Title page

Title : E-Decisions-on-Tap : Decision Support E-Services

Author names : Claus Skaanning and Janice Bogorad

Company : Hewlett-Packard

Address : Kongevejen 25, DK-3460 Birkerød, Denmark

Telephone number : +45 45991090

Fax number : +45 45991957

E-mail address : claus_skaanning@hp.com

E-Decisions-on-Tap: Decision Support E-Services

Background

Among the key challenges faced by those attempting to capitalize on the promise of e-services is the ability to enable agents and objects to reach correct decisions under conditions of uncertainty.

The transparent interoperation of e-services on behalf of customers often faces a critical problem. When processes must reach decisions under conditions of uncertainty, either the transparency is abandoned by relying on human interaction, or expert human decision-making capabilities must be somehow provisioned – with an immediate thought to look towards artificial intelligence technology. However, most artificial intelligence technologies (e.g. neural networks) are not exceptionally good at explaining the logic and interactions behind their processes, nor are they good at extracting information from their processes or conclusions. They also come up short in dealing with "real life" situations involving uncertainty, or missing or ambiguous data.

Hewlett-Packard has been working extensively with Bayesian network technology, which successfully addresses these issues. Our initial successes with the technology have been in applying it to troubleshooting complex devices. In the near future we will be using it for general decision support.

In this paper we will discuss our results to date and we will explore the opportunities for the future.

Breaking through the barriers of using Bayesian Networks

For several years Bayesian networks have been an integral part of some of the most intelligent services to be delegated to computers. They have been used in powerful search engines, safety assessment systems, guidance systems for autonomous vehicles, and automated medical diagnostic systems. The key advantage of Bayesian network technology is the ability to incorporate experience and uncertainty into the reasoning process, making it possible for computers to emulate human experts.

In the past, Bayesian network technology has seen limited practical deployment due to the background and expertise required for the construction of Bayesian network models. Thus, the technology had been restricted to a few point products and expensive systems, where this laborious and time-consuming process was justified by the benefits of automated, human-like reasoning. We wanted to break through this barrier, and to bring the power of this technology to our customers - to make it possible to easily and effectively build useful decision support systems. We also wanted to "de-mystify" this powerful technology, thus ensuring its broader application and commensurate benefit to all.

Hewlett-Packard entered a joint research project with the Decision Support Systems group (DSS) at Aalborg University in Denmark, which is regarded as one of the international leaders within the area of Bayesian networks. These activities comprise the Hewlett-Packard Laboratory for Normative Systems, partially funded by both Hewlett-Packard and the Danish Center for IT Research (CIT). CIT is an organization under the Danish Ministry of Research, which focuses on encouraging mutually beneficial projects between academia and industry. The purpose here was to join the strong Bayesian network research capabilities at Aalborg University together with HP's R&D forces, experience and strength in the area of customer support. The results of this partnership have been stunning, and have led to applications well beyond the area of services.

Bayesian networks are models of knowledge held by *experts* in some field or *domain*. We chose the *domain* of printing system support as a test environment for this research, and pulled together a team consisting of printer support *domain experts* and Bayesian network experts. As a team, we used considerable time and effort working through an iterative process of building a methodology for the construction of the Bayesian network models and an effective diagnostic engine, while actually developing some troubleshooting engines and models.

Initially, a Bayesian network expert created the troubleshooting models, based on lengthy and detailed (and sometimes heated) discussions with the *domain experts*, as we worked through the most effective methodology for knowledge acquisition. As we continued the iterative process, together with our *domain experts*, we created software tools to allow the domain experts to directly define the troubleshooting models on their own. These software tools allow experts in some field to effectively and efficiently create useful decision support systems without prior background in or knowledge of Bayesian network technology. The development of the *authoring tool* represents a major breakthrough in the area of Bayesian network utilization; the intellectual property associated with this breakthrough is protected under pending patents.

Bayesian Automated Troubleshooting System (*HP BATS*)

These software tools have been further developed, and now comprise the Bayesian Automated Troubleshooting System (*HP BATS*), including the *HP BATS Troubleshooter* and the *HP BATS Author*. These software tools, along with some of the underlying technology, will be described in some detail later in this paper. We will also explain the next step: making this technology broadly available and easy to use as e-services, called e-Decisions-on-Tap (e-DOT).

The *HP BATS Troubleshooter* guides the user through the optimum troubleshooting sequence, helping to resolve a problem in as few steps as possible – even if the user has a minimal understanding of the subject matter related to the troubleshooting activity. This is done by taking into account various factors such as: the most likely cause of the problem; the time, difficulty, or risk involved for each step; how much information can be gathered from each step; etc. At each step the troubleshooter adjusts its various internal variables, and its advice, dependant on the answers to the questions and results of all previous troubleshooting steps, as well as its inherent knowledge of the system.

The *HP BATS Troubleshooter* employs the Hugin Expert (Hugin) Bayesian network inference engine for its probabilistic calculations when computing the best next step. The Hugin engine is considered the leading inference engine of its type due to its speed of execution and scalability.

The Hugin Expert company also makes available an authoring tool for construction of Bayesian networks. While the *HP BATS Author* is a tool for construction of constrained Bayesian networks useful for troubleshooting, the Hugin authoring tool can be used for construction of general Bayesian networks. Due to the focus on troubleshooting and diagnosis, it was possible to make the *HP BATS Author* useful to people with no knowledge of Bayesian networks, enabling the specification using terms and knowledge from the troubleshooting environment exclusively.

Both the Hugin engine and the Hugin authoring system will become available as e-services through e-Decisions-on-Tap, to be named Hugin-on-Tap. HP is a major stockholder in the Hugin Expert company.

The *HP BATS Author* allows domain experts to create automated troubleshooting systems. Domain experts specify a model for a problem using intuitive terms that they are familiar with, such as causes, actions and questions – without having any knowledge of Bayesian networks. Due to the way Bayesian networks are assembled by the *HP BATS Author*, they are guaranteed to be of linear complexity, allowing very large and complex troubleshooters with the *HP BATS* methodology.

In an initial test of the *HP BATS* system, focused on troubleshooting an HP LaserJet printer, we found it faster to create more powerful automated troubleshooting systems than using other technologies (e.g. case-based reasoning). We also experienced instances where the authors had questioned a troubleshooting sequence that had been recommended by the *HP BATS* system, since it did not follow their own standard procedures - until further investigation proved that the system's recommendations were actually much more efficient.

