In Proceedings of the 5th International Topical Meeting on Nuclear Plant Instrumentation Control and Human Machine Interface Technology
(pp. 439-446). Albuquerque, New Mexico

Toward Designing for Trust in Database Automation

Pierre P. Duez and Greg A. Jamieson

Cognitive Engineering Laboratory, University of Toronto, 5 King's College Rd., Toronto, ON, M5S 3G8
{duez, jamieson}@mie.utoronto.ca

Abstract – *Appropriate reliance on system automation is imperative for safe and productive work, especially in safety-critical systems. It is unsafe to rely on automation beyond its designed use; conversely, it can be both unproductive and unsafe to manually perform tasks that are better relegated to automated tools. Operator trust in automated tools mediates reliance, and trust appears to affect how operators use technology. As automated agents become more complex, the question of trust in automation is increasingly important. In order to achieve proper use of automation, we must engender an appropriate degree of trust that is sensitive to changes in operating functions and context. In this paper, we present research concerning trust in automation in the domain of automated tools for relational databases.*

Lee and See [1] have provided models of trust in automation. One model developed by Lee and See identifies three key categories of information about the automation that lie along a continuum of attributional abstraction. Purpose-, process- and performance-related information serve, both individually and through inferences between them, to describe automation in such a way as to engender properly-calibrated trust. Thus, one can look at information from different levels of attributional abstraction as a general requirements analysis for information key to appropriate trust in automation.

The model of information necessary to engender appropriate trust in automation [1] is a general one. Although it describes categories of information, it does not provide insight on how to determine the specific information elements required for a given automated tool. We have applied the Abstraction Hierarchy (AH) to this problem in the domain of relational databases. The AH serves as a formal description of the automation at several levels of abstraction, ranging from a very abstract purpose-oriented description to a more concrete description of the resources involved in the automated process. The connection between an AH for an automated tool and a list of information elements at the three levels of attributional abstraction is then direct, providing a method for satisfying information requirements for appropriate trust in automation.

In this paper, we will present our method for developing specific information requirements for an automated tool, based on a formal analysis of that tool and the models presented by Lee and See. We will show an example of the application of the AH to automation, in the domain of relational database automation, and the resulting set of specific information elements for appropriate trust in the automated tool. Finally, we will comment on the applicability of this approach to the domain of nuclear plant instrumentation.

I. INTRODUCTION

As the state of the art of technology advances, our interaction with complex work environments is becoming increasingly mediated by computers and assisted by automation. Automation is used to reduce operator workload, through tasks such as managing complex processes, filtering or interpreting data, or helping operators make decisions. Automation is becoming more sophisticated, and more ubiquitous in its application.

However, the availability of automation does not ensure appropriate operator reliance on it. Especially in safety-critical systems, reliance on automation beyond its designed-for functionality (or in circumstances where it would be less effective) can have a significant negative impact, with potential financial, safety or mortal consequences if the automation fails. Conversely, a lack of reliance on properly-functioning automation can result in reduced efficiency and degraded performance. Manual interaction with a system might also reduce safety, if the automation is better-able to handle a task than a human operator.

Although trust is not the sole factor affecting reliance, it has been found to mediate operator reliance on automation [1, 2]. Therefore, in order to achieve appropriate reliance on automation, we must investigate methods of engendering appropriate operator trust in these automated tools.

I.A. Appropriate Trust

Appropriate trust in automation involves two dimensions. [1] The first is the *calibration* of trust, or the degree to which operator trust in automation corresponds to the automated tool's capability. If an operator's trust exceeds the automation's ability, this *over-trust* might result in inappropriate reliance on the automation. This

can include using the automation to achieve goals beyond its scope, or engaging it under conditions in which it is not designed to operate. An example of over-trust, and thus over-reliance, is relying on one's word-processing spell-check in lieu of proofreading a document; although incorrectly-spelled words will be found, homonyms or fragmented sentences will be overlooked.

Conversely, if an operator's trust in automation is itself exceeded by the automation's capability, then this *distrust* can cause the operator not to rely on the automation in circumstances when it would behave correctly (and might in fact exceed the operator's performance). For example, word processing tasks such as mail merging (creating a large number of form letters by automatically generating multiple letters based on names and addresses in a database) can be done much more quickly (and perhaps more accurately) by built-in automation than by hand.

