Automated End-To-End System Diagnosis of Networked Printing Services Using Model-Based Reasoning

George Forman, Mudita Jain, Masoud Mansouri-Samani,*
Joe Martinka, Alex Snoeren†
Software Technology Laboratory
HPL-98-41 (R.1)
September, 1998

E-mail:[gforman,jainm, martinka]@hpl.hp.com
mms@hpprsdl.sc.hp.com; snoeren@mit.edu

distributed service management,
client-server systems,
automated end-to-end diagnostics,
model-based reasoning,
MBR,
network printing,
artificial intelligence

Modern applications increasingly depend on services that are distributed across the infrastructure, for example, printing, mail and database services. This is especially so in mobile computing, where network access is used to compensate for the paucity of onboard resources. However, the increased dependence on distributed services poses a challenge to IT departments whose personnel and tools typically focus on individual components of the distributed service. This calls for IT management tools that have an end-to-end perspective on entire distributed services.

We developed a prototype of such a tool for distributed printing services and tested it in a large IT department. The diagnostic tool (1) discovers the dependencies between modeled components, (2) gathers information from the distributed environment, (3) diagnoses diverse system problems, and (4) monitors some components to allow proactive resolution of certain classes of problems before users encounter them. Our approach used model-based reasoning as the fundamental technology for the prototype.

IT personnel were delighted with this tool, especially its ability to correlate information from multiple heterogeneous subsystems. During a two-month experiment, its use was credited with fewer support calls overall, and a 7% improvement in the percentage of printing related problem calls that are resolved during the initial call to the help desk.

Internal Accession Date Only

*Hewlett-Packard Company, Solution Services Division, Santa Clara, California
†Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts

AUTOMATED END-TO-END SYSTEM DIAGNOSIS OF NETWORKED PRINTING SERVICES USING MODEL-BASED REASONING

George Forman, Mudita Jain, Joe Martinka,
Masoud Mansouri-Samani, Alex Snoeren[1]

Hewlett-Packard Laboratories
Palo Alto, California and Bristol, U.K.
gforman, jainm, martinka@hpl.hp.com,
mms@hpprsd1.sc.hp.com, snoeren@mit.edu

Modern applications increasingly depend on services that are distributed across the infrastructure, for example, printing, mail and database services. This is especially so in mobile computing, where network access is used to compensate for the paucity of onboard resources. However, the increased dependence on distributed services poses a challenge to IT departments whose personnel and tools typically focus on individual components of the distributed service. This calls for IT management tools that have an end-to-end perspective on entire distributed services.

We developed a prototype of such a tool for distributed printing services and tested it in a large IT department. The diagnostic tool (1) discovers the dependencies between modeled components, (2) gathers information from the distributed environment, (3) diagnoses diverse system problems, and (4) monitors some components to allow proactive resolution of certain classes of problems before users encounter them. Our approach used model-based reasoning as the fundamental technology for the prototype.

IT personnel were delighted with this tool, especially its ability to correlate information from multiple heterogeneous subsystems. During a two-month experiment, its use was credited with fewer support calls overall, and a 7% improvement in the percentage of printing related problem calls that are resolved during the initial call to the help desk.

Keywords*: distributed service management, client-server systems, automated end-to-end diagnostics, model-based reasoning, MBR, network printing, artificial intelligence*

1 Introduction

Business users in modern computing environments increasingly rely on distributed services, i.e., services that are provided by an ensemble of distributed components. For example, distributed printing services depend on the correct interoperation of printers, client computers, and one or more intermediate print spoolers. Mobile users have a heightened dependence on distributed services, as they tap into the infrastructure to help compensate for the resource constraints of their mobile devices [Rud98, Zen97].

The growing complexity and business importance of distributed services puts a great deal of pressure on IT departments, and may cause a breakdown in traditional support where the division of responsibilities within the IT shop *and their often low-level management tools* hinders the diagnosis and recovery of distributed services. The increasing dependence on distributed services

[1] Alex Snoeren worked on this project as a student research intern. He is now with the MIT Laboratory for Computer Science.

makes it harder for help desk staff to resolve problem calls—they must often cope with disparate independent tools manually to diagnose the various parts of the whole system and then coordinate repair across IT services. Increasingly, mobile systems mean additional problems with dynamic reconfiguration and users that are not familiar with the proper configuration. The result: more user down time. This situation heightens the need for management tools that are capable of addressing distributed services. We hypothesize that such tools will make a key difference to the future of IT management.