We will now describe the next development, bringing the power of *HP BATS* to a much broader and more flexible base in the form of e-services. A technical description of the *HP BATS* system is included late in this paper.

e-Decisions-on-Tap (e-DOT)

e-Decisions-on-Tap (e-DOT) will provide one of the essential ingredients for achieving the vision of services working together transparently on the Internet: the ability of these services to reason about preferences, tasks, and circumstances as an expert would - by considering past experience, current context, and available knowledge. e-DOT will provide an asset on the Internet that e-services can consult to make decisions and solve problems. Without this ingredient, most e-services may rely heavily on human intervention, and be relegated to very specialized or trivial tasks.

e-Decisions-on-Tap is an e-service deployed on the Internet which allows companies to easily create smart services for their customers.

Authors will be guided through the process of capturing expertise, experience and knowledge on the topic of their choice, into *Bayesian network models*. They may then choose to keep these *models* available to a select few through websites or e-services, or they may choose to publish them for broader exposure and usage. If granted access by the owner, others will then be able to add their knowledge to the model through a virtual library of published knowledge.

The e-Decisions-on-Tap infrastructure will provide the necessary computational backend for users - all they need is an Internet connection. Further, lightweight services in devices will be able to call upon e-DOT to solve problems or make decisions, just by speaking XML or by providing a browser to a user. One example of this could be a car consulting with e-DOT to provide the driver with mechanical advice when something is malfunctioning. Another example could be a customer receiving an expert pre-sales consultation session from e-DOT in the middle of the night - when no human sales staff is on duty. e-DOT will be able to make decisions and solve a wide range of problems that previously required human assistance.

Users of e-DOT will be able to create smart services through an intuitive interface, building from a library of Bayesian network models. They will have secure access to their environment and will be able to manage both their models and who has access to them.

The initial e-DOT offering will include HP's Bayesian Automated Troubleshooting System (*HP BATS*), which will be available to allow support providers to automate diagnosis and troubleshooting of customer systems. They may choose to completely automate this task by allowing a set of diagnostic e-services to inspect a customer system and consult with e-DOT to provide a solution, or they may choose to have e-DOT interact with a customer through friendly web pages with simple questions and suggestions. A technical overview of *HP BATS* is given later in this paper.

Initially, through *HP BATS*, e-DOT will provide end users with rapid, efficient and consistent problem correction through its *Intelligent Troubleshooting* e-service, and it will arm support providers with an automated means to address high labor costs associated with the complex problem space. The e-DOT *Knowledge Acquisition* lifecycle services hide the Bayesian complexities and allow domain experts to easily build, validate, deploy, maintain and extend automated diagnostics for complex problems that include a high degree of uncertainty. The solutions are consistently correct, and can be automatically refined based on experience. Historical data also provides feedback for product improvement.

We will now outline the initial offering of e-Decisions-on-Tap, followed by an explanation ofthe architecture and future directions. We will conclude the paper with a technical description of *HP BATS*.

e-Decisions-on-Tap Initial Release

As mentioned previously, the initial e-DOT offering is being created using HP's Bayesian Automated Troubleshooting System (*HP BATS*) as a base technology. e-DOT will subsequently be expanded in scope and functionality. The first two sets of e-services to be made available will be:

- ***Knowledge Acquisition*** services: to manage the full lifecycle of models that support the intelligent decision-making process.

- ***Intelligent Troubleshooting*** services to guide a user through the most efficient troubleshooting sequence.

These services will be provided using flexible payment schemes, including pay-per-use and subscription billing.

The Intelligent Decision Troubleshooter services will allow existing and emerging support providers to automate diagnosis and troubleshooting of customer systems. They may choose to completely automate this task by allowing an e-service to inspect a customer system and consult with e-DOT to provide a solution, or they may choose to have e-DOT interact with a customer through intuitive web pages with simple questions and suggestions.

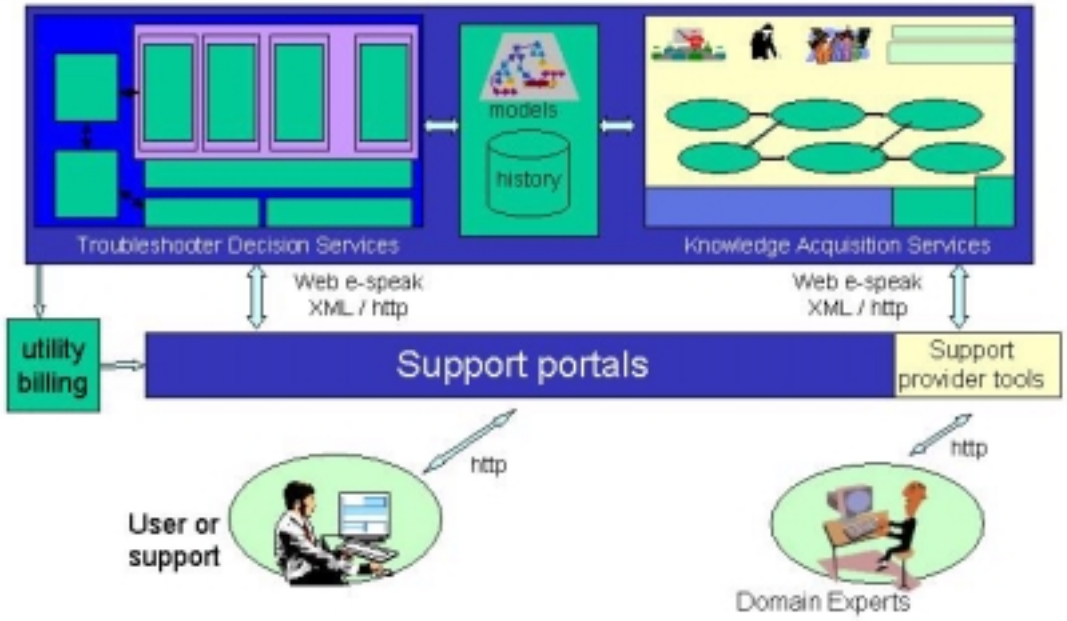

Figure 1 Overview of e-Decisions-on-Tap

Knowledge Acquisition E-services

Bayesian networks are models of knowledge held by experts in some field or domain. Knowledge acquisition from domain experts is essential to any project involving modeling of a domain. Efficient creation and maintenance of the Bayesian network models is the key to effective and successful employment of this technology in business environments. The results of our research and development work in this area, captured in the *HP BATS Author* desktop software tool, have been the technology base for our knowledge acquisition e-services. The Knowledge Acquisition e-services will allow domain experts to collaboratively build models without understanding the underlying Bayesian networks.

Domain experts will use a guided lifecycle implementation, with the first stage being the creation of models, where they will:

- List possible causes for the problems to be resolved
- Supply the required troubleshooting steps for resolution: actions and questions
- Provide “costs” for these troubleshooting steps: time, difficulty, risk, financial costs, etc.

- Provide “probabilities” of likelihood: that a cause occurs, that a step will solve the problem, for the step to be performed accurately, etc.

