SUPPORTING COMPLEXITY IN MODELING BAYESIAN TROUBLESHOOTING

Luigi Troiano
University of Sannio, Department of Engineering, CISELab, Viale Traiano, 82100 Benevento, Italy

Davide De Pasquale
Intelligentia s.r.l., Via A. De Blasio 34, 82100 Benevento, Italy

Keywords: Troubleshooting, Bayesian networks, Maintenance, Decision support tools.

Abstract: Troubleshooting complex systems, such as industrial plants and machinery, is a task entailing an articulated decision making process hard to structure, and generally relying on human experience. Recently probabilistic reasoning, and Bayesian networks in particular, proved to be an effective means to support and drive decisions in Troubleshooting. However, troubleshooting a real system requires to face scalability and feasibility issues, so that the direct employment of Bayesian networks is not feasible. In this paper we report our experience in applying Bayesian approach to industrial case and we propose a methodology to decompose a complex problem in more treatable parts.

1 INTRODUCTION

Troubleshooting, i.e. solving failures of, complex systems is a challenging task assuming that although effects are directly observable, malfunctioning sources cannot be deterministically located and removed. In general, Troubleshooting is aimed at (i) identifying precisely failure causes, and definitively removing them, (ii) verifying that a failure did not lead to other breakdowns, (iii) collecting accurate statistics regarding failures causes and occurrences, thus providing a feedback to designers and engineers in order to improve the product. And this requires intelligence to be accomplished successfully.

Recent advances in computational techniques, especially those in the field of Artificial Intelligence, provide new opportunities for supporting Troubleshooting. In particular, probabilistic reasoning represents a natural setting for processing troubleshooting strategies, given uncertainty in linking effects to malfunction causes. However, troubleshooting real systems requires to face complexity issues in modeling a thick network of relationships between effects and faults, considering their links among parts and sub-systems. In addition models have to be kept maintained over the time.

These issues mainly limited a wider adoption of supporting tools in industrial Service and Maintenance (S&M) activities when plant and machinery faults occur. These activities still largely rely on the technical competence of skilled personnel, although the increasing complexity of systems to maintain, the shorter retention of knowledge, also due to the higher personnel turnover, the skill shortage and the lack of training at enterprise level, and the need of limiting operational costs demand to assist them more by machine intelligence.

Therefore, the adoption of supporting tools does not relate only to mathematical and technological issues, but also and more prominently to issues concerning complexity and management of models. So it becomes mandatory to to look at complex objects as systems made of parts and sub-systems, each being a possible, sometimes hidden, source of failures.

In this paper we report our experience in designing and adopting a support tool based on Bayesian networks for industrial machinery, focusing more on usability and maintainability of models than innovating the mathematical framework. This contribution is organized as follows: in Section 2 we briefly overview approaches to Troubleshooting, Section 3 outlined the Bayesian approach, Section 4 described a method for modeling real world problems, Section 5 draws conclusions and future directions.

2 TROUBLESHOOTING: TECHNOLOGIES AND ISSUES

Several technologies relying on Artificial Intelligence and Decision Making have been proposed over the time in order to support Troubleshooting activities.

Information Text Retrieval (ITR) systems became popular in the 90s, especially with respect to document management of products and S&M reporting. Their main function is devoted to store in and retrieve from a knowledge base information regarding specific problems, products, past repairing tasks.

A major limitation of this approach resides in the amount of information that can be effectively managed, as complex industrial products generally entail a large number of retrievable data and documentation, thus leading to an excess of information that hides the actual needs and to read and understand documents written by others. In addition, often information is not self contained is one document or section, but split in several parts, requiring ability in relating sources.

In order to overcome limitations of early ITR approaches, a more sophisticated class of tools rely on Case Based Reasoning (CBR). In detail CBR processing can be outlined considering only four main steps: (i) *retrieve*: the user describes its problem submitting it to the CBR system, obtaining a set of cases that suggest which operations to take for solving the current problem; (ii) *reuse*: retrieved cases can be re-used to solve new problems; *revise*: a case used in the past is tested against new problems; *retain*: new cases can be compiled and retained in the CBR system.