The second dimension of appropriate trust is *resolution* of trust. This dimension describes the operator's sensitivity to changing conditions and applications on the ability of the automation. A high resolution of trust indicates that the operator is aware of changing conditions, and of their impact on the automation's ability. An operator demonstrating low sensitivity would be less aware of changing conditions, and would be more prone to over-trust or distrust of the automation under certain circumstances.

I.B. A model for trust in automation

A model has been developed [1] that describes a nested, closed-loop feedback model for trust in automation. According to the authors of this model, trust is an *attitude* that is informed by *beliefs* about the automation and its abilities. (In the nested loop, this trust attitude can serve to reinforce or reinterpret some of the initial beliefs.) This attitude of trust then informs an *intention* of whether or not to engage the automation. If the operator does decide to use the automation, a specific *reliance action* takes place to engage it. The automation then operates, with information about the automation being provided via a *display*[1]. The results of the automation's actions then serve to add, strengthen or challenge the operator's beliefs, thereby closing the loop.

At every stage of this feedback loop, however, are several factors that can affect an operator's beliefs, attitudes, intentions and actions, as well as the operation of the automation, that are beyond the control of the automation's designers. Figure 1 provides a summary of these external factors. Under certain circumstances, some

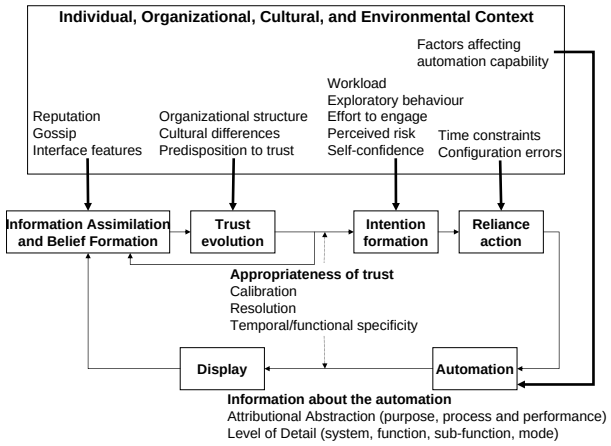

Figure 1: Lee and See's [1] model of trust and reliance

of these factors (such as the organizational structure in which an operator works, or the effort required to engage the automation) can be accounted for, or planned. Other factors, such as gossip and individual predisposition to trust, are notoriously difficult to predict or account for.

Although the degree to which a designer can control operator's trust in automation is limited, there is still an opportunity to influence this trust. Decisions made about the degree of automation in a tool [3] and how operators interact with this automation will affect operator trust and reliance. Guidelines and procedures mandating reliance upon automation can potentially circumvent the issue of trust altogether (although a measure of trust will form as operators learn about, and grow accustomed to, the automation).

In addition to designing the automation with trust in mind, it is worth investigating what information can be conveyed to operators in order to increase the appropriateness of their trust in the automation. This approach can be used in conjunction with automation design models and methods, and provides the focus for the research described in this paper.

I.C. Levels of attributional abstraction: Purpose, Process and Performance

Automation can be described at different degrees of attributional abstraction [1]. Information at different levels of abstraction can serve to engender an appropriate degree of operator trust in automation. Purpose-, process- and performance-related information can each serve as the basis for operator trust in automation. Additionally, inferences drawn between levels of attributional abstraction can be used by operators to reinforce or supplement information at a given level.

Purpose-related information, the most abstract of the three levels, refers to the purpose for which the automation was designed. It describes the problem (or problems) the automation was designed to solve, and the

[1] In this case, "display" is a very general term. It can mean an interface on a CRT, or it could be information displayed through documentation, logging, etc.

goals it attempts to achieve. Information at the purpose level corresponds to, among others, notions of fiduciary responsibility [4] and intentions [5] – among others – as bases for trust. With a proper understanding of an automated tool's purpose, an operator is less likely to trust it to perform a task outside its specifications.