Model Libraries

Causes and troubleshooting steps can be stored in reusable *Model Libraries* to greatly facilitate and accelerate the creation of models. Model libraries can be used in a single environment, within a team of *domain experts*, or they can be “published” externally. Published model libraries open opportunities in both directions: for *domain experts* to be able to leverage the expertise and work of others, and for the library authors to receive revenue in the form of royalties from their models being used by others.

A detailed account of the definition of causes and steps, as well as validation of these models and the use of model libraries, can be found later in this paper.

Life Cycle Management - Collaboration

The full life cycle for models will be supported in the Knowledge Acquisition e-services: creation, testing, validation, deployment, refinement, maintenance, adaptation and reporting. (The technology being used for validation is covered in the *HP BATS* section of this paper.)

Collaboration for teams of *domain experts* to work together on the creation of models and model libraries will also be supported. To allow for the possibility of physically disbursed teams of *domain experts*, Hewlett-Packard’s **e-Learning-on-Tap** technology for distributed collaboration will be employed.

Learning through Experience

e-DOT will be able to learn and adapt its models through experience. The intelligent troubleshooter e-services will store historic data concerning the use of models, and actual troubleshooting sequences being employed. This historic data can then be “fed back” into the models, adapting the original parameters specified by the *domain experts* to better reflect actual experiences. This will allow the models to adjust themselves to better reflect the “real world” in two ways: to give a more accurate picture than estimated, and to adjust as situations change as time goes by.

Valuable information will be captured in the historic data log of the various troubleshooting sequences, and data captured. This historic data will be made available for reporting purposes, such as identifying the most common problems with a product and their root causes.

Intelligent Troubleshooting E-services

The results of our research and development for diagnostic engines using Bayesian network technology, captured in the *HP BATS Troubleshooter*, have been the technology base for the knowledge acquisition e-services under development. The Knowledge Acquisition e-services will allow domain experts to collaboratively build models without understanding the underlying Bayesian networks.

The ***Intelligent Troubleshooting*** e-services guides a user through the most efficient troubleshooting sequence considering:

- The most likely cause of the problem
- The “cost” of performing a step such as time, difficulty, risk, financial cost, etc.
- The value of information to be gathered at each step.

The services adjust their advice at each step based on answers from previous steps, with correct and full representation of all uncertainty. The users (either the end-user or support personnel) access the services via a browser connection to a support provider.

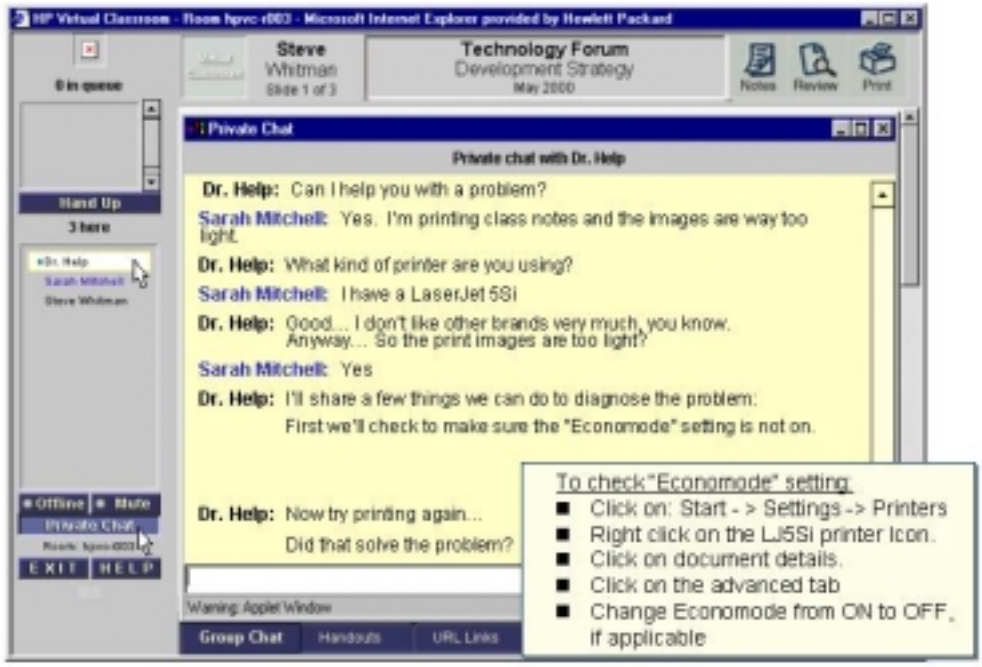

Figure 2 Example e-DOT troubleshooting transcript.

Automatic Data Acquisition

Another strength of Bayesian technology is the ability to incorporate data from the environment into any section of the model at any time. This provides a powerful potential - merging remote diagnostic data with the intelligence of troubleshooting sequences. Troubleshooting models can be populated by data collected by remote diagnostics, making a portion of the interactive troubleshooting questions unnecessary.

A test prototype was constructed to portray this potential, using a remote printer diagnostic tool and the *HP BATS troubleshooter* using a printing system model. In that particular test case, most of the questions that the customer would have been asked were gathered directly from the remote diagnostic, allowing the problem to be solved with a single step in many cases.

e-Decisions-on-Tap Architecture

The e-Decisions-on-Tap system architecture has to be able to support thousands of clients simultaneously using e-DOT services. The basic elements of the e-DOT system are the *models* and the *decision engines*. Models represent knowledge captured from domain experts, describing a problem area where e-Decisions-on-Tap can be of some assistance. Decision engines are compute engines that can load any model and execute it. Thus, a decision engine can be compared with a computer, and the model can be compared with the software.

One of the central portions of the e-DOT architecture is the *engine management*, which keeps a large number of active and passive engines running at all times. The active engines should be load-balanced

to ensure users satisfactory response times during execution, and the passive engines should be preloaded with models to ensure that new users very rarely experience delays starting up. Another central portion is *model version management,* which ensures management of multiple versions of the models, including staging the new versions as they become available. A third component is the *model broker*, which assists the user in selecting the most appropriate model for her problem. There are numerous ways to accomplish this task. The initial e-DOT offering will support browsable and searchable model hierarchies, with more sophisticated and automated facilities to follow.