To test this hypothesis we collaborated with the Hewlett-Packard Labs IT department, addressing their most significant problem, according to their analysis: distributed printing. In this domain, the points of failure include hardware, software drivers and incorrect configurations anywhere along the network path from the client machine, to the print server, to the printer itself (shown in the upper portion of Figure 1). The enterprise scale of these print services at HP Labs provides a test-bed of over 350 printers (50 different models), 9 official spoolers, over 1200 employees, homed in seven buildings at two sites. The clients and print servers are mixed HP-UX and Windows NT, with mobile computers typically running Windows 95. Spooled jobs regularly cross operating system platforms.

We prototyped and deployed a sophisticated tool to automate diagnosis and monitoring of this environment. The tool employs various remote access mechanisms to gather needed information from the distributed components. Control and diagnosis is managed by an HP-developed logic language and reasoning engine: Flipper [Pel95]. Besides system integration issues, our efforts were concentrated on building a logical model of the distributed printing service. Model-based reasoning enables the same model to be used, for example, for diagnosis, verification, monitoring, and documentation [Wil96, Roc97]. It is harder to leverage such re-use from traditional, imperative programming approaches.

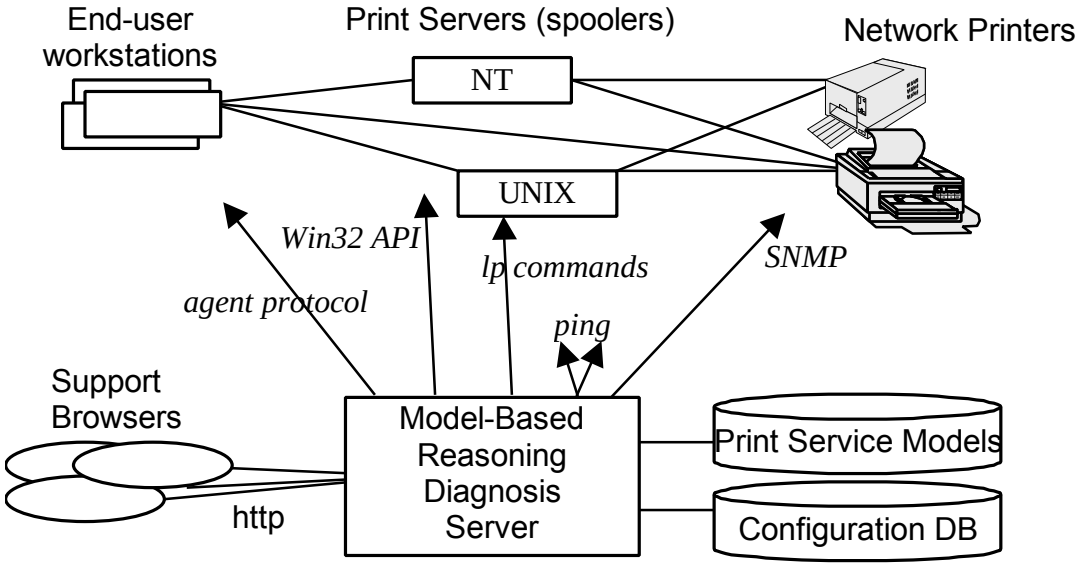

Figure 1: Conceptual model of the distributed print service (top) showing the various instrumentation methods (middle) employed by our prototype (bottom).

The next section describes highlights of the prototype's design and operation. Section 3 summarizes the main results and lessons learned. Section 4 discusses related approaches, and the last section concludes with future work.

2 Our Prototype

The upper portion of Figure 1 depicts the conceptual model of the distributed printing service (end-user workstations, print servers, and printers) with typical print job spooling relations among them. The lower portion of Figure 1 shows the architecture of our prototype. Central is a model-based reasoning server that is accessed via Java applets that run in the web browsers of the diagnosticians. Serving the user interface via the web makes our tool easily and simultaneously accessible to various potential user populations, including help desk, printer support, server operations, and end-users of the printing service[2].

