Practical Optimization Considerations For Diagnostic Knowledge Representation

Patricia Cornwell, Jaap Suermondt, George Forman, Evan Kirshenbaum, Aparna Seetharaman

Hewlett-Packard Laboratories
1501 Page Mill Road, Bldg. 1U
Palo Alto, California 94304-1126
{first_last}@hp.com

Abstract

Using practical criteria for the selection of a diagnostic knowledge representation, businesses can optimize the choice to their specific business goals, processes, and organization structure. This optimization involves compromises, as there are tradeoffs among ease of acquiring the diagnostic knowledge, ease of maintenance, and ease of use in troubleshooting. We present a set of considerations in the form of questions we have found to be practical yet more thorough than the *de facto* short list of considerations used by most organizations. The answers to these questions can be used to derive requirements for the diagnostic knowledge representation. These considerations can be tailored to address particular business and knowledge engineering considerations. While the paper does not provide an exhaustive set of considerations, our experience is that questions like these provide a lower-risk basis on which decisions about diagnostic knowledge representation can be made.

Introduction

Systems have become so complex that diagnosing them can be daunting to even the most experienced customer-support or IT professional. In addition, with intensifying competition in the support field and shrinking profit margins on hardware, there is tremendous pressure on support organizations to reduce costs. In response, organizations are investing in better ways to capture their troubleshooting knowledge and experience and make it an intellectual asset of their business.

These trends and cost pressures have rekindled an interest in technologies for knowledge representation and automated reasoning. Decision-support systems and knowledge bases are being deployed in real-world environments at a greater pace than ever before to reduce the amount of time and expertise it takes to resolve customer problems. However, the current wave of diagnostic applications is being deployed quietly and without much emphasis on the use of AI technology.

Managers in organizations responsible for diagnostic knowledge management may find themselves in unfamiliar territory when facing decisions regarding the appropriate form for diagnostic knowledge representation. Too often, the knowledge-representation decision is made based on short-term considerations rather than on an assessment of overall knowledge management throughout the lifecycle of the diagnostic knowledge assets. If the choice of knowledge representation is made based on short-term convenience, recent announcements by competitors, or "religious convictions," businesses risk failing to meet their business goals (such as customer satisfaction) and incurring unnecessarily high knowledge-management costs.

In this paper, we present sets of considerations in the form of questions that may be used to guide the design or choice of a knowledge representation for diagnostic decision-support systems. Our goal in this paper is not to advocate or disparage the use of any specific knowledge representation. Most knowledge representations have characteristics that make them an acceptable choice for some business situations.

Commonly Used Knowledge Representations in Troubleshooting Support

In the flurry of activity to capture organizational troubleshooting knowledge and provide decision support, managers often think about a few diagnostic knowledge acquisition considerations, but fail to consider long-term costs and benefits. The choice of technology may be based on having a motivated in-house engineer with aptitude in a particular technology or on short-term cost considerations such as avoiding licensing fees for off-the-shelf products.

Knowledge representations fall into groups based on fundamental characteristics. One distinction is between less structured techniques (such as searchable text) and more structured ones (such as decision trees). In this section, we briefly discuss some of the advantages and disadvantages

of such choices, which become apparent at different phases in the knowledge engineering lifecycle.

Less Structured Techniques

1. *Ease of Authoring and Distribution*
Historically, support organizations have operated in a paper-based world in which troubleshooting knowledge is maintained in technical manuals and in other paper documents, including records of previously solved problems. To harness the power of the computer (or more specifically, the disk), many organizations have simply stored these documents in electronic form. The main advantage of representing diagnostic knowledge in unstructured or minimally structured text is that it is easy to author. Textual descriptions by troubleshooting experts about problems they have encountered and how they have solved them fill gigabytes of disk space. Consequently, the scope of the knowledge can be quite broad. For many organizations, it is likely that solutions to the vast majority of troubleshooting problems that face the organization have been captured already and are available in some electronic form.

Another benefit of minimally structured text documents (which may include diagrams) is that they contain rich knowledge that lies "between the lines" of what was actually documented. Some of the knowledge is captured implicitly through the use of rich metaphors that convey valuable information to troubleshooting practitioners.

