S.; Kaluzhnin, L. A.; Tratch, S. S. The Generalization of the Wreath Product of Permutation Groups and Its Application to Description of Chemical Structure Formulas. In *Investigation in Algebraic Theory of Combinatorial Objects*; Klin, M. H., Faradgev, I. A., Eds.; VNIISI Publ.: Moscow, 1985; pp 175-186 (in Russian).
(34) Read, R. C.; Corneil, D. G. The Graph Isomorphism Disease. *J. Graph Theory* **1977**, 339-363.
(35) Goldberg, M. K. A Nonfactorial Algorithm for Testing Isomorphism of Two Graphs. *Discrete Appl. Math.* **1983**, *6*, 229-236.
(36) Wiswesser, W. J. *A Line-Formula Chemical Notation*; Crowell: New York, 1954.

ALF-A: A Knowledge Acquisition Tool for Troubleshooting of Laboratory Equipment

HENRIK ERIKSSON* and PER LARSES†

Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden

Received September 11, 1991

Expert systems can be used to support troubleshooting of technical equipment, such as laboratory instruments. A critical factor and bottleneck in the development of such knowledge-based systems is the acquisition of relevant knowledge for the task. Computer-based knowledge acquisition (KA) tools can support knowledge elicitation from domain experts by transforming structures elicited from or entered by nonprogrammers into knowledge bases. A KA tool (ALF-A) that supports the development of expert systems for fault diagnosis in DNA sequencing machines has been developed. Different categories of experts can use the tool directly (i.e., without an intermediate knowledge engineer) to enter their knowledge according to a predefined model of troubleshooting.

1. INTRODUCTION

Ensuring a high degree of availability for laboratory instruments (e.g., DNA sequencers) is important for most laboratories. Expert system technology offers an approach to consultation software that can facilitate troubleshooting. Such computer programs can lower the amount of time spent on fault finding as well as fault correction and, by enabling nonspecialists to do much of the work, reduce the burden on highly-qualified equipment specialists.

Expert systems are typically organized around *knowledge bases*, which contain knowledge structures for the problem area in question in a declarative format, and *inference engines* (problem solvers) that draw conclusions from the current input guided by the knowledge base. The performance of these systems is critically dependent on the contents of their knowledge base.

The process of modeling expertise, i.e., *knowledge acquisition* (KA) from domain experts, is a problem and bottleneck in the development of knowledge-based systems. Manual development of a reasonably complete knowledge base for troubleshooting a set of equipment can be a huge task, both in terms of the knowledge engineering effort and the expert time. Computer-based KA tools can be used to support knowledge acquisition from experts.

A KA tool (ALF-A) that facilitates the development of knowledge bases for fault detection in a DNA sequencer is described in this article. One of the salient aspects of this problem area is that the set of equipment required involves both chemistry (e.g., gel), hardware (e.g., laser units), and software (e.g., control programs). Consequently, experts in this area are primarily specialized in one of these aspects and, thus, are proficient only in one group of faults. Even though there exist dedicated shells for troubleshooting [e.g., TEST (*15*) and TestBench (TestBench is a trademark of Carnegie Group, Inc., and Texas Instruments, Inc.) (*3*)] and KA tools for troubleshooting [e.g., TDE (*13*, *14*)], it is difficult to find one that simultaneously meets requirements from all aspects of this type of equipment.

Due to the nature of the equipment, a model that reflects the troubleshooting task (shallow model) was used rather than a model of the chemical/physical processes and the equipment (qualitative model). The conceptual domain model is based on a *symptom-fault tree*, which controls the reasoning strategy. This model is accepted to a large extent by the experts involved in this project. The KA tool developed implements this conceptual domain model and allows experts to enter new symptoms and faults as well as define the relationships among them. Knowledge entered can be stored persistently on file or transformed into target knowledge bases.

This article is organized as follows: Section 2 presents the background and motivation of this project. Approaches to tool support for knowledge acquisition are discussed in Section 3. An overview of the KA tool is provided in Section 4. Section 5 treats the specification and generation of the ALF-A system. Related work is discussed in Section 6. Finally, a summary and conclusions are provided in Section 7.

2. BACKGROUND

One set of equipment that can be used to sequence DNA is the "Automated Laser Fluorescent (A.L.F.) DNA Sequencer" (Automated Laser Fluorescent (A.L.F.) DNA sequencer is a trademark of Pharmacia Biosystems AB) (*9*, *20*). The A.L.F. DNA Sequencer automates detection of fluorescently labeled DNA molecules. The system comprises (a) an electrophoresis and laser fluorescent subsystem, (b) the control and analysis software for PC compatibles, and (c) a sequencing reagent kit containing the chemicals required. Labeled DNA migrates through a gel and intercepts the laser beam that excites the fluorescent labels. The gel consists of 40 electrophoresis lanes. Photodiodes for each of the lanes detect emitted light. The system's capacity is two gels per day, which is equivalent to 8 kb at a rate of 1000 bases/h. After a run is completed, the computer calculates the DNA sequence and assigns ambiguity codes to hard-to-call bases.