Differently from ITR, users can provide more detailed description of failure events mostly based on symptoms, thus allowing to retrieve problems solved in the past by similarity to the current one. This approach is able to better model the human experience in problem solving, addressing the issue of refining generic queries as shown by ITR. Therefore CBR performs better when dealing with large knowledge bases. However, they heavily rely on the case description given by user in natural language, affecting the quality of results as documents are indexed by descriptions that can be imprecise, inaccurate and erroneous.

Both ITR and CBR do not provide a structured path to problem solution in order to automate Troubleshooting. This led to employ Decision Trees (DT) to assist S&M operators in taking repairing actions. DT are able to drive maintenance operations by modeling decision paths and their possible consequences, including resource costs, utilities, chance event outcomes and other factors.

Often this approach requires domain experts to design and validate the tree structure, although parameters can be obtained by training the model against statistical data. During troubleshooting, users can retrace the structure, following suggested decisions until the problem is solved. Modeling decision trees becomes quickly cumbersome and maintaining them very complex and error prone, especially for unexperienced users. Human experience still plays a relevant role in keeping up-to-date this kind of models, as modifications can invalidate the structure producing a high impact on model maintenance costs.

Maybe Expert Systems (ES) have been the most widely approach investigated and applied to Troubleshooting. An expert system attempts to solve a problem by explicitly modeling knowledge and inferring solutions by causal reasoning as human would do. For this reason, they are generally specific to domains. In using this kind of tools, (i) creation of a knowledge base is made by capturing the expert's domain knowledge, and (ii) solutions are found by logically inferring conclusions given a set of collected premises. This leads to have white-box models, thus more intelligible, while other techniques mainly provide black-box models. Other advantages are consistent answers for repetitive decisions, robust theoretical framework in decision making, improved user experience, and reduced management cost.

Main limitations regard impossibility of obtaining creative and unsuspected responses, as human experts generally are able to provide. Quality of results depends on quality of knowledge encoded when the model is designed, thus often leading to errors and wrong decisions due partial knowledge. Complex and dynamic problems are difficult to encode.

3 BAYESIAN APPROACH

Technologies presented so far share in common some limitations that can be summarized as follows:

1. Troubleshooting inherently deals with uncertainty, as malfunctioning causes cannot be deterministically identified even by humans. Instead most of outlined approaches make the link between symptoms and causes deterministic.
2. They generally do not employ a divide-et-conquer strategy, thus badly fitting real world systems, made of complex aggregations of parts, each able to affect correct functioning of the whole system or of other parts.
3. As knowledge is hard-embedded, models are difficult to maintain and evolve.

Bayesian Networks (BN) Perl (Pearl, 1988) provide an answer to issues above as: (i) they are a probabilistic reasoning technique, thus they naturally and explicitly model uncertain sources of information; (ii) they can be built by aggregation or decomposition, according respectively to both bottom-up and top-down design strategies; (iii) they allow to model local relations but, as they rely on strong theoretical basis given by Probability Theory, they still preserve coherence of the whole and make models easy to maintain and evolve

According to Heckerman, Breese and Rommelse (Heckerman et al., 1994) the purpose of Troubleshooting is to generate a low cost plan in repairing a system. A repairing plan consists in a list of tasks, observations and repairing actions to perform in order to reach the goal of solving the malfunctioning condition. They describe a typical Troubleshooting process as a sequence of the following activities:

1. If the component works properly terminate,
2. If not either
   (a) select component to replace or repair
   (b) select an unobserved variable for observation or
   (c) call Service
3. Go to Step 1

Following this approach, in a Bayesian network we can link faults (parent nodes) to repairing actions (children nodes) by probabilistic relations (edges). This leads to determine how likely an action is able to repair a specific failure. Similarly, Jensen et al. (Jensen et al., 2001a) define a troubleshooting strategy as an optimal sequence of repairing actions minimizing the overall expected cost[1] of the sequence. In their ground breaking work both Heckerman and Jensen have introduced the following entities: *faults*, *repairing actions* and *observations*[2]. Heckerman considers also *non reparable* faults.