The captured diagnostic knowledge may be deployed by a wide range of means. Informal processes may include practices like broadcasting e-mail within the troubleshooting team. More structured processes may involve authoring the diagnostic knowledge right into a centralized or replicated file system with search and indexing capabilities or even having a separate organization that certifies each submitted diagnostic document before deploying it to intended users.

The primary authoring disadvantage is that the diagnostic author faces the writer's challenge of where to begin in describing what they want to communicate asnd how to ensure that the resulting document is complete enough. While most people are good at recognizing what would make a customer problem-resolution document easy for them to use again, they do not always have insight into what would make the document easy for others to use. Unstructured diagnostic knowledge can be difficult to use and is notoriously difficult to maintain.

2. *Cost of Use and Maintenance*
Use of a common framework or organization of the documentation supports maintainability. Once users have found relevant documents to read, the common format can improve efficiency in searching within the document. Troubleshooters and knowledge-base maintenance personnel who search an on-line, self-help diagnostic knowledge base are typically faced with many irrelevant documents. Keyword searches, topic hierarchies, and data mining techniques have been employed to help retrieve the documents that are most likely to be relevant, but have been limited in their success. In addition, the lack of structure in the textual knowledge representation limits machine interpretability, restricting opportunities to apply automated techniques for working with the knowledge.

Quality assurance is much more difficult than for some other forms of diagnostic knowledge. Guidelines or checklists for maintainability and quality are more difficult to automate. And precisely because the documents capture richer knowledge in indirect forms (such as the use of metaphors), those involved in maintenance and quality assurance must have a higher degree of domain-specific troubleshooting expertise to ensure they don't degrade the quality of the diagnostic knowledge when they change the document.

Highly Structured Techniques

1. *Ease of Use*
At the other end of the spectrum are machine-interpretable troubleshooting tools that rely on more structured knowledge representations and reasoning algorithms. Integrated with instrumentation of the system under test, all or a portion of the troubleshooting process can be automated. Some companies are investing in AI technologies such as decision trees, Bayesian belief networks, neural networks, rule-based reasoning, and other representations of diagnostic knowledge for creating troubleshooting tools. Because such techniques allow automation of at least portions of the troubleshooting process, they have promise for easing users' efforts significantly.

2. *Cost of Authoring and Maintenance*
Although such representations may be powerful during troubleshooting, building and maintaining diagnostic systems based on them can be a difficult process. Diagnostic knowledge captured as neural networks or Bayesian belief networks relies on machine learning from data and/or interviews of troubleshooting experts by knowledge engineering experts. The availability of clean troubleshooting data forms one barrier to diagnostic knowledge creation. Few managers are willing to devote precious engineering resources to "data scrubbing." In addition, few organizations have a ready supply of knowledge-based software engineers with skills in the design and implementation of diagnostics written in these forms of knowledge representation. Fewer still have a staff of maintenance engineers with these skills.

Because few troubleshooting experts are also skilled in structured knowledge-based systems techniques, the knowledge must be elicited by knowledge engineers with

specialized knowledge representation expertise. Even using this method of knowledge capture, we find that the troubleshooting experts need training in reasoning methodologies in order to provide reliable information. Techniques like fuzzy logic can be misused without developing a solid understanding of the concepts, an understanding we have not found to be widespread among either domain experts or knowledge engineers.

Considerations for Choosing a Diagnostic Knowledge Representation

As suggested earlier in this paper, optimization of diagnostic knowledge representation involves prioritizing criteria across the lifecycle of the knowledge engineering and management process. This section begins by covering criteria that appear to dominate in today's high-pressure customer-support environments. It then enumerates classes of considerations (in the form of questions) that can be used to establish specific criteria that can be optimized across the whole knowledge management lifecycle.

Answers to these questions suggest requirements for the kind of knowledge representation to use during each phase of the lifecycle. The optimization process prioritizes the requirements that are most important to be met by the knowledge representation. Generally, if a troubleshooting module is authored according to one knowledge representation, it will be maintained and used in troubleshooting without translation to another knowledge representation. However, a business may instead decide that "optimization" will be accomplished by letting the authors create new diagnostic knowledge in a format that is comparatively easy for them to write and then having that knowledge translated into a form that is easier to maintain and/or use.