* Address correspondence to this author at his present address: Stanford University Medical Center, MSOB X215, Stanford, CA 94305-5479.
† Present address: Enator Kunskapssystem AB, St. Larsgatan 12, S-582 24 Linköping, Sweden.

At this point in time the A.L.F. DNA Sequencer is a relatively new product, and there are only a handful of expert troubleshooters available. An expert system has the potential of reducing the burden on these experts, by helping users to locate and remedy faults. Expert systems can also assist in the training of new specialists. The purpose of the project described in this article, ALF-A, (ALF-A stands for *A.L.F. Acquisition*) was *not* to actually implement an expert system in this domain, but rather to develop enabling technologies in the form of a KA tool specific to this domain.

3. TOOL-SUPPORTED KNOWLEDGE ACQUISITION

Knowledge acquisition can be seen both as a modeling activity and as an expertise-transfer process. During early phases of development the modeling metaphor describes knowledge acquisition best. At this stage a *knowledge engineer* establishes an initial understanding of the domain (e.g., a vocabularly) and builds models of expertise in terms of the problem-solving behavior as well as the knowledge structure. KADS (*25*) is an established approach to knowledge modeling that produces such a conceptual model of the domain (including models of domain, inference, task, and strategic knowledge).

During later phases of knowledge acquisition the transfer metaphor can be used to describe the process. At this stage, a common language (model) for communication between the domain experts and the knowledge engineers involved should be established. Both the modeling and the transfer aspects of knowledge acquisition are considered as bottlenecks, and there is a need to find methodologies, techniques, and tools that can aid knowledge acquisition.

A number of different types of KA tools have emerged. Current approaches to KA tool support can roughly be categorized into four major types according to what aspect of knowledge acquisition they support:

Conceptualization Tools. This class of tools aids domain conceptualization and elicitation from experts. Some of the tools are based on psychological theories (e.g., personal construct theory) or techniques (e.g., card sorting). There are also KA tools that provide computer-supported protocol analysis (i.e., protocols from expert interviews). Examples of tools include ETS (*1*) and KITTEN (KSSO) (*22*).

Task-Specific KA Tools. Two of the most common tasks are *analysis* (e.g., classification, debugging, and diagnosis) and *synthesis* (e.g., configuration, design, planning, and scheduling). Combinations of tasks exist. Examples of task-specific KA tools are TDE (*13*, *14*) and SALT (*16*).

Domain-Oriented KA Tools. When the character of knowledge acquisition has shifted from modeling to emphasizing transfer, it is possible to build domain-specific KA tools that are based on the conceptual domain model. Such domain-oriented (or model-based) KA tools allow experts to edit knowledge structures according to the model. These tools address the mid and late stages of the development process and are not meaningful at early analysis and modeling phases. Examples of domain-oriented KA tools include OPAL (*18*), P10 (*6*), and KAVE (*21*).

Refinement Tools. These tools aim at improving an existing knowledge base, for instance, by identifying missing parts as well as inconsistencies and ask the user for more information. Such tools are particularly useful at later project stages, e.g., testing, debugging, and maintenance. Examples of refinement tools are TEIRESIAS (*4*) and SEEK2 (*12*).

There are also other KA tools that do not easily fit into these categories. Musen (*17*) classifies KA tools according to the model they present to their users. Surveys of knowledge acquisition tools and techniques are provided by Neale (*19*) and by Boose (*2*).

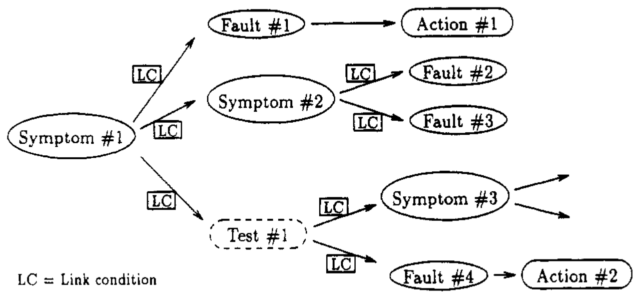


Figure 1. Sample fault tree.

We chose to perform a manual domain analysis and to develop a domain-oriented KA tool for the task of troubleshooting the A.L.F. DNA Sequencer in this project. Part of the motivation for this was the number of expert categories involved.

4. SYSTEM OVERVIEW

ALF-A is a domain-specific KA tool based on a conceptual model of troubleshooting knowledge for the A.L.F. DNA Sequencer. Domain experts can use ALF-A directly to enter, edit, and review their knowledge. Thus, ALF-A adopts the domain-oriented approach and uses the same knowledge acquisition strategy as KA tools such as OPAL (*18*) and P10 (*6*).

4.1. Fault-Tree Model. A quite shallow model was chosen for troubleshooting the A.L.F. DNA sequencer. The motivation for this decision was (a) limited project time, (b) many system components are difficult to model (partly because the instrument is relatively new), and (c) only a few components of each kind exist in the system.