Future Directions for e-Decisions-on-Tap

e-Decisions-on-Tap will not only use Hewlett-Packard technologies such as the *HP BATS* system for knowledge acquisition and troubleshooting; it will be expanded to include decision support technologies from other world class providers. As an example, Hugin Expert will participate in the e-DOT infrastructure with their tools for construction and execution of Bayesian networks. Technologies from paradigms other than Bayesian networks will also find a place in the e-DOT infrastructure, e.g., neural networks, genetic algorithms, etc.

e-Decisions-on-Tap will incorporate all the technologies required to provide general decision support in many complex domains, and intuitive graphical user interfaces for construction of the underlying models, whether the model is an *HP BATS Troubleshooter*, a Hugin Bayesian network, or something completely different.

e-DOT services will contain two basic elements, *knowledge acquisition services* and *decision support services.*

The knowledge acquisition services consist of:

- Building new decision models using tools such as the *HP BATS Author* or the Hugin Expert system shell. These tools will allow collaborative construction of models.
- Learning new decision models from existing data, or partial learning of models to be completed using an authoring tool.
- Testing and validation of decision models using a tool such as the *HP BATS* validation system.
- Maintaining models by either refining them manually or using automated adaptation systems for fine-tuning parameters.
- Deployment of models, and model version management

The decision support service involves actually employing the decision models:

- Matching the user problem to the appropriate decision model through menu structures, browsable model category structures, text searches, or a more sophisticated search and selection process – in some causes automated, possibly through the use of higher level models or diagnostic data collected directly from the environment.
- Using the decision model to make some decision or solve some problem.

HP BATS Technical Description

The development of tools for the construction of Bayesian networks at Aalborg University led to the creation of a company, Hugin Expert A/S, which now sells and markets a world-leading software tool. Hewlett-Packard is a major stockholder in Hugin Expert A/S. The Hugin engine for efficient inference in Bayesian networks is embedded in the *HP BATS* troubleshooting engine.

Traditionally, the time and expertise required to build Bayesian network models had been a bottleneck to the application of this powerful technology in practice. *HP BATS* has addressed and overcome this barrier, through the creation of tools to efficiently build automated troubleshooting systems, with no knowledge of the underlying technology, or even experience in the methodology. It takes less than a couple of hours to learn all the basics of using *HP BATS* for constructing troubleshooters. After some practical use, and particularly for more complex environments and more advanced functionality, additional follow-up training of a couple of hours is advisable. Once the process is familiar, simple troubleshooting systems can be constructed in a matter of few hours, while more comprehensive and complex systems (e.g. an operating system or complex software application) could be created in a matter of a few man-months.

The *HP BATS* tools utilize Bayesian networks for representing uncertainty in complex troubleshooting problem domains. Bayesian networks provide a way of modeling problem areas using probability theory. The Bayesian network representation of the problem can then be used to provide information on some variables given information on others. A Bayesian network consists of a set of variables (nodes) and a set of directed arcs connecting the variables. Each variable has a set of mutually exclusive states. The variables, together with the directed arcs, form a directed acyclic graph (DAG). For each variable v with parents $w_1,...,w_n$, there is specified a conditional probability table $P(v|w_1,...,w_n)$. If v has no parents, this table reduces to a marginal probability distribution $P(v)$. For further introduction to Bayesian networks the reader is referred to [4] and [1].

Bayesian networks have been used for many application domains which deal with uncertainty, such as medical diagnosis, pedigree analysis, planning, debt detection, bottleneck detection, etc. However, the major application area has been diagnosis, which lends itself very well to the modeling techniques of Bayesian networks, i.e., underlying factors that cause diseases or malfunctions that cause symptoms.

The currently most efficient method for exact belief updating in Bayesian networks is the junction-tree method [5] that transforms the network into a so-called junction tree. The junction tree basically clusters the variables such that a tree is obtained (i.e., all loops are removed) and the clusters are as small as possible. In this tree, a message passing scheme can then update the beliefs of all unobserved variables given the observed variables. This scheme is used in the Hugin inference engine which is utilized by the *HP BATS* troubleshooting logic for determining the best next step. For a detailed description of construction of and inference in junction trees, see [4].

In the following sections we will describe the theory and ideas underlying the *HP BATS* technology. For more detailed descriptions of the theory see [6] and [7].

A Sample Application : Troubleshooting of e-LOT

We have chosen to present a troubleshooter for a commonly occurring problem when using the e-Learning-on-Tap service accessible from the Internet at http://e-learning.hp.com/ : the problem of getting access to the service. Numerous settings and errors can cause this problem such as the presence of firewalls, invalid access keys, network problems, problems on the client PC, and whether Java is enabled in the browser. This is a rather complex problem causing much work for the e-LOT support people. To remedy this problem, the e-LOT team has constructed an automated troubleshooter using the *HP BATS Author* that will be deployed in the e-Decisions-on-Tap system.

A sample troubleshooting sequence for this troubleshooter might be as follows with a web browser or computer asking the questions, and a customer supplying the answers :

1. Do you successfully enter the preloading phase of the HP Virtual Classroom? -- YES
2. Ensure that Java is enabled in your web browser. Does that help? -- NO
3. Can you access the HP Virtual Classroom from another machine on the same network? -- YES
   (Answering YES here indicates a problem on the local machine)
4. What web browser are you using? -- Internet Explorer
5. Can you access another Java-enabled web site? -- NO
   (Answering NO here indicates a problem with the local Java system / setup)
6. What operating system are you using? -- Microsoft Windows (any)
7. Do you successfully complete the preloading phase of the HP Virtual Classroom? -- NO
8. What is your Internet Explorer security zone setting? -- LOW
9. Ensure that your Internet Explorer browser does not reject signed code. Does that help? -- NO
10. Install or replace the imagehlp.dll file. Does that help? -- YES

The problem was caused by a corrupted or missing imagehlp.dll file. A relatively long and complex troubleshooting sequence was required to determine this problem. The *HP BATS Troubleshooter* attempts to optimize the steps such that the user is given the shortest possible sequence to solving his problem. In follow-up research at Aalborg University it was shown that the sequences generated by *HP BATS* were very close to the optimal ones. The complexity of finding the optimal sequence is exponential, so it has only been possible to do the comparison for smaller troubleshooters.

With the above solution, an e-service is used to solve problems with another e-service, resulting in very low labor costs and a support solution that is available to the customer 24 hours 7 days a week providing high quality assistance with both easy and complex problems.

Troubleshooting building blocks : causes and steps

If we want to troubleshooting a malfunctioning device with *HP BATS*, we represent the possible underlying causes, $F_1,...,F_n$ (F for fault), repair actions, A_1, ..., A_k, that can potentially solve the problem, and questions, Q_1, …, Q_m, that can potentially provide useful information about the causes. In a printing system domain, causes could, for instance, be the printer driver, the spooler, etc., and actions could be "Reinstall the printer driver", "Try another printer parallel cable", etc. Questions could be "Does the control panel say TONER LOW?", etc. Each repair action A_i has a probability P_i of solving the problem and a cost C_i. The cost may be combined from several cost factors such as the time it takes to carry out the action, money required to buy requisites, etc.

The measure of efficiency for a troubleshooting sequence is the *expected cost of repair*, ECR. That is, if A_1, ..., A_k is a sequence, then sometimes the sequence is stopped after *i* steps with a certain total cost, and sometimes after *j* steps with another total cost. The expected cost of repair is the mean value of the cost until the sequence stops.