Typical Concerns in Selecting a Knowledge Representation

The often-implicit process of selecting a knowledge representation is typically dominated by a limited set of issues:

1. Would this technology make any improvement in troubleshooting (no matter how small) over what we do today?
2. Is there a champion for this technology within the organization? Is there in-house expertise in this technology?
3. How can we avoid losing our current investment? How do we leverage any knowledge we have already captured? What will run on our installed infrastructure?
4. How can we get something deployed quickly? What can we buy off the shelf that the vendor will support?
5. Is there a marketing advantage to using this technology? Is it what our competitors are using?

While these questions reflect significant concerns, they focus almost exclusively on minimizing the cost of diagnostic knowledge acquisition and time-to-market. The subsections that follow extend this set of questions with a more systematic look at the knowledge lifecycle, specifically at how the knowledge is obtained, maintained, and used. Answers to these questions and their relative priorities will depend on the organization's business goals, processes, and structure.

How to Apply the Consideration Questions

The consideration questions below are based on (1) experience in optimization across typical knowledge engineering and management lifecycles that include knowledge acquisition, maintenance, and use; and (2) analysis of existing and desired practices in customer support knowledge management within the Hewlett-Packard Company.

In previous applications of this technique to the optimization of software knowledge (Cornwell 1996), a much more extensive list of questions was used over a three year period in consulting with diverse HP businesses. Most organizations applied a relevant subset of the questions, scanning the entire set to gain a better intuitive sense of what would be the most critical considerations for their business. Inevitably, the organizations identified their own additional questions, catalyzed by the question list.

We include a brief example of the practical application of the considerations to making a choice of diagnostic knowledge representation following the enumeration of questions. For each phase of the knowledge engineering and management lifecycle, the questions are organized to cover (1) business goals and strategy, (2) knowledge engineering process, and (3) organization structure and available resources.

How will the knowledge be acquired?

1) Business Goals and Strategy
   a) *Acquisition Strategy*: How will the business acquire diagnostic knowledge? Is there management commitment to purchase off-the-shelf diagnostics? Can/will some diagnostic knowledge be learned from available customer support data? How much of the diagnostic knowledge will be proprietary to the producer of the troubleshooting system? How much will be proprietary to the user of the system?
   b) *Update Frequency*: What is the goal for the time between problem discovery and deployed diagnostics?
   c) *Coverage*: How many diagnostics will there be? How many problems will be covered? How many possible symptoms or questions can/must be captured?

d) *Integration*: Is there a goal for all diagnostic knowledge to be integrated? Are there intentions to provide seamless extensions to handle the newest systems and discovered troubleshooting knowledge?

e) *Reuse*: Are diagnostic authors expected to ensure that new diagnostics do not duplicate existing diagnostic knowledge? Is there or will there be a diagnostic component base promoted for use in building more complex diagnostics? How well does the method for reusing existing diagnostics match the skills of the diagnostic authors and integrators? What is the desired granularity of a single diagnostic component?

f) *Accountability*: Are managers held accountable for aligning project priorities with these business goals or are they rewarded for taking the route to quick/cheap diagnostic creation, regardless of the *stated* business priorities?

2) Knowledge Acquisition Process

a) *Knowledge Sources*: How does the organization intend to acquire new diagnostic knowledge: human authoring, machine learning, purchasing? Is diagnostic knowledge about the domain (kinds of systems to be diagnosed) already available in another form (e.g., manuals, documented diagnostic protocols, descriptions of previously solved problems)? Where will the data come from if machine learning will be used? Do diagnostic modules or applications from third parties need to be integrated with the diagnostic knowledge base? Are troubleshooting experts readily available as domain experts?

b) *Separate/Embedded Authoring Process*: Is diagnostic authoring an integrated part of the overall customer support process, where the business expects diagnostics to "fall out" of helping customers solve their problems? Are troubleshooting system development engineers expected to generate diagnostics as a side task to building a useful infrastructure? Is there a need to capture meta-knowledge about the diagnostic knowledge, e.g., by whom, when and why the knowledge was created or acquired?

c) *Available Tool Support*: What knowledge authoring tools are in current use by diagnostic knowledge authors? Which kinds of diagnostic authoring aids could be used (e.g., templates, examples, generators, etc.)? Will managers support the purchase and maintenance of authoring tools that are better matched to the available authors and the intended use of the diagnostic knowledge they create?