Fault nodes describe observable problems for a particular system while *repairing actions* represent operations that can be taken in order to repair at least one fault. Finally *observations* represent visible operating states of machine/system. Actions A_i have associated a repairing cost C_i and repairing probability P_i. The latter depends on evidence collected so far by observations, as some faults become more likely than others.

In order to evaluate an optimal plan, given a sequence of repairing actions $\mathbf{A} \equiv < A_1, \ldots, A_n >$, both Heckerman et al. (Heckerman et al., 1994) and Jensen et al. (Jensen et al., 2001a) argue that the optimal repair sequence is obtained by minimizing the function

$$ECR(< A_1, \ldots, A_n >) = \sum_i ECR(i) \qquad (1)$$

where *ECR* stands for Expected Cost of Repair, each contribution is given as $ECR(i) = C(\varepsilon^{i-1})P(\varepsilon^{i-1})$, and ε^{i-1} is the evidence that the first $i-1$ actions failed. In general, computing (1) can be problematic. However some simplifying assumptions can be made. Kalagnanam and Henrion (Kalagnanam and Henrion, 1990) assume the following conditions:

1. There is a single fault and only one component is responsible of the device failure
2. Cost of repairing actions is independent on the sequence of actions
3. There is no observation interleaved between two repairing actions

Under these circumstances, the optimal repairing sequence is the one assuming actions ordered by decreasing efficiency defined as $C_i/P_i|\varepsilon$.

This order can change as far as we collect evidence on failures, thus varying the repairing probability $P_i|\varepsilon$ of following actions.

Observations are merely considered as state analysis steps along the troubleshooting process, used to reduce uncertainty regarding the set of possible faults. However, they can drastically change the optimal order of repairing actions. Therefore, we can assign a repairing cost due to observations, i.e. *Expected Cost of Observation* (ECO) defined as

$$ECO(Ob|\varepsilon) = \sum_{s \in St(Ob)} P(Ob = s|\varepsilon) ECR(\mathbf{A}|Ob = s, \varepsilon)$$

where $St(Ob)$ is the set of possible observation states, $\mathbf{A}|Ob = s, \varepsilon$ is the sequence of available actions ordered by efficiency given the ε and $Ob = s$. Indeed that is the cost of repair expected when new evidence (i.e. $Ob = s$) is added to collected evidence ε. Performing an observation is convenient when $ECO(Ob|\varepsilon)) > ECR(\mathbf{A}|\varepsilon)$.

Bayesian approach to Troubleshooting is extremely flexible, simple and facilitates the model maintenance. Examples of commercial applications of Bayesian Troubleshooting are Microsoft MSBNx (Kadie et al., 2001), Hewlett Packard SACSO (Jensen et al., 2001b), and more recently Dezide Advisor$^{\text{TM}}$. However modeling structure and estimating parameters (i.e. conditional probabilities, costs, etc.) becomes soon cumbersome even for slightly complex problems. So if on one side several large and medium sized companies have experimented the Bayesian

[1]For cost we intend in a general meaning monetary and non-monetary costs, such as MTTR - Mean Time To Repair costs.

[2]Jensen considers observations in terms of questions to the user.

Figure 1: BalanceSystems BVK4.

Troubleshooting (Khanafer et al., 2008; Verma et al., 2004), on the other only smaller and limited case studies have been considered, as models were not able to scale complexity, and large effort was required in designing the Troubleshooting process, in defining faults, observations and actions, and finally in relating them. This led to consider the problem of how to deal with increasing complexity, preserving maintainability and usability of models.