Process-related information refers to the process or algorithms by which an automated tool achieves its purpose. This information serves to demonstrate how automation achieves its goals given the resources and information available to it. Process-related information corresponds to concepts such as persistence [4] and integrity [6] that form bases for trust. An understanding of the process (or processes) underlying automation can alert the operator to potential pitfalls should they try to engage the automation under certain circumstances (e.g., lack of resources).

Performance-related is the most concrete category of information. It refers to information describing the present and past behaviour of the automation. It corresponds to notions of competence [4] and ability [6] as bases for trust. Knowledge of performance-related information, for instance, the behaviour of the automation the last time it was run under certain circumstances, can serve as a definite indicator to the operator of the automation's ability to perform the next time those circumstances are encountered.

These three categories of attributional abstraction describe the information that can serve to engender appropriate operator trust, effectively acting as a high-level information requirements analysis. However, they do not provide a concrete description of the information necessary for appropriate trust. We have therefore developed a method for determining precise information requirements for a given automated tool, at all three levels of attributional abstraction.

II. THE ABSTRACTION HIERARCHY

The Abstraction Hierarchy (AH; [7]) is an analytic tool used to model a system. It describes a system at successive levels of abstraction. Rasmussen [8] observed that in a single task, operators might switch between several meaningful and useful internal representations of a system. The AH thus provides a formative model of a system. The top (most abstract) level of the AH describes the purpose of the system – what it was designed to achieve. At the opposite end, the bottom (most concrete) level describes the components that constitute the system, and that (through internal processes) are used to achieve the system's purpose. Traditionally, in domains such as process control, aviation or computer networks, this lowest level of abstraction is physical. However, in purely virtual systems (such as a software tool), this lowest level of abstraction can describe the resources (e.g., memory and disk space) necessary for the system to function. Middle levels of abstraction describe internal functions and processes (at varying degrees of abstraction) by which components that make up the system achieve the system's purpose.

Functional means-ends links are identified between levels of abstraction. These links connect resources or internal processes to the system goals, and vice versa. When following a link to a level of increased abstraction, one answers the question of *why* an element is part of the system, that is, what contribution it makes (directly or indirectly) to the system's purpose. Conversely, following a link to a level of lower abstraction explains *how* a goal or process is implemented (or achieved).

Constructing an AH requires a thorough understanding of the system being analyzed. Comprehensive descriptions and methods for developing an AH have been developed elsewhere [9, 10], and will not be reproduced here.

II.A. Using the Abstraction Hierarchy to Identify Information Requirements

As discussed above (and in [1]), appropriate trust in automation involves an understanding of the automation's abilities (calibration) under varying circumstances (resolution). To that end, information at three levels of attributional abstraction describes the automation in such a way as to facilitate such an understanding.

The connection between the AH and these categories of information is a straightforward one. If one develops an AH scoped to the automation (that is, an analysis of the subset of the full system that is affected by, and affects, the automation), then the top level of this analysis will correspond directly to the highest level of attributional abstraction. Subsequent levels correspond to process-related information. Finally, current and past values for the components and processes that constitute this subsystem describe the relevant performance-related information.

As described earlier, the top level of the AH describes the purpose that a system was designed to achieve. In an AH scoped to the automation, the "system" in question is that that which is relevant to the automation. The top level describes the purpose of the subsystem, from the point of view of this automation; that is to say, it describes the *purpose* that the automation was designed to achieve.

Each lower level of the AH[2] describes the functions and processes by which the elements of the previous level (starting with the top, purpose-related level), or description, of the system are implemented. Thus, these lower levels all outline (to varying degrees of abstraction)

[2] There is no set number of levels for an Abstraction Hierarchy, although they typically have between four and six.

the *process* by which the automation achieves its purpose, based on the resources it requires. The bottom level of the AH, that in physical systems describes "the physical conditions for purposeful function of a system" [8, p.10] (and in pure software systems describes the virtual resources necessary for proper operation of the rest of the [sub-] system), also describes part of the automation's process. These resources, physical or virtual, are the inputs to the automation-related subsystem, without which the automation cannot achieve its purpose.

Finally, the current and past values of all elements (resources, functions and processes) comprise the set of information pertinent to the automation's *performance*. To understand automation's performance, not only is it important to measure the degree to which its purpose is met; one must also know the degree to which its internal processes or algorithms behave as expected. A more appropriate degree of trust in an automated tool can be engendered if the operator is aware of the collateral impact that the automation's operation has on the rest of the system. Current information allows an operator to track the behaviour and performance of automation that has been engaged; historical (or typical) information on the performance of automation can inform the attitude of trust that might lead to the operator engaging that automation in the first place.