In summary, the conceptual domain model supported by ALF-A is based on *symptoms* and *faults*. Symptoms indicate some kind of anomaly in the system, and faults are failures or problems that can be corrected to remove the anomaly. Faults can cause one or several symptoms, and a symptom may suggest a number of faults. This model was partly inspired by *fault-tree diagnosis* (*3*).

At the root of a fault tree there is a symptom (see Figure 1). Nodes beneath the root are subsymptoms, faults, or actions. Leaves are always remedy nodes (actions). The model assumes that faults can always be corrected, e.g., by replacing parts.

A condition can be associated with each link in the graph. By checking such *link conditions* (which are logical expressions), it is possible to distinguish between different causes and to focus on subsymptoms or faults. Explicit *test nodes* can be introduced to specify certain tests to perform. It is possible to declare the time requirements for each test to avoid particularly time-consuming test runs if possible (i.e., prefer easy tests).

Fault trees are used in the knowledge base to guide the problem solver. The inference mechanism traverses the tree in a depth-first manner and makes decisions upon which path to take based on tests, link conditions, and information provided at runtime. Figure 1 provides an example of a fault tree according to this model. In this case the inference mechanism will start by focusing symptom 1 (which is the fact that there is a problem with the equipment). The system proceeds by determining the truth value of each of the link conditions, for instance, by querying the user or by asking the user to perform certain tests (such as test 1). The system will continue by focusing on the next symptom selected (e.g., symptom 2) or, if a fault is reached, suggesting a remedy (e.g., fault 1 and action 1).

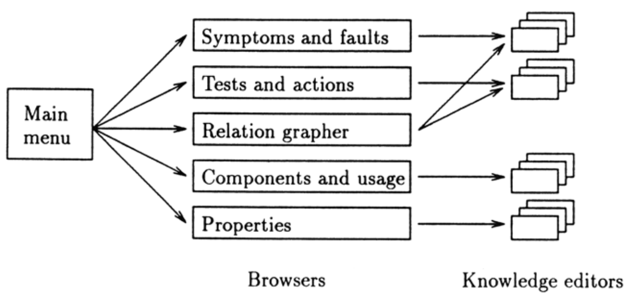


Figure 2. Dialogue structure in ALF-A (simplified).

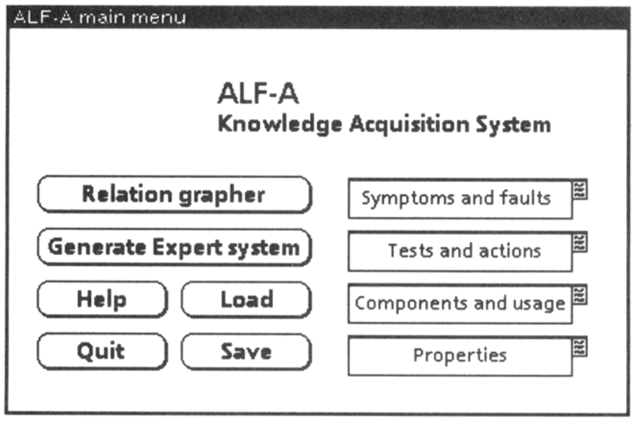


Figure 3. Main menu in ALF-A. Users can open various browsers and issue commands, e.g., save/load and knowledge base generation, from this main menu.

The structure of symptoms, faults, and rules at link conditions are, in this particular model, quite domain-specific, i.e., some of their properties are specific to troubleshooting A.L.F. DNA sequencers. ALF-A allows domain experts to enter relevant knowledge for this type of fault-tree diagnosis. However, ALF-A was not intended to be a general knowledge acquisition tool for troubleshooting. For instance, the user interface is tailored to this particular domain and a relatively small group of experts.