The user-interface component instructs the general purpose reasoning engine to load the models specific to the distributed printing service; models may exist for other distributed services as well [Mar97]. A configuration database for the local site serves to define organizational policies, supported devices, etc. This database is also used as a cache for slowly changing information, since automated diagnosis tends to be slow in large distributed environments without substantial concurrency.

The middle portion of Figure 1 shows the instrumentation methods used for gathering state from the distributed infrastructure. These include SNMP, network *ping*, remote procedure calls to the Win32 printer management API, remote shells for UNIX commands (e.g. *lpstat*), and a protocol for communicating with our "*PrintMedic*" agent. This agent runs on the client's machine when its owner asks for printer system diagnosis support. In the general case, remote agents are needed for distributed diagnosis to improve performance where reference locality is important, as well as to enable remote access to the management information on remote systems that have security or legacy issues.

2.1 Model Information

In modeling the print service components, we conformed as much as possible to the applicable Common Information Model schemas of the Desktop Management Task Force standardization effort [CIM]. We wrote the models in HP's Flipper Modeling Language (FML), a declarative, rule-based language derived from previous HP Labs research [Pel95]. These models are the main structuring tool, containing a typed inheritance hierarchy defining the relevant managed objects and associated rules. The rules express behavior (model), discovery (query) and repair (action). A model's objects can also import or inherit the definitions contained in other models. Figure 2 illustrates the elements of such models.

Our prototype's model consisted of 4400 lines of FML statements. An object hierarchy based on CIM's single inheritance proved to be somewhat troublesome, but workable. In the rule source

[2] Although in this study we did not pursue the end-user population, they could potentially use such a tool to isolate the cause of their troubles before contacting the help desk, ideally accelerating the time to repair. This could also be useful during off-hours.

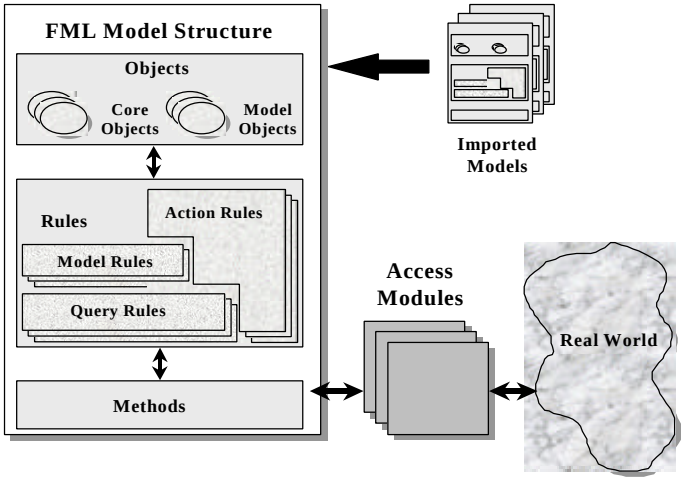

Figure 2: Modeling language components comprised of object-based rules that model behavior, queries and actions, with associated methods for accessing the real world.

sample below, a typical rule, **okToPrintOn** is defined in terms of ten other rules. This rule tests the status of a particular printer from a selected desktop. After checking network connectivity, the rule discovers the spooler(s) involved, and recursively calls other rules that check their health. Finally, the printer itself is checked for correct status.

```
[ComputerSystem m] okToPrintOn [Win32Printer p]
IF
        // Check that the machine itself is on the network
  [m] networkReachable &
        // First check that the printer really does point to a
        // physical printer.
  [p] mapsToPhysicalPrinter [RealPrinter pp] &
        // Now check that the physical printer does not have any
        // critical problems, i.e., offline, out of paper, etc.
  [pp] readyToPrint &
        // Check that the spooler and TCP processes are running on
        // the machine.
  [m] spoolerServiceOK &
        // Check that the printer queue is not paused.
  [p] queueStatusOK &
        // If printer is redirected to Server s (UNIX or NT),
        // recursively diagnose machine s, and printer sp.
  (([p] redirectsTo [ComputerSystem s] usingShare [Printer sp] &
    [s] serverOKToPrintOn [sp]) |
        // If printer directly connected, check job queue
        // for the printer on the machine.
    ([p] directlyPrintsFrom [m] & [p] queueOK)) &
        // Finally, check the printer configuration on machine,
        // the driver version, language, color attributes, etc.
  [m] printerConfigurationOK [p].
```