4 A METHOD FOR MODELING TROUBLESHOOTING

A real system is made of subsystems entailing a large number of parts, each able to affect the failures of the whole system. Parts are generally instances of classes of components, therefore their behavior depends on their design and serial production. This can lead quickly to complex models that is not possible to manage at the glance. It would be desirable to reuse knowledge and decompose the problem in order to face complexity.

In this section we outline a strategy that makes possible to model Troubleshooting problems for complex systems using Bayesian approach. As an example of a real world system let us consider BVK4, a balancing machine manufactured by Balance Systems (see Figure 1).

BVK4 is a class of semi-automatic balancing machines for rotating parts. The machinery is made by three three sub-systems:

- Measuring Station aimed at estimating (i) initial unbalance of parts (ii) an optimal position of parts to be processed, (iii) unbalance of parts after been processed
- Working Station performs the balancing of parts
- Control Panel grouping together all controllers, namely the *electronic controllers*, *lock off grabs controllers* used for blocking pieces during balancing and *user input controllers* managed by Human/Machine User Interface.

In order to face such a complexity, we propose the following strategy:

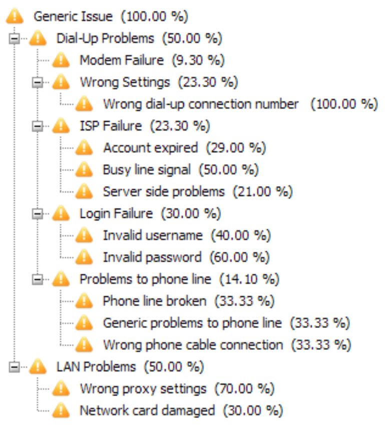
1. Model the system as made of re-usable parts
2. Identify possible failures and organize them in groups
3. Link the system-level failures to part-level failures
4. Estimate the relative fault frequencies
5. Identify actions able to solve the failures, and related costs
6. Assign a repairing probability to actions given the failures
7. Provide a set of observations able to reveal failures
8. Relate observation states to failures

In describing our approach we will make use of a prototype supporting tool named *Dygnose*, part of a research program led by Balance Systems s.r.l. in cooperation with University of Sannio. As an example of application we will consider the troubleshooting of Internet Service Provider (ISP) connection problems.

Dygnose makes explicit the structure of systems in parts. Parts can be elementary components or subsystems. For example a car is made of parts such as engine, mechanics and passenger compartment. Each of these parts is a system itself, thus decomposable itself in subsystems. Obviously, failures at system level depend on failures at part level, and that relations between parts leads to relations between failures. Therefore a first step in facing a Troubleshooting modeling problem consists in decomposing a system in more elementary components, and to analyze them individually.

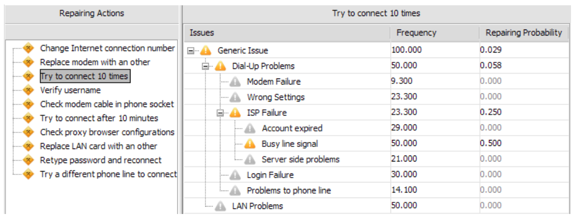
Another source of abstraction relies in the fact that individuals belong to classes characterized by similar (or even the same) structure, thus entailing similar failures, symptoms and repairing actions. For example each individual car belong to a series, as produced according to the same design. Therefore, although each car is specific, the whole series will entail similar faults and solutions to them. In addition, generalization can be led at different levels of abstraction, so that different classes can be generalized into a larger class, in order to pool troubleshooting issues in common. Each class can refer to a more general class of systems, inheriting structure, possible failures, symptoms and repairing actions.

System are characterized by a set of failures organized in a tree structure aimed at classifying them.



- Generic Issue (100.00 %)
  - Dial-Up Problems (50.00 %)
    - Modem Failure (9.30 %)
    - Wrong Settings (23.30 %)
      - Wrong dial-up connection number (100.00 %)
    - ISP Failure (23.30 %)
      - Account expired (29.00 %)
      - Busy line signal (50.00 %)
      - Server side problems (21.00 %)
    - Login Failure (30.00 %)
      - Invalid username (40.00 %)
      - Invalid password (60.00 %)
    - Problems to phone line (14.10 %)
      - Phone line broken (33.33 %)
      - Generic problems to phone line (33.33 %)
      - Wrong phone cable connection (33.33 %)
  - LAN Problems (50.00 %)
    - Wrong proxy settings (70.00 %)
    - Network card damaged (30.00 %)

Figure 2: Dygnose: Possible ISP connection failures.