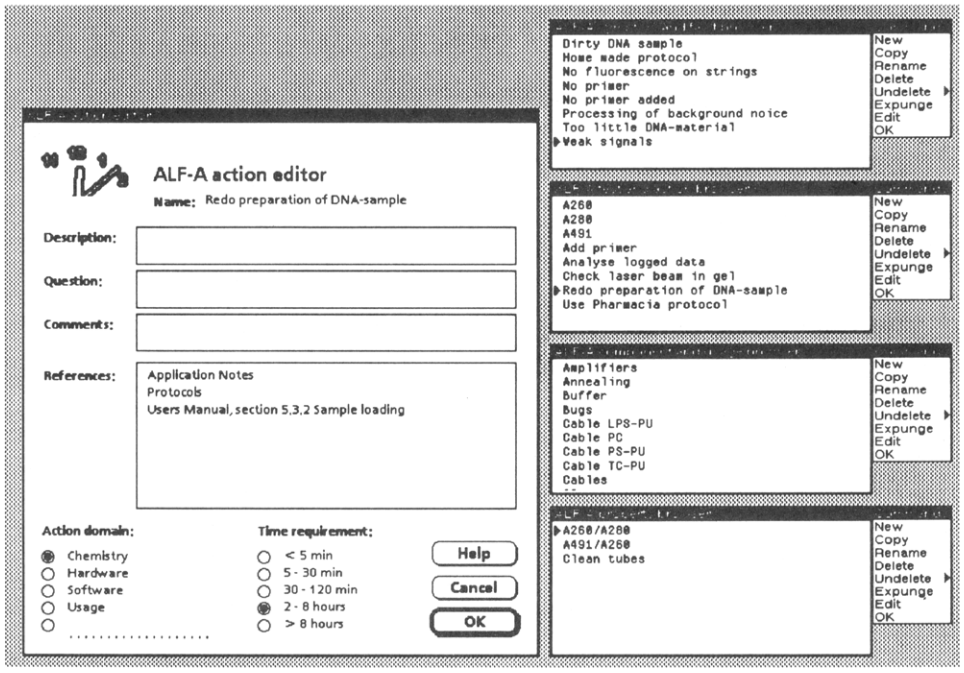
4.2. Knowledge Editors. The ALF-A system comprises a number of *knowledge editors*. The knowledge editors constitute the user interface for the experts. There are editors for *symptoms*, *faults*, *remedies*, *components*, *properties* (or parameters), and *link conditions*. The user can create and access such objects through *browsers*. The dialogue structure and the main menu in ALF-A are shown in Figures 2 and 3, respectively.

The main menu provides access to the browsers for (1) symptoms and faults, (2) tests and actions, (3) components and usage, and (4) properties. These browsers allow the user to create, copy, and delete objects as well as to bring up knowledge editors for editing. Figure 4 shows the four browsers. Another form of browser is the *relation grapher*, which is a graphical editor for fault trees. As indicated in Figure 3, there are also functions for save and load, as well as knowledge base generation in ALF-A.

Figure 4 also shows a knowledge editor for actions to take when a fault is found. This form includes fields for (a) a brief description of the action; (b) a comment (if required); (c) references to manuals, instructions, etc.; (d) the action domain (i.e., chemistry, hardware, software, or usage); and (e) the time requirements (see Section 4.1 for an explanation of time requirements).

Figure 5 shows the combined knowledge editor for symptoms and faults. The symptom edited is concerned with weak signals. There are fields for (a) a brief description of the problem; (b) a question text for the end-user; (c) a comment (if required); (d) references to manuals, persons, etc.; (e) the fault domain (i.e., chemistry, hardware, software, or usage); and (f) related concepts (i.e., tests, actions, or components). During discussion with the experts it became clear that certain abnormalities can be classified both as faults and as symptoms that can be corrected but also refined. Therefore, a combined knowledge editor for symptoms and faults is used, and the experts may indicate whether the abnormality should be considered as a symptom, as a fault, or both (see Figure 5).

By means of the relation grapher, experts can edit fault trees similar to the one depicted in Figure 1. The relation grapher is used to relate symptoms and faults to tests and actions (see Figure 6). There are functions for creating, moving, copying, renaming, deleting, and editing (i.e., opening underlying knowledge editors) nodes in the graph. Links between nodes can be created and deleted. The relation grapher also provides access to an editor for link conditions (not shown). Link


ALF-A action editor

Name: Redo preparation of DNA-sample

Description:

Question:

Comments:

References: Application Notes
Protocols
Users Manual, section 5.3.2 Sample loading

Action domain: [✓ Chemistry] Hardware Software Usage

Time requirement: < 5 min 5 - 30 min 30 - 120 min [✓ 2 - 8 hours] > 8 hours

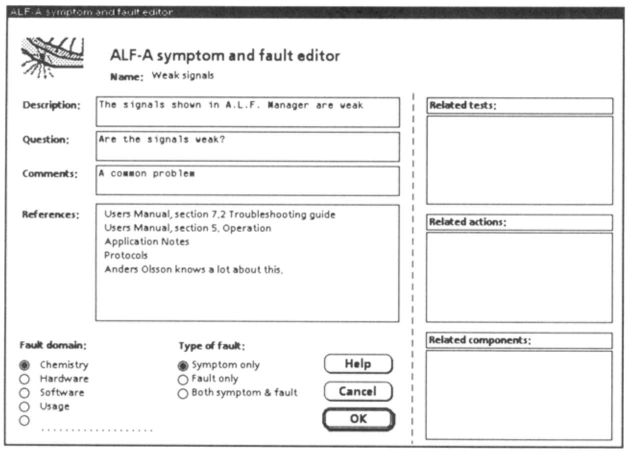
```
Dirty DNA sample
Home made protocol
No fluorescence on strings
No primer
No primer added
Processing of background noice
Too little DNA-material
▶Weak signals
```

```
A260
A280
A491
Add primer
Analyse logged data
Check laser beam in gel
▶Redo preparation of DNA-sample
Use Pharmacia protocol
```

```
Amplifiers
Annealing
Buffer
Bugs
Cable LPS-PU
Cable PC
Cable PS-PU
Cable TC-PU
Cables
```

```
▶A260/A280
A491/A260
Clean tubes
```

Figure 4. Sample action knowledge editor (left) and the four browsers in ALF-A (right).