3) Organization Structure and Available Resources

a) *Authors*: What knowledge representation is natural for the authors to use in codifying diagnostic knowledge? What training is available to authors? Is diagnostic authoring a primary responsibility for the authors? In how much time is an author typically expected to create a new diagnostic module or tool? Are authors experienced knowledge-based software engineers? Could the authors capture diagnostic knowledge in executable code or scripts? Do/can authors leverage existing diagnostics or other forms of diagnostic knowledge in authoring new diagnostics? How do the authors initially capture diagnostic knowledge: control flow of questions and answers? symptoms and root causes? lists of possible root causes, given a symptom? Do/can they characterize correct behavior of systems or only problems and resolutions?

b) *Knowledge Integrators*: Do knowledge integrators have responsibility for integrating third-party diagnostic components or applications? Do they have responsibility for ensuring knowledge consistency across the knowledge base?

c) *Troubleshooting Domain Experts*: What knowledge representations do the troubleshooting experts use to describe troubleshooting diagnostics? Do these domain experts think about troubleshooting in terms of reasoning under uncertainty? Are product development or support engineers available for authoring diagnostic knowledge about the products that are part of the troubleshooting domain?

How will the knowledge be maintained and supported?

1) Business Goals and Strategy

a) *Update Frequency:* Are those who do maintenance primarily responsible for delivering the updates to the users? How rapidly must they be able to understand the diagnostic, modify it, test it, and release a new version? What is the volume of diagnostics going through maintenance? Is maintenance done on a pro-active or reactive basis? Is it done at all? How frequently does the class of systems under test change? How frequently does the troubleshooting system change?

b) *Support:* Is support provided for the use of the troubleshooting system or diagnostic knowledge?

c) *Accuracy:* Does the maintenance process include testing the accuracy of the diagnostics? Are users being promised high accuracy?

d) *Coverage:* Do the maintenance tools ensure that changes to the diagnostics do not reduce coverage? Does maintenance sometimes include extension of the coverage for the diagnostics? Do diagnostics for multiple versions of the system need to coexist in the knowledge base?

2) Knowledge Maintenance and Support Process

a) *Separate/Embedded Maintenance and Support Procesess:* Is knowledge maintenance a separate process? Does the business use maintainability

metrics? When and how does diagnostic knowledge become obsolete and get removed from the knowledge base?

b) *Available Tool Support:* Are tools for maintenance available commercially or are they developed in house? Will managers support the purchase and maintenance of knowledge-maintenance tools? Do those maintaining the diagnostic knowledge need tools that detect and/or maintain dependencies among parts of the knowledge base? What is available for those providing usage support?

3) Organization Structure and Available Resources

a) *Diagnostic Knowledge Maintainers:* Is maintenance the primary responsibility of those doing the maintenance? What training or skills do they have? What training can they get? Are knowledge maintenance people troubleshooting experts? Can they resolve inconsistencies in the diagnostic knowledge?

b) *Diagnostic Knowledge Usage Support:* Are there people or tools available to help users of the diagnostic knowledge, either to learn how to use the knowledge or to help them find appropriate diagnostic information? How available are the people/tools to the troubleshooters? Are troubleshooters willing to seek out usage support or must the diagnostic knowledge be "intuitive" to use?

How will the knowledge be used?