The bindings in the parameters of these rules are modeled objects; their types are italicized in the listing. Some of these objects, such as *Printer* and *Win32Printer* (derived from *Printer*) are logical components, whereas *RealPrinter* is a physical component. The same rule can be used for monitoring, repair (using 'action' rules to make a configuration correct), and discovery (by leaving a variable unbound).

Most of the rules in the system are shorter than the one listed above. The reasoning engine uses these rules in a backward-chaining system designed for performance in a large system. The system chains backwards from high level goals in order to limit instrumentation overhead for real world facts on a need-to-know basis. The relative simplicity of these rules is due to the existence of a reasoning system. It acts to prove rules by discovering bindings for the rule parameters, using backtracking when necessary.

This approach taken here for networked printer systems is applicable to a broad array of distributed applications and services where inherent structure and relationships can be exploited by modeled behavior. Previous prototypes by our group had demonstrated management tools using model-based reasoning monitoring and diagnosing service failures (e.g. performance degradation) of web-enabled distributed transaction systems [Mar97].

2.2 User Interface Operation

We now describe a typical interaction with the help desk. After a user calls with a printing problem, help desk has the user run the *PrintMedic* agent[3]. It begins by displaying the local host name, which the user reads back over the phone to the help desk[4]. Help desk then directs the diagnosis server to contact the agent, which then enumerates the list of local printer configurations and returns spooler binding information from the client's print management system. The help desk selects which printer configuration to pursue. The reasoning engine uses the models to check the configuration and state of the client, all intermediate servers, and the target printer for consistency. If there is a backlog of jobs on the server, the engine also checks all other servers that are known to spool to the printer, checking that at least one is making forward progress. A scrollable window in the lower half of the user-interface (Figure 3) enumerates all the problems found with any of the distributed components or with their mutual consistency.

The example in Figure 3 shows that the selected target printer, *lj12*, requires user intervention, and that multiple spooling queues destined for that printer are backing up. This example highlights that mutual interference of distributed systems can complicate troubleshooting and that a reasoning tool can help keep track of the implicit interdependencies. The other panels of the user-interface provide for drill-down on a specific printer to an adjustable level of detail, or to the collection of jobs queued at one or all servers that are destined for a specified printer.

One of the user interface panels provides solely for proactive monitoring. This panel is reproduced in Figure 4. When monitoring is enabled, the user-interface periodically updates two sorted lists. The first shows all active server queues, with those at the head of the list diagnosed as “stuck.” The other shows a list of IT supported printers and their top diagnoses, sorted by severity.

[3] This step would be transparent in production systems as support agents are made ubiquitous.

[4] This agent functionality was added in the second month of the experiment, as experience indicated that many users did not know the network name of their desktop system. Scheduling two-iteration release and observe cycles admitted feedback-directed improvements, such as this example.