The value P_i/C_i is the efficiency of the action A_i. Under certain assumptions, if

$$\frac{P_i}{C_i} > \frac{P_j}{C_j} \qquad \textbf{(2)}$$

it will be best to perform action A_i first. Eq. (2) is true if there is only one fault (the single fault assumption), and each action can at most discover one cause. Under these conditions, the optimal sequence of actions can be found by ordering all actions with respect to their efficiencies. This sequence will, on the average, yield the lowest possible ECR.

Troubleshooting sequences under these conditions are discussed further in [3]. If these assumptions are satisfied, it is straightforward to show that the expected cost of repair for an optimal sequence A_1, ..., A_k, is

$$ECR = C_1 + (1-P_1)C_2 + (1-P_1-P_2)C_3 + ... + (1-\sum_{i=1}^{k-1} P_i)C_k \qquad \textbf{(3)}$$

In many real-world domains, the single-fault assumption is fulfilled. However, it is rarely true that actions can discover only a single cause, or at least it is very awkward for domain experts to structure

the information under these constraints. If domain experts are allowed to specify actions that can discover multiple causes, it is not possible to find the optimal sequence by sorting the actions with respect to efficiency, and the expected cost of repair can not be computed with Eq. (3). Approximate algorithms for finding a good sequence and computing its ECR are presented in the following sections.

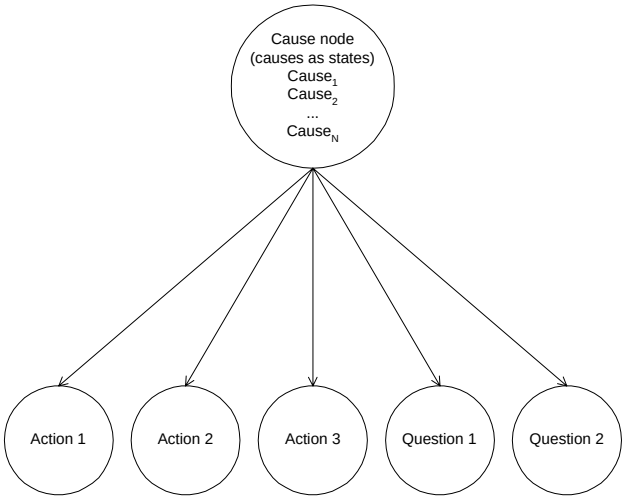

Figure 3 An example of the very simple Bayesian network structure used for troubleshooters.

The computation of the optimal sequence of actions only tells us which action it is best to perform first, but we need to check whether it is best to start out with a question. To determine whether it is best to ask question i first, we compare ECR of the optimal sequence with ECO_i :

$$ECO_i = C_i + \sum_{Q_i=s} P(Q_i = s) \times ECR(Q_i = s)$$

where C_i is the cost of finding an answer to the question, s is an answer to the question, and $ECR(Q_i=s)$ is the expected cost of the optimal sequence given that Q_i has been answered with s.

The efficiency and simplicity of the representation and algorithms described in this paper depend on the assumption that there is only a single fault, i.e., the single-fault assumption. The single-fault assumption seems natural in many domains where malfunctioning is discovered quickly. As soon as some component in the device stops functioning, the entire device will stop working properly and this will usually be discovered quickly. It is very rare that multiple components stop functioning at the same time. If multiple faults are present, the algorithms presented in this paper will also solve the problem, however, not in an optimal manner. However, we are investigating efficient algorithms for finding good sequences without the single-fault assumption.

All *HP BATS* models are represented as so-called naïve Bayes structures with a single parent node representing all the causes, having many child nodes representing the actions and questions. An example is shown in Figure 3. This simple structure is possible because of the single-fault assumption, and it further has the benefit that all actions and questions are independent given the causes. This leads to Bayesian networks with linear storage complexity, and efficient algorithms for finding the best next step.

In the following it will be described how causes, actions and questions are represented and how the required information is acquired from domain experts.

Causes

Causes are organized as states in the parent node of the naïve Bayes net, however, for the knowledge acquisition of cause probabilities, domain experts can organize the causes in a tree structure with the problem-defining node in the top, then causes as children, and potentially sub-causes of these as grandchildren. A small example of such a cause tree is given in Figure 4.

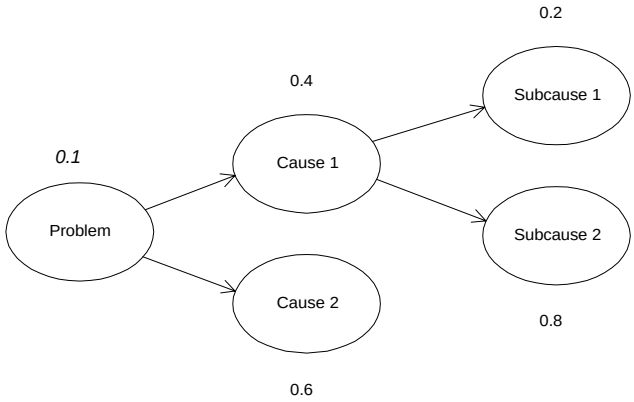

Figure 4. A simple Bayesian network with an example probability assignment.

Probabilities for causes are acquired based on the cause tree in the opposite of the causal direction, i.e., the domain experts specify probabilities for causes conditional on the presence of their parent cause. For the example in Figure 4, domain experts have to specify P($Cause_1$ | Problem), P($Cause_2$ | Problem), P($Subcause_1$ | $Cause_1$) and P($Subcause_2$ | $Cause_2$). An example specification of these probabilities can be seen in Figure 4. These probabilities can then easily be transformed to probabilities that can be used in the naïve Bayes net.

Actions

Actions are troubleshooting steps that, when carried out by the user, can potentially make the problem go away. There are two types of actions, *repair actions* and *test actions.* Repair actions (e.g., reseat the printer's parallel cable) can solve the problem and thus end the troubleshooting process whereas test actions (e.g., move the printer to another room and try printing) change the configuration of the system to test whether the problem goes away. A test action thus only provides information that can be used later in the troubleshooting process and the troubleshooting process continues no matter what the answer.

The knowledge acquisition for actions consists of three steps, (i) listing the causes that the action can discover, (ii) eliciting probabilities that the action discovers these causes, (iii) eliciting the cost factors of the action.

The assessment of the probability that an action discovers a cause has been split into three pieces, (i) P(A=yes | f, correct, requisites), the probability of the action solving the problem assuming that the cause f is present, the action is performed correctly and all requisites are in perfect order, (ii) P(correct), the probability that the action is performed correctly, and (iii) P(requisites), the probability that all requisites are in working order. The three probabilities are then combined into one :

$$P(A = \text{yes} \mid f) = P(A = \text{yes} \mid f, \text{correct}, \text{requisites}) \times P(\text{correct}) \times P(\text{requisites}) \qquad \textbf{(4)}$$

The probability elicitation is much easier for the domain expert if it is split into three pieces, as they do then not have to balance several factors simultaneously in their minds.