1) Business Goals and Strategy

a) *Intended Users:* Who is targeted to use the troubleshooting system or diagnostic knowledge? Is there a single targeted type of user, such as expert troubleshooting professionals, less-knowledgeable troubleshooters, or system end users? Are there other targeted uses of the diagnostic knowledge, such as training customer support engineers, design engineers, or quality assurance engineers?

b) *Troubleshooting Domain Characteristics*: Is it possible to automate some diagnostic tests? Are some faults only diagnosable programmatically? Are there important cost differences among diagnostic tests or repairs? Are the costs dependent on the order in which the tests or repairs are executed/asked? Do the diagnostics need to contend with multiple faults? Is there connectivity between the user and the system being diagnosed?

c) *Speed of problem resolution*: How much time does the troubleshooter have to diagnose the problem, using the diagnostic knowledge? Is there previously-obtained knowledge that could be used to expedite the problem resolution?

d) *Accuracy*: What are the consequences to the troubleshooter if the problem is not solved correctly the first time? What are the consequences to the end user?

e) *Coverage*: How many of the potential system faults need to be covered by the diagnostics? (E.g., have customers been promised a minimum degree of coverage?) Is the desire to find the fault or to narrow it down to a small set of candidates for the user to choose from?

2) Troubleshooting Process and Other Uses

a) *Separate/embedded troubleshooting process*: Is the primary task diagnosis or repair? Is troubleshooting the primary job of the troubleshooter who uses these diagnostics? Do the troubleshooting data (e.g., information specific to a troubleshooting session) need to be seamlessly integrated with other customer support knowledge?

b) *Available tool support*: What customer support or self-support troubleshooting tools are in current use by the troubleshooters targeted to use the diagnostic knowledge? Does the diagnostic knowledge need to integrate with these tools? Is it intended that the troubleshooting results are fed back to product developers or other parts of the business?

3) Organization Structure and Available Resources

a) *Troubleshooting professionals*: Are the diagnostics targeted for use by troubleshooting professionals, such as full-time IT support staff? What depth of expertise do the troubleshooters generally have? What is the variation in depth of expertise? Are there tasks that only certain troubleshooters should be trusted to perform accurately or safely? Would the troubleshooter be irritated by questions for which the answers "seem obvious" or which the troubleshooter can anticipate will not be useful? Do the troubleshooters prefer automated diagnostics or interactive ones? Do they want to be able to "jump ahead" to more specific questions? Do they take pride in the diagnostic knowledge they have stored in their human memory? Do they prefer to do their jobs with diagnostic knowledge bases? Diagnostic troubleshooting tools? Will troubleshooters want to "look inside" the diagnostics? Will they be expected to infer knowledge from the choice of knowledge representation? For example, will they be expected to understand that the diagnostic is one that characterizes uncertainty?

b) *Inexperienced troubleshooters*: Are the diagnostics targeted for use by people whose primary responsibilities are in other fields? Will the users need to be walked through the diagnostic steps in detail? Will the users understand standard diagnostic troubleshooting terminology? How much knowledge does the user have of the system under diagnosis? Do the targeted users prefer an interrogative style of troubleshooting (question-answer dialogue), use of automated diagnostics, visual

models of the system under diagnosis? (This implies that user studies have been conducted.) Do the troubleshooters need to be able to defer questions? How comfortable are the troubleshooters with providing an answer like "I don't know"? Does the diagnostic system need to be able to not believe what the troubleshooter tells it (i.e does the diagnostic system need to address unreliable information)? Is there a range of preferences among diagnostic knowledge users for how the troubleshooting session is conducted?

c) *Organizational Learning/training:* Will the diagnostic knowledge be used to train new troubleshooters? Help Desk professionals? Diagnostic authors? Others in the business? Will the diagnostic knowledge be mined to create models of the family of systems that can be diagnosed? Models of the diagnosis process used? Other forms of derived knowledge?

d) *Intelligence Gathering:* Does the knowledge representation need to capture intelligence about the system or the use of the system?

Example

This example is intended to illustrate the use of some of the above considerations in identifying criteria for the choice of knowledge representation.[1]

Online Self-Support

A large commercial business plans to provide product customers with web-based access to diagnostic knowledge. The company manufactures many product families, with new product introductions every few months for many of these families. Customers have requested a web-based tool for getting answers to their most common questions about products that appear not to be working correctly. Currently, customers have to call a Customer Care Center for these answers. (Note that the answers below are clustered by topic across the lifecycle. The vagueness and generality of the answers is typical of what an organization will provide.)