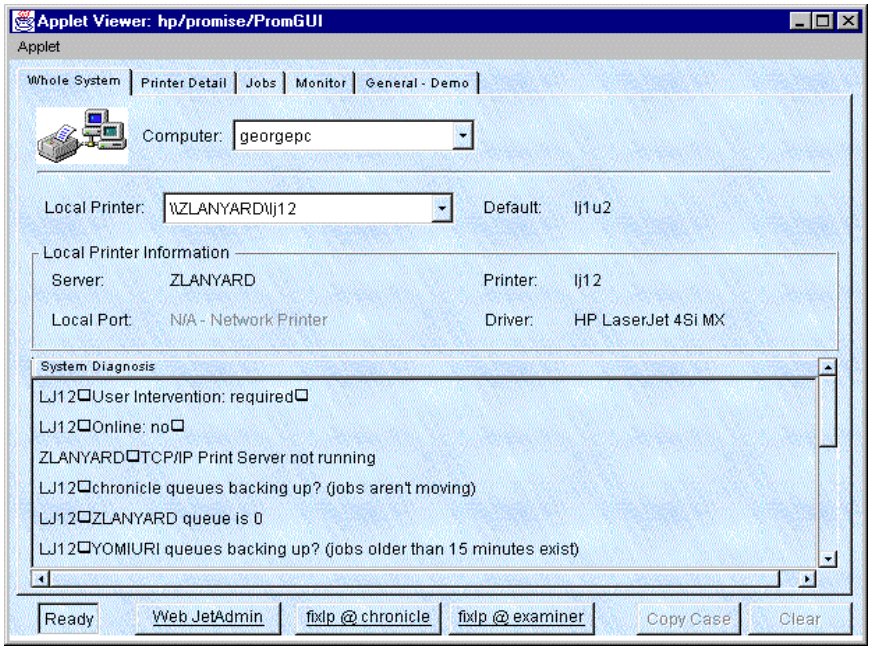

Figure 3: Help Desk user screen showing the whole system diagnosis tab. This summary screen provides for end-to-end system diagnostics. Other tabbed panels provide detail drill-down for specific components.

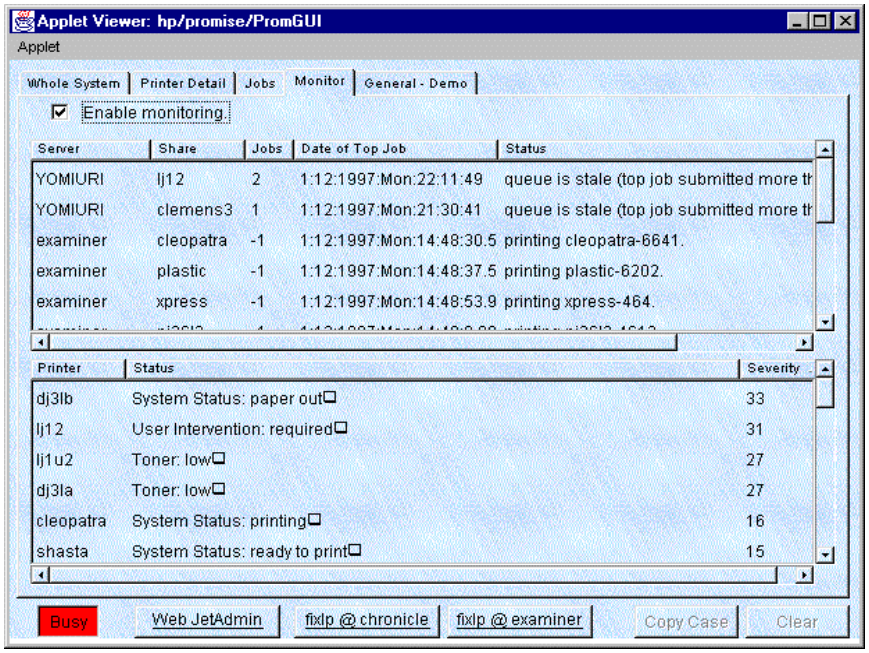

Figure 4: This screen provides continuous monitoring of selected parts of the distributed print infrastructure. The upper panel shows print server queues that are backing up. The lower panel shows troubled printers.

3 Results

In this section we summarize the experience from the use of this tool at the IT help desk, and discuss fundamental issues in concurrency and robustness for any such tool.

The IT personnel who use our tool like it—enough so that they invested scarce resources to keep the prototype running when the experiment ended. During the two-month long experiment, the help desk has been able to resolve 7% more of the printing calls without generating trouble tickets for other members of the IT team. Even in cases where the issue is not resolved immediately, they claim that it aids in quickly identifying the right IT expert to fix the problem.

Particularly appreciated was the tool's ability to gather information from many sources and present it in one place. Prior to the deployment of our prototype, help desk personnel would omit some fundamental checks, such as whether the printer was online, based on their intuition of typical problem calls, but anecdotal evidence suggests that these assumptions often go unchecked for long periods of time, delaying the diagnosis.