ALF-A symptom and fault editor

Name: Weak signals

Description: The signals shown in A.L.F. Manager are weak

Question: Are the signals weak?

Comments: A common problem

References: Users Manual, section 7.2 Troubleshooting guide
Users Manual, section 5. Operation
Application Notes
Protocols
Anders Olsson knows a lot about this.

Related tests:

Related actions:

Related components:

Fault domain: [✓ Chemistry] Hardware Software Usage ....

Type of fault: [✓ Symptom only] Fault only Both symptom & fault

Figure 5. Sample symptom and fault knowledge editor.

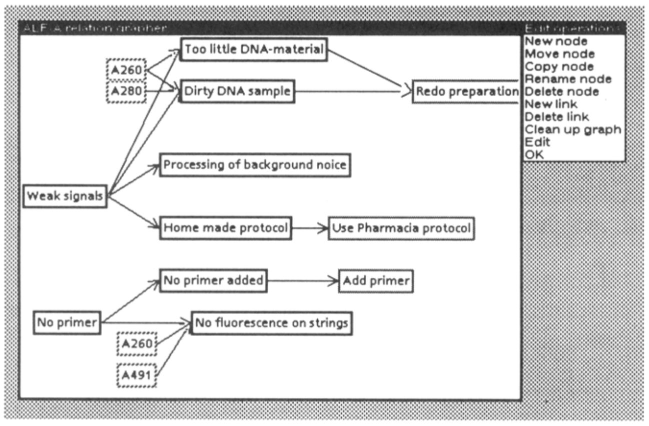


Figure 6. Relation grapher.

conditions are logical expressions (e.g., A260/A280 >= 1.7 and CleanTubes = True).

There is also a property editor that can be used to enter new properties of the equipment (not shown). The values of these properties are used in the reasoning process (in particular at link conditions). The property editor has fields for (a) a question text for the user (if the value is required at runtime); (b) a help text for the user; (c) the property category (i.e., chemistry, hardware, software, or usage); and (c) the property type (e.g., boolean, numeric, or keyword).

4.3. Knowledge Representation. The internal knowledge representation in ALF-A is based on *knowledge modules* and *update rules* that are used to transfer contents between knowledge editors and knowledge modules. ALF-A uses a set of *transformation rules* to transform the contents of the knowledge modules into Epitool (Epitool is a trademark of Epitec AB) (*5*) knowledge bases. A detailed description of these internal components can be found elsewhere (*7*).

Although a working expert system was not the primary goal of this project, a *core* knowledge base has been developed for testing purposes. This core knowledge base features basic declarations for the knowledge representation used (fault tree) and mechanisms for traversing the fault tree. The declarations include class definitions for tree nodes, symptoms, faults, tests, actions, and current status (status object). The KA tool generates subclasses or (static) instances of these predefined classes in the core knowledge base.

Table I summarizes some facts about the ALF-A system.

5. SPECIFICATION AND GENERATION OF ALF-A

ALF-A was developed by interviewing experts in the field and by analyzing the need for KA tool support. A large part of the implementation was automatically generated from a specification.

Table I. Facts about ALF-A

implementation environment	DOTS
addnl implementation language	Common Lisp (Medley) and CLOS
platforms	Xerox 1186 and Sun workstations
target language	Epitool
effort	2 person-months (including KA)
knowledge editors	27
knowledge modules	30
update rules	13
transformation rules	14

5.1. Meta-Tools. In general, the implementation of domain-specific KA tool support is a laborious task. Creating graphical user interfaces is time-consuming and requires skilled programmers. Implementation of the transformation from structures edited in knowledge editors into knowledge base code is often a nontrivial programming task. Normally, a KA tool is required to cover a large number of applications to justify the investment.

Recently, a number of meta-level KA tools have emerged. These meta-tools are essentially shells for implementation of domain-specific KA tools. Target KA tools can be specified to a meta-tool according to different principles, or *meta-views*, for instance, according to a model of the problem-solving method or according to an abstract model of the target KA tool architecture. Examples of such meta-tools include PROTÉGÉ (*17*) and DOTS (*7*). The meta-tool DOTS has been used to specify and generate ALF-A.

5.2. Establishing a Conceptual Domain Model. Much effort has been devoted to finding a suitable conceptual domain model which could be the basis for the KA tool. The purpose was *not* to acquire knowledge, but rather to elicit the knowledge structure required to develop the KA tool. Scarce expertise more or less forced the knowledge engineer into acquisition from multiple experts. A positive effect of this problem was that many experts contributed to the conceptual domain model and, thus, broadened the basis for the model.