Business

- *Intended Uses:* The diagnostic knowledge is only intended to be used for the web-based customer self-support service. It will leverage from other diagnostic knowledge sources, but will not be used for other purposes.
- *Acquisition Strategy:* The company must produce its own diagnostics, as the products are ones that it manufactures. The diagnostic knowledge will be proprietary to the company.
- *Maintenance Strategy:* Product support engineers and quality assurance engineers who find a problem with the diagnostic has permission to modify it. Those who take calls in the Customer Care Center from customers who were unable to get the diagnostic to work correctly will report the problem to the diagnostic author.
- *Update Frequency*: When a new product is announced, the company has made a commitment to have an intial set of diagnostics available on the web covering common installation problems and differences between this product and previous products in the family. New products in each product family are announced with an average frequency of twice a year. Problems with diagnostics are classified with a priority. Urgent ones get immediate attention from the author.
- *Coverage:* For each product, there may be 5 to 20 distinct types of problems for which diagnostic knowledge needs to be made available. Diagnosis may be as simple as checking 2 or 3 system behavior variables (e.g., power on, cable secure, error message number) or may involve answers to over 50 questions. Coverage analysis will not be performed.
- *Accuracy:* The QA process serves as some validation of the accuracy of the diagnostics. However, the diagnostics are not mission-critical components.
- *Support:* No support will provided for using the diagnostics. If users can't get the online diagnostic system to work as they expect, they can call the Customer Care Center for the traditional kind of support.
- *Integration:* It would be nice for the diagnostic knowledge to be integrated (appear uniformly accessible with information sharing between diagnostics), but it isn't essential for the success of the web page.
- *Reuse:* Since most products share many characteristics with previous products in the family, the diagnostics for each could be based on common diagnostic knowledge. Management does like the idea of reuse to reduce the cost of producing diagnostics for new products.

Processes

- *Knowledge Sources:* Product design and support engineers in the product R&D organizations will have the primary responsibility for providing the diagnostic knowledge. The company does not intend to use machine learning immediately.
- *Separate/Embedded Process:* QA engineers will author diagnostics to test the quality, performance, and usability of the product with consultation from the product design and support engineers. Diagnostic maintenance is considered part of customer support.
- *Available Tool Support:* The diagnostic authors typically use a text processor to capture the diagnostic knowledge and a combination of programming language tools (e.g., C++, perl) to write the diagnostics. Managers have no budget for new tools, and have a

[1] Another example is diagnosis systems for VLSI manufacturing described in (Hekmatpour 1993).

policy of only purchasing commercially-supported tools.

Organization Structure and Available Resources

- *Authors*: Most authors are already comfortable writing decision trees in a standard programming language. The quality assurance engineers have traditionally authored a single diagnostic module for each new product, capturing all of the anticipated problems by the observed problem (e.g., blue screen, loss of audio output). Most products are considered to be too complex for anyone to be able to do a complete model of correct behavior. Managers are willing to provide up to one week of training for all authors if a standard knowledge representation and reasoning technique is adopted. Authors have about two months to create the initial set of diagnostics that must be released with new product availability. No significant number of diagnostic authors (quality assurance engineers) or domain experts (product design and support engineers) have formal training in knowledge-based software engineering methods. Little if any attention will be paid initially to knowledge consistency.
- *Troubleshooting Domain Experts*: The product design and support engineers who serve as domain experts are a mix of hardware, mechanical, and firmware engineers.
- *Diagnostic Knowledge Maintainers*: Maintenance of the knowledge base will not be the primary responsibility of any of the people involved in the maintenance process.
- *Users*: The people using the web-based customer self-support troubleshooting tool will generally be end users of the products or, for business customers, their IT support engineers. Some users will want to jump ahead or offer a guess about what might be wrong, but initial troubleshooting tools do not need to support that.

Conclusion

The quick turn-around time for producing new diagnostics is a critical factor in selecting appropriate diagnostic knowledge representation. The two-month development time argues for a simple choice of knowledge representation. The company should probably avoid knowledge representations and reasoning methods that involve creating sophisticated models of system behavior, both due to time constraints and to the likely skills of the authors. To determine whether knowledge representations that capture probabilities or other forms of uncertainty could be used, it would be essential to assess the particular skills of the quality assurance engineers. It would be less likely that most product support engineers would have the kind of knowledge and skills needed to quickly produce those kinds of diagnostics, or that they could pick up the skills with the limited amount of training budget the business has approved.