Repairing Actions

- Change Internet connection number
- Replace modem with an other
- Try to connect 10 times
- Verify username
- Check modem cable in phone socket
- Try to connect after 10 minutes
- Check proxy browser configurations
- Replace LAN card with an other
- Retype password and reconnect
- Try a different phone line to connect

Try to connect 10 times

Issues	Frequency	Repairing Probability
Generic Issue	100.000	0.029
Dial-Up Problems	50.000	0.058
Modem Failure	9.300	0.000
Wrong Settings	23.300	0.000
ISP Failure	23.300	0.250
Account expired	29.000	0.000
Busy line signal	50.000	0.500
Server side problems	21.000	0.000
Login Failure	30.000	0.000
Problems to phone line	14.100	0.000
LAN Problems	50.000	0.000

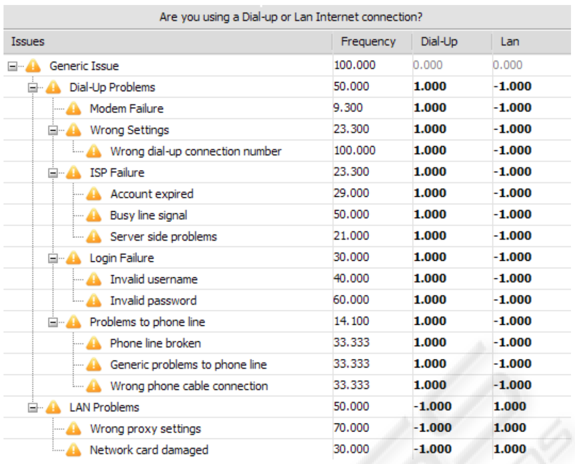
Figure 3: Dygnose: Probabilities of repairing actions.

In modeling ISP connection problems, we first create the fault tree as depicted in Figure 2. In particular, we have classified failures in two categories: Dial-up and Lan problems. Each of these is further specified. The percentage value associated to each fault represents the relative frequency of happening within the class of problems it belongs to.

As noted before, there is relationship between the failures of parts and system they belong to. For example if there is a modem failure at connection level, this is due to a set of possible failures at device level. This linkage can have a double nature: (i) it can specify at part level a problem or (ii) it defines a causal relation, so that a problem at part level produces a side-effect at system level. Both meanings can be just modeled by considering a failure cause at system level that is a proxy of its counter part at component level.

After the failures tree is specified, we can provide a set of repairing actions as troubleshooting solutions. Each action is likely able to solve a given failure. This is done by assigning a repairing probability with respect to each failure, as depicted in Figure 3. As default we assume that the repairing probability is 0 if not differently specified.

In addition each repairing action entails a cost when the action has place. This cost is multidimensional, entailing a usage of different resources



Are you using a Dial-up or Lan Internet connection?

Issues	Frequency	Dial-Up	Lan
Generic Issue	100.000	0.000	0.000
Dial-Up Problems	50.000	1.000	-1.000
Modem Failure	9.300	1.000	-1.000
Wrong Settings	23.300	1.000	-1.000
Wrong dial-up connection number	100.000	1.000	-1.000
ISP Failure	23.300	1.000	-1.000
Account expired	29.000	1.000	-1.000
Busy line signal	50.000	1.000	-1.000
Server side problems	21.000	1.000	-1.000
Login Failure	30.000	1.000	-1.000
Invalid username	40.000	1.000	-1.000
Invalid password	60.000	1.000	-1.000
Problems to phone line	14.100	1.000	-1.000
Phone line broken	33.333	1.000	-1.000
Generic problems to phone line	33.333	1.000	-1.000
Wrong phone cable connection	33.333	1.000	-1.000
LAN Problems	50.000	-1.000	1.000
Wrong proxy settings	70.000	-1.000	1.000
Network card damaged	30.000	-1.000	1.000