During the initial interviews, the *tutorial interview* (*11*) was used. The expert gave an overview of the domain and explained the sequencing process and much of the terminology used. At subsequent interviews, the expert was asked to list all the components in the A.L.F. DNA Sequencer related to his/her area of expertise and to explain symptoms and faults that may arise in each component. The important result of this kind of interview was a catalogue of symptoms useful at later expert meetings.

The next stage included a series of *structured interviews* (*10*, *19*) that aimed at eliciting as much knowledge as possible for a number of symptoms and components. This led partly to a dead-end since this approach (i.e., going into details of components and subcomponents) assumed a quite deep model for troubleshooting, which is unsuitable in many aspects for this type of equipment. A much more suitable interview method was employed in discussing *typical* (or hypothetical) *cases* (introspection). This approach contributed much more in covering as many aspects as possible of the domain.

5.3. Development Strategy. The specification of ALF-A began as soon as an initial conceptual domain model was established. A prototype version of the KA tool was developed and demonstrated to the experts after about half of the time devoted to knowledge acquisition had been used. The prototype contributed to subsequent knowledge acquisition through feedback from the experts and through improving the understanding of KA tool support among the experts. Although only one pilot version was developed, an incremental approach was taken in the sense that successive versions of the system were specified, generated, and debugged (by the knowledge engineer) during the development process.

The first part of ALF-A to be specified involved the knowledge editors, i.e., the user interface. When all parts of the user interface were specified and the specification had stabilized, the internal knowledge representation in the emerging KA tool was defined. As a last step—when the user interface and internal representation were debugged—the transformation into knowledge bases was specified. This approach made it easier to adapt to changes in the emerging conceptual domain model and postpone design decisions regarding internal representations as well as knowledge base structures. Table I shows a breakdown of the various architectural components in the ALF-A specification.

5.4. Additional Components. A number of desirable functions in ALF-A could not be specified directly in DOTS. However, DOTS provides facilities for defining external knowledge editor and representation types. Certain components of ALF-A had to be implemented by conventional programming in Common Lisp and CLOS (*24*). This included specialized pop-up menus for references to the relevant technical documentation and special link facilities for the fault tree editor (relation grapher). About 500 lines of Lisp definitions and code were written manually and declared external to DOTS by the knowledge engineer.

5.5. Implementation Issues. As shown in Table I, the development time for ALF-A was 2 person–months, whereas P10 (which can be regarded as a KA tool of similar size) required 6 person–months of manual implementation in Lisp (*6*, *8*). In this context it is important to remember that DOTS was still undergoing debugging and testing during the ALF-A project and that P10 adopted the conceptual domain model supported from a previous project.

Initially, Xerox 1186 was used as a platform for DOTS and ALF-A. Insufficient CPU power and primary memory for this task resulted in performance deficiencies. The shift to Sun SPARCstations (SPARCstation is a trademark of Sun Microsystems) greatly improved the speed and removed many of the performance deficiencies faced in the ALF-A project.

In the current version of DOTS, it takes about 3 min and 15 s to generate the complete ALF-A system from its specifications. (This figure refers to Medley running on Sun SPARCstation 1.) The result is about 2600 lines of definitions and Lisp code intended to run together with the DOTS library. However, loading generated versions of ALF-A and starting them can take much longer time (up to 20 min).

6. RELATED WORK

TDE (*13*, *14*) is a tool that enables knowledge engineers and trained experts to edit *failure-modes*, i.e., a representation of the derivation of units under test and their standard (correct) behavior. TDE is specialized to TEST (*15*), a domain-independent shell for the troubleshooting task.

OPAL (*18*) is a graphical knowledge editor for cancer treatment protocols. Physicians can use OPAL to enter plans to a cancer-therapy expert system known as ONCOCIN (*23*).

P10 (*6*) is a KA tool that acquires planning knowledge from experts in protein-purification planning. The KA tool is tailored to the domain of liquid chromatography techniques. P10 allows its users to edit partial plans (skeletal plans) and generates knowledge bases from them.

KAVE (*21*) is a KA tool that acquires rules for an advisory system in the domain of ventilation therapy. Experts enter their knowledge by completing rule schemata (i.e., forms). Both P10 and KAVE are domain-specific, whereas TDE is more general. ALF-A can be regarded as quite domain-specific.

7. SUMMARY AND CONCLUSIONS

ALF-A is a specialized KA tool for troubleshooting laboratory equipment, in particular, DNA sequencers. ALF-A is intended to be used by instrument specialists to enter and edit their knowledge according to a conceptual domain model of troubleshooting. From the structures entered, ALF-A generates knowledge bases for use in expert systems.

Although ALF-A was originally intended to support a particular set of equipment (the A.L.F. DNA sequencer), it may be used for other types of devices that involve chemistry, hardware, and software. Since ALF-A is implemented in a meta-tool, it can easily be tailored to specific requirements. DOTS has proven to be a useful tool (shell) for developing a domain-specific KA tool for troubleshooting the A.L.F. DNA Sequencer, and DOTS has significantly improved the development speed for ALF-A.