However, the company may want to invest in training for the use of case bases and less complex forms of case-based reasoning. Because diagnosis is focused on product problems rather than more complex system problems, CBR might be adequate. The availability of a few commercial CBR authoring tools may make it attractive to this company.

However, when customers start to ask for more sophisticated web-based troubleshooting tools (e.g., ones that look at networks, distributed applications problems, or multiple-fault situations), they may also need to rethink their choice of knowledge representation and diagnostic reasoning technique. In addition, customers may quickly begin requesting downloadable automated diagnostics that can answer the questions they don't understand or don't know how to answer.

Perhaps the biggest challenge will be moving away from decision tree technology, because it is notoriously difficult to maintain. Ease of maintenance is critical in the organization structure and processes described.

Due to the culture of the web, it is possible that there would be customer acceptance or even pull for text documents in the form of FAQs. This form of diagnostic knowledge would require the company to rethink who should serve as authors and maintainers of this form of knowledge. It could be that engineers in the Customer Care Center would need to be given the job of generating FAQs.

Optimization in the choice of knowledge representation in this situation could include innovation or invention of a new or hybrid knowledge representation and reasoning technology that will meet more of the needs of this business and its customers.

Discussion

System diagnosis is a promising area for applications of AI research, and AI can supply promising technology for customer support organizations. The domain experts in the customer support field are reasonably technical, which may help bridge the communication gap that has existed in many other application domains of AI technology. Furthermore, the knowledge in troubleshooting is reasonably explicit, well defined, and deep, which allows for the development of highly-accurate diagnostic systems.

The good match between customer support problems and AI diagnostic solutions has led to an increase in commercial implementation efforts. However, this attraction to AI technology by customer support organizations brings risks for AI as well. A similar burst of enthusiasm occurred in the early 1980s, and a combination of unrealistic predictions and a lack of examples of real-world successes resulted in a backlash. It is essential that AI experts demonstrate how the technologies can be applied in real-world settings and on a realistic scale.

Much AI research in diagnostic reasoning has focused on issues of flexibility and performance while not focusing much on authoring and knowledge maintenance issues. However, the knowledge representations that dominate in real-world settings are those that are easy to author, with much less emphasis on flexibility and performance. This choice is typically based on short-term needs and availability.

A well considered tradeoff should form the basis of an investment in a particular knowledge representation. In this paper, we have provided a number of questions that can be used to characterize business needs for diagnostic knowledge representation for use in troubleshooting systems and customer support.

To take advantage of the current interest in AI and to make a significant business contribution, AI researchers must address the needs of customer support organizations. For many flexible and diagnostically accurate knowledge representations, the gap between the needs for the troubleshooting application and the properties of the knowledge representation can be bridged through attention to the choice of representation and by providing the right tools. Speaking from both an AI and a troubleshooting perspective, it is in the interest of our field to bridge that gap.

Acknowledgments

We thank our colleague Joe Martinka for many relevant discussions on the comparison of knowledge-representation techniques.

References

Christadoulou, E. and Keravnou, E.T. 1997. Integrating multiple problem solvers in knowledge-based systems, *The Knowledge Engineering Review*, Vol. 12:2, 181-207.

Cornwell, Patricia. 1996. HP Domain Analysis: Producing Useful Models for Reusable Software, Hewlett-Packard Journal, August, 46 – 55.

Hoffman, Christopher et al., Analysts. 1998. Technical Support Knowledge Systems: A Comprehensive Market Landscape of the Problem-Resolution Industry, International Data Corporation.

Hekmatpour, Amir. 1993. A Methodology and Architecture for Interactive Knowledge-Based Diagnostic Problem-Solving in VLSI Manufacturing. Ph.D. diss. In Electrical Engineering (Computer Engineering), University of California, San Diego.

Studer, Rudi et al. 1998. Knowledge Engineering: Principles and Methods. In *Data & Knowledge Engineering*, 25: 161 – 197.