As stated earlier, the development of an AH requires a thorough understanding of the system being analyzed. However, this thorough approach leads to a rich set of information elements that satisfy the requirements analysis for appropriate trust in automation. Moreover, this understanding is necessary to develop the automation itself; identifying the information requirements at an early stage in the automation's lifecycle therefore represents only a small additional effort.

III. CASE STUDY: DB2 AND THE SELF-TUNING MEMORY MANAGER

One application to-date of this method of identifying information requirements for appropriate operator trust in automation is in the domain of relational databases. DB2® is IBM®'s relational database software. In DB2 Version 9.1 for Linux®, UNIX® and Windows® (DB2 V9.1), IBM introduced the Self-Tuning Memory Manager (STMM; [11]). This is an automated tool designed to automatically monitor memory usage within a database instance, and dynamically allocate it in such a way as to alleviate memory-related bottlenecks. We applied our method of identifying information requirements by first conducting an analysis of the STMM, and then identifying the information elements based on the resulting AH.

III.A. Abstraction Hierarchy of the STMM

The STMM was developed in order to optimize memory allocation among the different memory consumers, in order to improve query time on large workloads. The STMM operates by engaging in periodic tuning cycles. In each cycle, the STMM compares the calculated resource requirements from tuned Memory Consumers (MCs) – elements within DB2 that take up a certain amount of memory – and then reallocates memory between these, first by shrinking one MC and then (once the memory has been disclaimed) by growing another. There is no explicit interface for the STMM. Rather, it is engaged when the `SELF_TUNING_MEM` configuration parameter is set to `ON` and two or more tunable MCs are set to `AUTOMATIC` size.

The top level of the AH, scoped to the STMM, describes the purpose for which the STMM was designed. The purpose of this automation is threefold: to optimize the total memory used by DB2, to optimize the allocation of memory within DB2 among the different MCs, and to converge on an optimal (or near-optimal) state in a timely manner. Thus, at a high level of abstraction, the STMM can be described as a affecting a system whose three purposes are: optimal total memory usage, optimal memory allocation, and quick convergence. This therefore constitutes the top level of our AH (see Figure 2).

The second level of the AH describes the system in terms of abstract functions and balances. In the language of this level, the subsystem of DB2 is controlled through the balancing of memory pages. Every tuning cycle, MCs might disclaim pages of memory. This memory is either temporarily (until the next phase of the tuning cycle) or permanently (if the total database memory utilization shrinks) unclaimed. If the MCs succeed in disclaiming the memory within the tuning cycle (some MCs take longer than others to release memory), then that memory can be reassigned to other MCs. If it is not disclaimed within the tuning cycle, it is held over and potentially reassigned to other MCs in the next cycle. Meanwhile, the automation is also dynamically changing the tuning interval (that is, the time between tuning cycles). During periods of turbulent memory profiles (that is, when the demands against the database are in flux), the STMM will increase the time between tuning cycles, to avoid tuning towards transient near-optimal states. Conversely, if the memory profile is stable, it will decrease the time between cycles in order to achieve faster convergence. Tuning intervals for active databases can range between 30 seconds for stable memory profiles and 10 minutes for turbulent profiles. Finally, the STMM also employs a multiple-input, multiple-output process [12] to ensure configuration stability, and to prevent oscillations between several near-optimal states.