Implementing a KA tool similar to ALF-A (or reimplementing ALF-A) would require considerably less effort in future projects. Given an improved version of DOTS, a knowledge engineer trained in meta-tool usage, and sufficient expert time, it should be possible to cut down the development effort even further. Even though ALF-A has been demonstrated to a number of domain experts and some state of consensus has been reached, there still remains much to be done on the validation of both the conceptual domain model and its implementation in the KA tool.

ACKNOWLEDGMENT

Domain expertise was provided by Ingela Berglund, Margret Bywater-Ekegärd, Robert Inglis, Anders Olsson, Carl Pereswetoff-Morath, and Lesley Perrin at Pharmacia Biosystems AB.

REFERENCES AND NOTES

(1) Boose, John H. A knowledge acquisition program for expert systems based on personal construct psychology. *Int. J. Man-Mach. Stud.* **1985**, *23* (5), 495–525.
(2) Boose, John H. A survey of knowledge acquisition techniques and tools. *Knowl. Acq.* **1989**, *1*(1), 3–37.
(3) Bradley, J.; Harbison-Briggs, K. The symptom-component approach to knowledge acquisition. *SIGART Newsl.* **1989**, *108*, 70–76.
(4) Davis, Randall. Interactive transfer of expertise: Acquisition of new inference rules. *Artif. Intell.* **1979**, *12* (2), 121–157.
(5) Epitec AB, Linköping, Sweden. *Epitool Development Environment Reference Manual*, 1989; Version 4.
(6) Eriksson, Henrik. Domain-Oriented Knowledge-Acquisition Tool for Protein Purification Planning. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 90–95.
(7) Eriksson, Henrik. Meta-tool support for customized domain-oriented knowledge acquisition. In *Proceedings of the Fifth Banff Knowledge Acquisition for knowledge-Based Systems Workshop*, Banff, Canada, Nov 1990; Boose, John H., Gaines, Brian R., Eds.; pp 6.1–6.20.
(8) Eriksson, Henrik. Specialized knowledge acquisition tool support compared to manual development—a case study. In *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications*, Miami, FL, Feb 1991.
(9) Freeman, Mark; Baehler, Carla; Spots, Steven. *Automated* laser-fluorescence sequencing. *Bio/Technology* **1990**, *8*, 147–148.
(10) Freiling, Mike; Alexander, Jim; Messick, Steve; Rehfuss, Steve; Shulman, Sherri. Starting a knowledge engineering project: A step-by-step approach. *AI Mag.* **1985**, *6*(3), 150–164.
(11) Gammack, John G. Different techniques and different aspects on declarative knowledge. In *Knowledge Acquisition for Expert Systems: A Practical Handbook*; Kidd, Alison, L., Ed.; Plenum Press: New York, 1987; Chapter 7.
(12) Ginsberg, Allen; Weiss, Sholom. SEEK2: a generalized approach to automatic knowledge base refinement. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI'85*; 1985; pp 367–374.
(13) Kahn, Gary S. From application shell to knowledge acquisition system. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence, IJCAI'87*; 1987, pp 355–358.
(14) Kahn, Gary S.; Breaux, Edwin H.; DeKleer, Peter; Joseph, Robert L. A mixed-initiative workbench for knowledge acquisition. *Int. J. Man-Mach. Stud.* **1987**, *27*(2), 167–179.
(15) Kahn, Gary S.; Kepner, Al; Poppner, Jeff. TEST: a model-driven application shell. *Proc. AAAI-87*, **1987**, 814–818.
(16) Marcus, Sandra; McDermott, John. SALT: a knowledge acquisition language for propose-and-revise systems. *Artif. Intell.* **1989**, *39* (1), 1–37.
(17) Musen, Mark A. Conceptual models of interactive knowledge acquisition tools. *Know. Acq.* **1989**, *1*(1), 73–88.

(18) Musen, Mark A.; Fagan, Lawrence M.; Combs, David M.; Shortliffe, Edward H. Use of a domain model to drive an interactive knowledge-editing tool. *Int. J. Man-Mach. Stud.* **1987**, *26*(1), 105–121.
(19) Neale, Ian M. First-generation expert systems: A review of knowledge acquisition methodologies. *Knowl. Eng. Rev.* **1988**, *3*(2), 105–145.
(20) Pharmacia Biosystems AB, Sweden. *A.L.F. DNA Sequencer Users Manual*, 1990.
(21) Shahsavar, Nosrat. *Knowledge Acquisition and Refinement for a Domain-Specific Expert System.* Linköping Studies in Science and Technology, Licentiate Thesis 229. Linköping University, 1990.
(22) Shaw, M. L. G.; Gaines, B. R. KITTEN: knowledge initiation and transfer tools for experts and novices. *Int. J. Man-Mach. Stud.* **1987**, *27*(3), 251–280.
(23) Shortliffe, Edward H.; Scott, A. Carlisle; Bischoff, Miriam B.; Campbell, A. Bruce; van Melle, William; Jacobs, Charlotte D. ONCOCIN: an expert system for oncology protocol management. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence, IJCAI'81*; 1981; pp 876–881.
(24) Steele, Guy L., Jr. *Common LISP: The Language*, 2nd ed.; Digital Press: 1990.
(25) Wielinga, B. J.; Schreiber, A. Th.; Breuker, J. A. KADS: a modelling approach to knowledge engineering. *Knowl. Acq.*, in press.