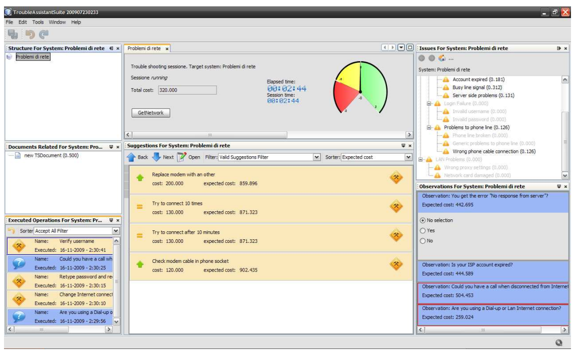
Figure 4: Dygnose: Relating observations to failures.

such as time, budget, people, materials and even user attitude.

In resolving the ISP connection problem, we consider the following actions:

- Change Internet connection number and retry to connect (cost=100)
- Replace modem with an other (cost=200)
- Try to connect 10 times (cost=130)
- Check username currently in use (cost=110)
- Retype password and reconnect (cost=110)
- Check modem cable in phone socket (cost=120)
- Try to connect after 10 minutes (cost=130)
- Check proxy browser configurations (cost=130)
- Replace Lan card with an other (cost=200)
- Try a different phone line to connect (cost=140)

Failure symptoms are the last aspects to consider. Observations are the means by which the failing system is looked at and symptoms detected. In particular, each observation depicts a particular observable aspect of the system by a set of operating conditions. These conditions can reveal some insights regarding the actual failure. However symptoms are not deterministically related to failures, and a conditional probability distribution is required to link the twos. In order to reduce the complexity of specifying such a parameter, observations and failures can be linked by a correlation index. This index varying between -1 and 1, makes possible to consider symptoms able to certainly exclude a failure (correlated -1) and to identify the failure (correlated $+1$); we assume 0 when it is not possible to determine such a relation. An example in modeling observations is provided in Figure 4. Namely, we considered:


TroubleAssistantSuite 200907230233
File
Edit
Tools
Window
Help
Structure For System: Problemi di rete
Problemi di rete
Documents Related For System: Pro...
Executed Operations For System: Pr...
Sorter
Suggestions For System: Problemi di rete
Back
Next
Open
GetNetwork
Issues For System: Problemi di rete
Observations For System: Problemi di rete

new TSDocument (0.500)

Trouble shooting sessione. Target system: Problemi di rete

Sessione *running*

Total cost: 320.000

Elapsed time: 00:02:44
Session time: 00:02:44

Sorter: Accept All Filter

Name: Verify username
Executed: 16-11-2009 - 2:30:41

Name: Could you have a call wh
Executed: 16-11-2009 - 2:30:25

Name: Retype password and re
Executed: 16-11-2009 - 2:30:15

Name: Change Internet connect
Executed: 16-11-2009 - 2:30:10

Name: Are you using a Dial-up o
Executed: 16-11-2009 - 2:29:56

Filter: Valid Suggestions Filter   Sorter: Expected cost

Replace modem with an other
cost: 200.000   expected cost: 859.896

Try to connect 10 times
cost: 130.000   expected cost: 871.323

Try to connect after 10 minutes
cost: 130.000   expected cost: 871.323

Check modem cable in phone socket
cost: 120.000   expected cost: 902.435

System: Problemi di rete
Account expired (0.181)
Busy line signal (0.312)
Server side problems (0.131)
Login Failure (0.000)
Invalid username (0.000)
Invalid password (0.000)
Problems to phone line (0.126)
Phone line broken (0.000)
Generic problems to phone line (0.000)
Wrong phone cable connection (0.126)
LAN Problems (0.000)
Wrong proxy settings (0.000)
Network card damaged (0.000)

Observation: You get the error "No response from server"?
Expected cost: 442.695
No selection
Yes
No

Observation: Is your ISP account expired?
Expected cost: 444.589

Observation: Could you have a call when disconnected from Internet
Expected cost: 504.453

Observation: Are you using a Dial-up or Lan Internet connection?
Expected cost: 259.024

Figure 5: Dygnose: Supporting the user in solving a failure.