In the first month of the experiment, the diagnosis panel showed only the problem diagnoses, sorted by decreasing severity. However, IT personnel wanted to see more details, e.g., when the printer was last re-booted and which checks completed satisfactorily. Their principal reasons for this were to gain confidence in the tool and to help continue diagnosis when the models fail to find the problem. This will inevitably happen since models are generally incomplete. Even if they were complete at deployment, they are unlikely to be updated in lock step with changes in the behavior of the targeted distributed system.

Diagnosis of distributed systems can require a great deal of information about the system. While caching is useful here, startup performance needs to be considered as well. Solutions in this area include predictive caching and incremental processing of results. The implementation of the reasoning engine is currently single threaded; as a result, information collection can take a significant portion of the interactive response time. We are disappointed with the response time of some diagnostic rules, which sometimes take minutes to resolve.

We believe concurrency will be essential in collecting information for real-time diagnosis, particularly where the failure or slowness of any component in responding might delay the overall diagnosis. In this special case of diagnosing *distributed* systems, parallelism can be attained by simultaneously querying multiple remote components for information, or by asking each component for a self-diagnosis.

Writing robust code for diagnosis of distributed components is a challenge and presents continued opportunity for research. Such code is expected to function properly even when the environment is not. Traditional applications in such circumstances can either quit safely or operate at reduced functionality. Additional software engineering research may help with such programming rife with complex exception handling.

4 Related Approaches

We considered different choices for knowledge representation and reasoning available to us in constructing our diagnosis tool. These include case-based reasoning [Kol91], rule-based reasoning [Dav88], decision trees, Bayesian probability networks [Hec95a], and model-based reasoning [Dav84].

Case-based and rule-based representations are methods of reasoning backwards from known symptoms to known problems. In case-based systems, problems are enumerated, and symptoms are explored to distinguish between cases. The method of representing the knowledge and its use in inference are intimately linked. We wanted to exploit the inherent relationships between functioning parts of the printer system, preferring to describe correct behavior of components

rather than documenting all its failure modes. In our experience in the distributed systems domain, this description is often the easier task.

A decision tree is a tool for inference that is constructed to represent a particular domain. Note that as there is also no explicit model of the internal workings of a system, the creation of the knowledge need not require much time, since it can be arbitrarily incomplete. However, when these methods fail to produce a diagnosis, there is no information for the diagnosticians about which components were checked out as being "OK," and where they might next direct their attention. Also, the resulting representations are not modular in any easily maintainable fashion.

Bayesian network technology could be applied in this arena to reason about the uncertainty, successively suggesting the next most useful observation to acquire and the most likely causes of failure for printer systems [Hec95b]. Specifying the many conditional probabilities required for this network is a barrier to knowledge authorship. This project demonstrated that useful diagnosis does not require the explicit consideration of the uncertainty when diagnosing distributed systems' fallible observations.

Recent work in producing an object-oriented variation on Bayesian networks [Kol97] could be leveraged, instantiating network fragment classes for each computer and printer involved in the diagnosis. Further, its object-orientation aligns with our model-based representation. This Bayesian guidance to the model-based inference algorithm might be valuable in domains with significant uncertainty where results are conditional.

By contrast, in model-based reasoning, the specification (or knowledge representation) of the domain may be considered independently of the inference techniques used. De Rocha and Westphall presented a successful model-based approach for proactive network management [Roa97]. Their work focused on discovering trends for heterogeneous network congestion using monitoring and declarative rules of behavior. Eberhardt et al. used models to abstract behavior and component relationships, not for operation diagnosis, but for creating a virtual environment for TMN network system testing during design [Ebe97]. Such practices lead naturally into extending these models for diagnosis in an operational setting.

5 Conclusion

Supporting enterprise-wide distributed systems, such as the printing infrastructure at Hewlett-Packard Labs, is typically a difficult task. IT personnel need to know and use a diverse collection of tools, which individually focus on a small homogeneous piece of the system without knowledge of the interdependencies. As a result, the user must also understand the interrelations among specific components and re-run the tools on all the appropriate components, e.g., all spoolers serving jobs to a printer. We addressed this need with a practical experiment using model-based reasoning to capture the component behaviors for a help desk support problem. This tool markedly increased the quality and speed of a relatively complex distributed printing diagnosis. We believe this approach neatly applies to other distributed service problems.