The following cost factors are elicited for troubleshooting steps, (i) time, (ii) risk - of breaking something else, (iii) money - required to carry out the step, (iv) insult[1] - potential insult to user if this step is suggested. The cost factors are combined linearly to form the overall cost of the step :

$$C = \alpha T + \beta R + \chi M + \delta I$$

Questions

Questions are troubleshooting steps that gather information that can be used in the troubleshooting process. There are two types of questions, *symptom questions* and *general questions.* *Symptom*

[1] In printer systems, steps such as "Check whether the printer is turned on" or "Check whether the parallel cable is plugged in" can be insulting to experienced users.

questions concern symptoms of effects of the problem, and *general questions* concern something that could have caused or created the problem.

For symptom questions, the domain experts must elicit the probabilities of the question answers given each associated cause, and probabilities of the answers given that none of the associated causes are present. Questions are represented as children of the cause variable. Symptom questions are easy to represent as the elicited probabilities can be used directly in the probability table of the node.

For general questions the domain experts must elicit the probabilities of the associated causes given the answer to the question, and the prior probabilities of the answers to the question. Thus, the direction of the arcs in the Bayesian network is reversed here. When eliciting these probabilities, the following equation must be maintained :

$$P(F) = \sum_{Q=s} P(F \mid Q = s)P(Q = s) \qquad \textbf{(5)}$$

Thus, the probabilities that must be specified here are very much interdependant.

General questions are also represented as children of the cause variable in the Bayesian network, however, to represent them this way we need to reverse the probabilities from P(F | Q) to P(Q | F). This can be done by applying Bayes' formula :

$$P(Q \mid F) = \frac{P(Q \mid F)P(F)}{P(Q)}$$

Bayes' formula is a general equation that allows you to reason with conditional probabilities, i.e., the probability of an event A conditional on an event B occurring is written as P(A | B). With Bayes' formula it is then possible to reverse the conditional probability to P(B | A). This is useful if domain experts find it easier to specify P(A | B), but P(B | A) is needed in the application.

Due to the single-fault assumption and assuming that questions are independent of each other given the causes, it can be shown that the probabilities in *HP BATS* can always be reversed using Bayes' formula.

Costs are elicited in the same way as for actions.

Algorithms for Troubleshooting

Under the single-fault assumption and a one-to-one correspondence between causes and actions, the optimal sequence of actions is found by sorting the actions in descending order with respect to. P_i / C_i as described in Section 2.2. However, with the *HP BATS* approach, actions can discover multiple causes and different actions can discover some of the same causes, i.e., actions are dependent. When actions are dependent, it is, in general, impossible to find an optimal sequence in less than exponential time. However, an approximate algorithm that finds sequences of reasonable quality is (for k actions) :

Algorithm 1

1) For i = 1, ..., k
   a) Sort actions that haven't yet been performed wrt. P_i / C_i – pick the best one, A^*
   b) Assume that A^* is performed and adjust cause probabilities wrt. A^* failing : $P(F_j \mid A^*=\text{no}, e)$

By exploiting the fact that all actions and questions are independent given knowledge of the cause, it is possible to compute a good sequence of actions using the above algorithm. For further details on this, see [6]. These algorithms also make it possible compute whether it pays to ask a question first very efficiently.

Limited n-step look-ahead for questions

With algorithm 1, we have approximate n-step look-ahead for actions. However, we only have one-step look-ahead for questions as we only compare two situations, (i) Q is not asked at all, and (ii) Q is asked first. In *HP BATS* we have introduced a solution to this problem by computing a number P_Q^I for

all questions Q. This number represents the probability of a question identifying a cause, and can be compared on equal terms with the probabilities of actions solving the problem. Thus, questions can be used in algorithm 1 similar to actions. The formula for computing P_Q^I is as follows :

$$P_Q^I = \max_F \max_s \frac{P(F \mid Q = s, e) - P(F \mid e)}{1 - P(F \mid e)} \times P(Q = s \mid e)$$

To further improve the treatment of questions, we have extended the simple comparison of [3] between (i) ECR : performing the best sequence of steps, never asking the question, and (ii) ECO_Q : first asking the question, and then performing the best sequence of steps based on the answer. If (ii) is the best solution, we further compare with (iii) $ECO_Q^{(2)}$: perform the best action, then the question, and then the best sequence of remaining steps based on the answer. This way we avoid the bias towards the question of the dual comparison between (i) and (ii) where the question will always win if the only alternative is to never ask it.

Authoring

This section describes the *HP BATS Author* tool for acquisition of troubleshooting knowledge. The tool is generic and can be used for any application where the basic underlying assumptions are fulfilled. The most crucial is the *single-fault assumption* that requires at most one fault in the system.

The *HP BATS Author* implements the following fundamental ideas:

- The domain expert needs no knowledge of Bayesian networks. All terms are expressed in a manner that is intuitive to the domain expert. All Bayesian network structure is created implicitly by the tool.
- The domain expert can specify probabilities in the most natural / intuitive manner. Domain experts are allowed to specify probabilities in the non-causal direction if they prefer. In fact, the tool attempts to determine by the use of wizards which direction is the most natural to the domain expert in the given situation.
- The domain expert specifies potential causes and troubleshooting steps which map to underlying Bayesian network structures. These structures are combined into a Bayesian network for the troubleshooter in a sound manner.
- The tool allows maximum reuse of building blocks consisting of causes and troubleshooting steps. Logical units or components of the domain can be represented as *modules* in a *library* that can be used when constructing troubleshooting models for new error conditions.

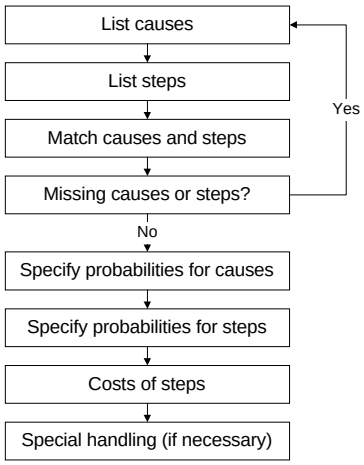

Figure 5 The knowledge acquisition process.

Knowledge Acquisition Process

The *HP BATS Author* implements the knowledge acquisition process illustrated in Figure 5, however, without enforcing the specified ordering of steps.

Causes and steps are matched by considering for each action which causes it can potentially discover and, for each question, which causes it is associated with. During this process actions that do not discover any causes and causes that are not discovered by any action are often revealed, requiring the domain expert to go back and add the missing information.