- Connection type (States: Dial-up, Lan)
- No response from server (States: Yes, No)
- ISP account expired (States: Yes, No)
- Phone calls available (States: Yes, No)

Once the Troubleshooting problem has been fully specified, the model can be translated to a Bayesian Network. The goal is to collect evidence on what is the actual failure, according to observations and attempts to remove it.

In executing a troubleshooting model, a Service employee is supported by suggesting a series of possible repairing actions ordered by decreasing ECR (see Figure 5). In addition, observations are listed by decreasing ECO.

Although actions and observations are proposed according to optimal criteria, the user is free to perform them in any order until the problem is solved. As far as new evidence is collected, fault, repairing action and observation probabilities are updated, and actions/observations are sorted accordingly.

Modeling real world problems can be problematic. The BVK4 troubleshooting model is currently described by 9719 lines of text, entailing 2839 properties and 12687 parameters, even considering a reduced structure made of 56 sub-systems and 69 parts. The model describes 241 possible failures, 102 observations and 117 repairing actions. However, although referring a complex system, we were able to complete the modeling in about a working month employing three designers: 1 full-time and 2 part-time (2 hours per day). The model was tested against several problems, and was able in general to lead the user to the problem resolution.

5 CONCLUSIONS

Bayesian Troubleshooting provides an interesting theoretical framework to develop a new generation of decision support tools in solving failures, able to overcome limitations of former technologies. Although strong theoretical basis and availability of computational methods make this approach really promising, and effective application to real world problems, such as servicing industrial machinery, requires to develop a divide-et-conquer strategy to model complex systems and tools to make it feasible. In this paper we presented a method to decompose Troubleshooting modeling in more treatable steps and Dygnose, a tool designed to support it. In a preliminary experimentation we successfully considered an industrial machine. However, developing an effective decomposition strategy is not sufficient by itself. Indeed, new questions are posing. For instance how to test and debug a complex model is an open issue we aim at answering in the future.

REFERENCES

Heckerman, D., Breese, J., and Rommelse, K. (1994). Troubleshooting under uncertainty. Technical report, Microsoft Research.

Jensen, F., Kjaerulff, U., Kristiansen, B., Langseth, H., Skaanning, C., Vomlel, J., and Vomlelová, M. (2001a). The sacso methodology for troubleshooting complex systems. *Artif. Intell. Eng. Des. Anal. Manuf.*, 15(4):321–333.

Jensen, F. V., Skaanning, C., and Kjaerulff, U. (2001b). The sacso system for troubleshooting of printing systems. In *SCAI '01: Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*.

Kadie, C., Hovel, D., and Horvitz, E. (2001). Msbnx: A component-centric toolkit for modeling and inference with bayesian networks. Technical report, Microsoft Research Technical Report MSR-TR-2001-67.

Kalagnanam, J. and Henrion, M. (1990). A comparison of decision alaysis and expert rules for sequential diagnosis. In *UAI '88: Proc. of 4th Annual Conf. on Uncertainty in Artificial Intelligence*, pages 271–282. North-Holland Publishing Co.

Khanafer, R., Solana, B., Triola, J., Barco, R., Nielsen, L., Altman, Z., and et al. (2008). Automated diagnosis for umts networks using bayesian network approach. In *IEEE Transactions on Vehicle Technology*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Verma, V., Gordon, G., Simmons, R., and Thrun, S. (2004). Real-time fault diagnosis. *IEEE Robotics & Automation Magazine*, 11(1):56 – 66.