UTAB: A Computer Database on Residues of Xenobiotic Organic Chemicals and Heavy Metals in Plants

J. E. NELLESSEN and J. S. FLETCHER*

Department of Botany and Microbiology, University of Oklahoma, Norman, Oklahoma 73019-0245

Received September 30, 1991

The UTAB Database contains information concerned with the uptake/accumulation, translocation, adhesion, and biotransformation of both xenobiotic organic chemicals and heavy metals by vascular plants. UTAB can be used to estimate the accumulation of chemicals in vegetation and their subsequent movement through the food chain. The database contains actual data from papers in the published literature dating from 1926 for organic chemicals and from 1976 for heavy metals. At present the database is comprised of more than 37 000 records pertaining to 900 different organic chemicals, 21 heavy metals, and over 350 plant species. Each record contains information on a single combination of species, chemical, and dose. Other information includes the application and destination sites, amount accumulated, rates of uptake or translocation, products and sites of biotransformation, experimental condition parameters, and the source paper. Thus, the database can be used to quickly obtain specific data pertaining to a chemical, plant species, mine spoil, etc. or it can be used for the comparative analysis of a set of data pertaining to groups of chemicals and plants.

INTRODUCTION

The accumulation of toxic chemical residues in the environment has been a well-publicized concern since the 1960's.[1] Pesticides used in agriculture, metals from sewage sludge, mining and refinery operations, and miscellaneous other chemicals from industrial processes such as PCBs (polychlorinated biphenyls) and PAHs (polycyclic aromatic hydrocarbons) may all leave toxic residues in the environment. Food chain contamination with insecticides such as DDT, resulting in deleterious effects on predatory birds, are well known.[1] In the U.S. 631 000 tons of pesticides were produced in 1989.[2] After exports and imports 416 000 tons remained for usage in the U.S. with approximately 55% as herbicides, 35% as insecticides, and 10% as fungicides.[2] Some authors suggest that a considerable quantity of these applied pesticides do not reach target organisms and may thus impinge upon surrounding ecosystems.[3] In the U.S. with the development of regulatory policies and agencies, many of the most toxic and persistent chemicals have been banned. But new chemicals are continually being developed, tested, and used. Metals in sewage sludge,[4] lead along roadsides,[5] and more recently selenium poisoning in birds[6] and its implications for human health[7] have all been topics of research. Cadmium is a toxic trace metal often present in sewage sludge, but its presence as a contaminant in phosphate fertilizers may be far greater.[8]

Since plants play an essential role in the ecosystem as primary producers by extracting resources from the abiotic environment, their potential involvment in the fate of xenobiotic chemicals in the ecosystem cannot be ignored. Plant roots may draw metals and organic contaminants out of the soil and translocate them through the stem to be accumulated in the foliage. Animals feeding on the foliage will thus become exposed. Plant species can differ in their ability to extract compounds and elements from the soil and transport them to the foliage. For example, in the southwest U.S. certain plant species in the genus *Astragalus* (milkvetch) are selenium accumulators[9] and contain enough selenium to be toxic to cattle.[10] Although selenium is an essential nutrient to animals[11] (required in very small quantities) and deficiencies are often reported,[12] certain plant species hyperaccumulate selenium and are toxic. Consideration of such differences between plant species in their tendencies to accumulate or not accumulate certain chemicals is very important when it comes to the restoration of contaminated ecosystems.

Data on chemical residues, whether toxic or not, are continually being published in a wide variety of journals. With the ever-increasing volume of published residue data it becomes increasingly hard to benefit from this pool of data when it is so widely scattered. It has been our goal to assemble this vast array of data into a computerized database called UTAB. UTAB stands for the uptake/accumulation, translocation, adhesion, and biotransformation of both organic chemicals and metals by vascular plants.

The UTAB Database is a computerized information resource that permits the rapid retrieval and comparison of chemical residue data. UTAB can be used to estimate the accumulation of applied chemicals and their biotransformed products in vegetation and the consequent exposure to wildlife pertaining to problems of food chain contamination. Data pertaining to a specific chemical, plant species, or process (e.g., uptake, translocation, etc.) can be obtained from the database. UTAB thus serves as a rapid means of accessing numerical data in minutes with no need to acquire and read the published papers from which the database was derived.

Crop, weed, native, and wild plant species exposed to metals or xenobiotic organic compounds have been considered.