The *HP BATS Author* allows the user to construct a library of commonly used modules. If such a library exists, new models can be constructed much faster by reusing existing modules with already specified causes and steps. All the stages of Figure 5 except stage #5 can then be skipped for these causes and steps. The prior probabilities of causes will depend on the specific error condition, so these cannot be pre-specified in the library module.

In the previous section, the basic building blocks, causes, actions and questions were described and in this section it will be explained how the knowledge acquisition process for these building blocks is supported by the tool. Finally, it will be described how the building blocks are combined into models or library modules.

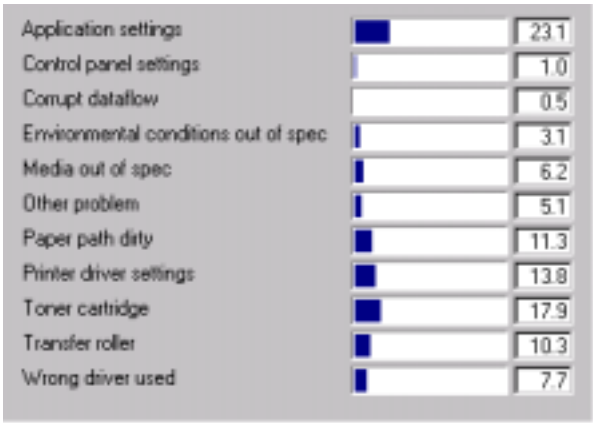

Figure 6 An example of probability assignment for a set of causes in the KA tool.

Causes and the *HP BATS Author*

Probability assignment of causes is simple in the *HP BATS Author*. First, a set of causes with the same parent in the cause tree is selected. Then, the user specifies probabilities for the causes conditional on the parent cause and error condition. If the causes in question are on the top level, i.e., they have no parent causes, then the probabilities are specified conditional only on the error condition. The user can use graphical sliders as shown in Figure 6. Figure 6 shows a list of causes of the error condition "Light print" from the printer domain. Other cause information such as the name and an explanation can be specified with a cause editor.

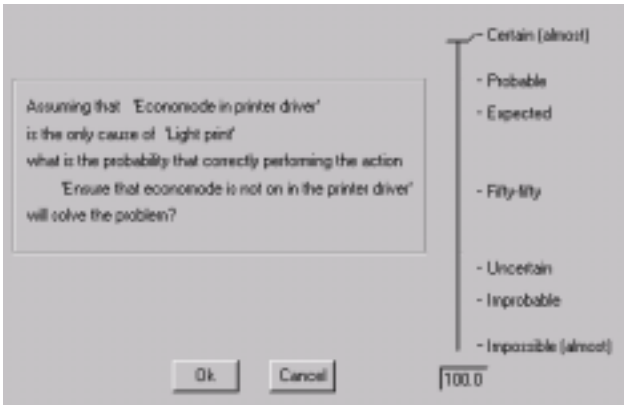

Figure 7 Eliciting the probability of an action solving a cause.

Actions and the *HP BATS Author*

The *HP BATS Author* has an action editor for specifying actions in an error condition model. This action editor allows the domain expert to specify the causes that are discovered by the action. Again, a graphical slider can be used to set the probability of the action solving a cause, as seen in Figure 7 where we have a typical elicitation - an action that discovers a cause with absolute certainty. As the action is assumed to be performed correctly, these elicitations are mostly very simple and rapidly performed. The graphical slider in Figure 7 was borrowed from [2].

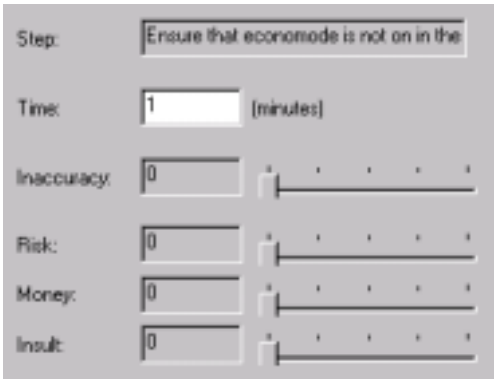

Figure 8 Cost editor

The *cost editor* is depicted in Figure 8. The various cost factors can be specified with graphical sliders. Time is specified in minutes but with the other factors a five level scale is used (0=none, low, medium, high, very high).

The inaccuracy factor is used for specifying the uncertainty involved with the action, i.e., how much the user's result is trusted. This factor is converted into *P(correct)* used in Eq. (4) above.

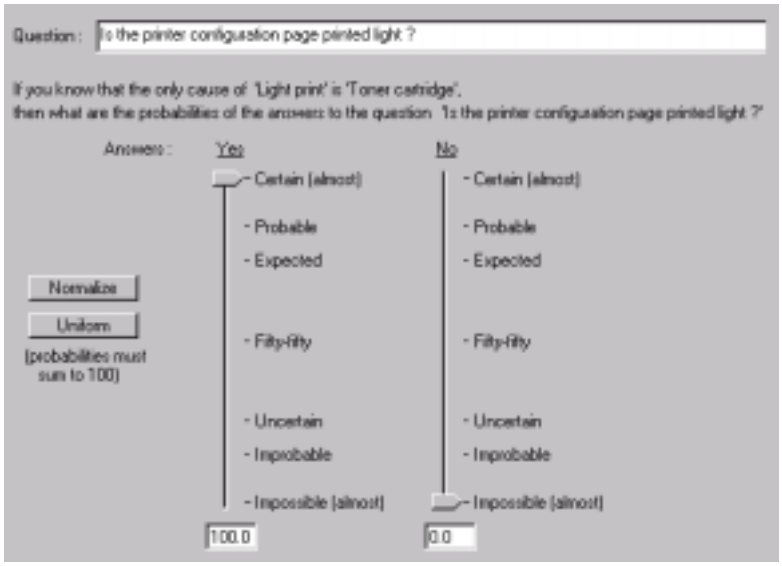

Figure 9 Probability elicitation for a question that concerns a symptom.

Questions and the *HP BATS Author*

The *HP BATS Author* has a question wizard for specifying questions in an error condition model. The question wizard first helps the domain expert decide the question's type, i.e., whether the question concerns a symptom of the causes, or whether it concerns something that could have caused or created the problem. Determining the type of the question allows the *HP BATS Author* to provide the domain expert with the most intuitive and natural way of eliciting the probabilities, i.e., as the probability of the question conditional on the causes, or as the probabilities of causes conditional on the question.

Typically, the domain expert is not familiar with conditional probabilities and is not able to choose the best way to elicit the probabilities, so by determining the type of the question the *HP BATS Author* leads directly to the best method.

There are questions that are not easily categorized but it is our experience that for these questions it usually does not matter which way the probabilities are elicited.