With the growing business importance, complexity and interdependencies among distributed services, IT departments require management tools that shoulder more of the burden than is traditional. Specifically, the tools should have knowledge of the interconnections and automate much of the diagnosis. Towards that end, great leverage can be gained by providing a machine-readable model of how the system is supposed to function (model-based reasoning), along with a database describing the proper configuration of the local instances. Mobility complicates the effort by requiring this knowledge to be encoded in a parameterized fashion so that correct configurations can be dynamically generated for new locations and situations.

In order to deploy this new breed of tools, third-party software must expose appropriate management information about their components using standardized interfaces and schema, such as the Application Response-time Management [ARM] and the Common Information Model [CIM] standards. Even so, they will need to evolve or be supplemented by better levels of behavioral modeling. The management models need to capture the relationships among components as well as the ways they interact. These advances will be important for the future of automated IT management to support mobile computing, quality of service, and service level monitoring of distributed systems.

Acknowledgements

We are grateful for implementation assistance from Claudio Bartolini, Marco Casassa Mont, Elizabeth Ogston, (HP Laboratories - Bristol) and members of the HP Labs SmallWeb project. We benefited from the irreplaceable knowledge, feedback, and patience of members of HP Labs IT organization. The three anonymous reviewers' comments helped to focus the final version of the paper. Finally, we want to thank Tracy Sienknecht who supported and encouraged our research.

References

[ARM] Application Response-time Measurement standard, *http://www.hp.com/go/ARM*

[CIM] Common Information Model, *http://www.dmtf.org/work/cim.html*, a Desktop Management Task Force (DMTF) proposed Web-Based Enterprise Management (WBEM) standard, *http://wbem.freerange.com*

[Dav84] R. Davis. Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence*, 24:347-410, 1984.

[Dav88] R. Davis and W. C. Hamscher. Model-Based Reasoning: Troubleshooting. In *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, editor H.E. Shrobe, Morgan Kauffman, San Mateo, CA, pp. 297-346, 1988.

[Ebe97] R. Eberhardt, S. Mazziotta, D. Sidou. Design and Testing of Information Models in a Virtual Environment. Integrated Network Management V, pp. 461-472, May 1997.

[Kol97] D. Koller and A. Pfeffer. Object-Oriented Bayesian Networks, In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 302-313, August 1-3, 1997.

[Hec95a] D. Heckerman, J.S. Breese, K. Rommelse. Decision-Theoretic Troubleshooting. *Communications of the ACM*, 38(3):49-57, March 1995.

[Hec95b] D. Heckerman, M. Wellman. Bayesian Networks. *Communications of the ACM*, 38(3):27-30, March 1995.

[Kol91] J. L. Kolodner. Improving Human Decision Making through Case-Based Decision Aiding. *AI Magazine*, 12(2):52-68, 1991.

[Mar97] J. Martinka, J. Pruyne and M. Jain. Quality-of-Service Measurements with Model-Based Management for Networked Applications, Technical Report HPL-97-167 (R1) *http://www.hpl.hp.com/techreports*, Hewlett-Packard Labs, 1997.

[Pel95] A. R. Pell, K. Eshghi, J-J. Moreau and S. Towers. Managing in a Distributed World. *Integrated Network Management IV*, editors A. S. Sethi and Y. Raymond, Chapman & Hall London, pp. 95-105, 1995.

[Roc97] Mareo A. da Rocha and Carlos B. Westphall. Proactive Management of Computer Networks using Artificial Intelligence Agents and Techniques. *Integrated Network Management V*, Chapman & Hall London, pp. 610-621, May 12-16, 1997.

[Rud98] A. Rudenko, P. Reiher, G. J. Popek and G. H. Kuenning. Saving portable computer battery power through remote process execution. *Mobile Computing and Communications Review*, 2(1), January 1998.

[Wil96] B. C. Williams and P. P. Nayak. Immobile Robots: AI in the New Millennium. *AI Magazine*, pp.17–34, Fall 1996.

[Zen97] B. Zenel and D. Duchamp. General-purpose proxies: solved and unsolved problems. In *Proceedings of the Sixth Workshop on Hot Topics in Operating Systems*, May 5-6, 1997.