For very simple types of questions that either eliminate or identify causes based on the answer, the *HP BATS Author* provides short cuts to allow the domain expert to directly specify the eliminated or identified causes. Of course, depending on the question type, he will still have to either specify prior probabilities for the question answers, or probabilities of non-eliminated and non-identified causes conditional on the question answers.

P(Question | Causes)

The *HP BATS Author* provides graphical sliders for editing the probabilities of answers to questions conditional on a specific cause being present, see Figure 9.

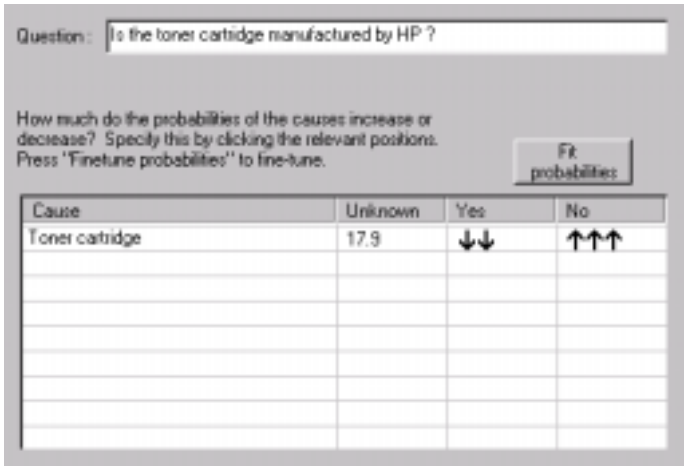

Figure 10 Probability elicitation for a question that concerns something that could have caused the problem.

P(Causes | Question)

For specifying the probabilities of causes conditional on question answers, the *HP BATS Author* provides the domain expert with two levels of precision. In Figure 10 and Figure 11 are shown the approximate specification method. In Figure 10 is shown an overview of the causes that are associated with the question and arrows indicate how the probability of the cause is modified for each answer to the question. Figure 11 shows the choices available.

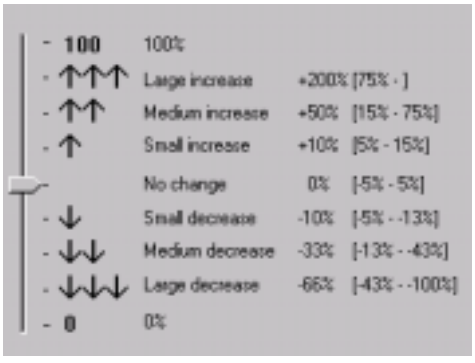

Figure 11 Probability elicication for questions.

As mentioned earlier, the probabilities specified here always have to satisfy Eq. (5), so the domain expert's selection of arrows may not necessarily be consistent with other probabilities. The button labeled "Fit probabilities" in Figure 10 starts an algorithm that attempts to satisfy as many of the domain expert's wishes as possible. The algorithm simply runs through the causes one at a time,

attempting to satisfy the wishes for that cause. If the wishes cannot be satisfied, it is attempted to satisfy them partly, e.g., change three up arrows to two, etc. The assumption is that it is better to satisfy two wishes partly than satisfy one wish completely and not satisfy another one.

Often, the approximate specification in Figures 10 and 11 is sufficient - and often the domain expert is not capable of assessing the probabilities with any higher degree of accuracy anyway. However, in some situations more precision and accuracy is required. For these situations the *HP BATS Author* provides a highly complex set of sliders for all the probabilities in Eq. (5) where any slider can be set as wanted, and all affected sliders will be adjusted accordingly.

Combining the Pieces

In the previous sections it was explained how the three pieces, causes, actions and questions, could be specified with the *HP BATS Author*. Causes are constructed as a tree which is later collapsed to a single cause indicator variable in the Bayesian network. Actions and questions are represented as children of this cause indicator variable. The simplicity of this Bayesian network allows for very efficient algorithms for finding the best sequence of steps.

The *HP BATS Author* allows the domain expert to construct a new error condition model from scratch by following the KA process laid out in Figure 5, or, if a library of modules exists, appropriate modules can be used to quickly populate a new error condition model.

The benefits of allowing reuse can be illustrated with examples from the printer domain. For example, the toner cartridge component can be a cause of many different error conditions, most of them related to print quality. It is our experience that with a well-populated library of modules, the development time for new error conditions can be cut at least in half, as 70-80% of the causes can be directly copied from the library.

Thus, the *HP BATS Author* allows for a great deal of reuse between models. Further, the tool maintains consistency between shared modules such that when information changes in a library module this is mirrored in all the error condition models where this module was used.

The *HP BATS Author* also provides excellent search and replace facilities that makes maintenance and migration highly efficient.

Validation

The *HP BATS* system contains facilities for efficient validation of new models. The basic idea is to generate a large number of cases by simulating user feedback to the Troubleshooter. These cases can then be validated by a domain expert to see whether the model is functioning correctly.

The validation system consists of two components, a *case generator* and a *case evaluator*. The *case generator* has the ability to produce random or special cases. The random cases are simulated by picking user feedback with respect to the probabilities of answers. These probabilities depend on the prior probabilities estimated by the domain experts themselves and observations from previous steps in the sequence. The outcome of a random case – problem solved or not – can be represented as a binomial distribution, making it possible to calculate confidence intervals over the quality of the model.

The special cases are generated by traversing the complete troubleshooting tree spanned by the model. For large models, a complete traversal is not feasible, however, so the user is able to stop the case generator whenever it is desired. Among the cases traversed, some criteria are used to select cases for evaluation :

1. Lengthy cases : cases with many steps
2. Costly cases : cases with high total cost of all steps
3. Failing cases : cases that fail to solve the problem

This allows the domain expert to focus his attention on the weakest areas of the model.

References

[1] Cowell, R.G., Dawid, A.P., Lauritzen, S.L., and Spiegelhalter, M.R.C. (1999). Probabilistic Networks and Expert Systems. Springer-Verlag, 1999.

[2] van der Gaag, L., Renooij, S., Witteman, C., Aleman, B., and Taal, B. (1999). How to Elicit Many Probabilities. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, 1999.

[3] Heckerman, D., Breese, J., and Rommelse, K. (1995). Decision-theoretic Troubleshooting. *Communications of the ACM*, 38:49-57.

[4] Jensen, F.V. (1996). *An Introduction to Bayesian Networks*. UCL Press, London.

[5] Jensen, F.V. and Lauritzen, S.L. and Olesen, K.G. (1990). Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Computational Statistics Quarterly*, 4:269-282.

[6] Skaanning, C., Jensen, F.V., and Kjærulff, U. (2000). Printer Troubleshooting Using Bayesian Networks, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE) 2000*, New Orleans, USA, 19-22 June, 2000.

[7] Skaanning, C. A Knowledge Acquisition Tool for Bayesian-Network Troubleshooters. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, 2000.