The third level of the AH describes the system in terms of memory resources and memory resource

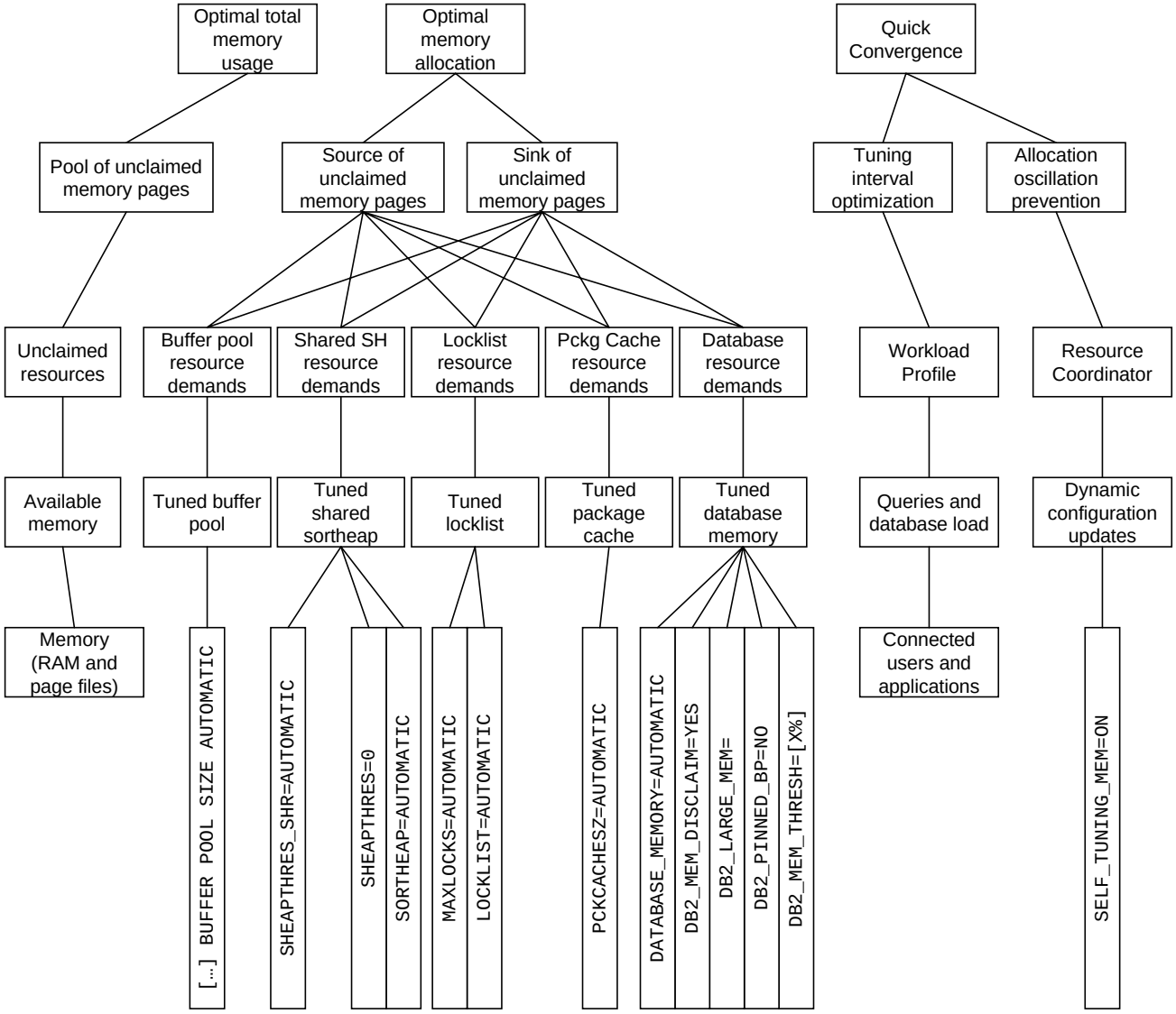

Figure 2: An Abstraction Hierarchy of DB2, scoped to the Self-Tuning Memory Manager

demands. Each tuned MC calculates its own resource demands in terms of a universal metric: Seconds Saved per Page of Memory (SS/PM). Two numbers are calculated for each MC, based on the profile of the current workload against the database: the potential benefit that would occur (in terms of reduced query time) with the addition of a single page of memory, and the penalty that would be paid (in terms of increased query time) should a single page of memory be taken away from the MC. A central resource coordinator compares the numbers as reported by each MC, and revokes memory (resources) from the MC or MCs that would incur the smallest increase in execution time, in order to assign it to the MCs that would benefit the most. The same workload profile that informs the MCs' calculations also serves to determine the time interval before the next tuning cycle (based on its consistency between cycles).

Descending another level of abstraction, the description of the system begins to reflect the resources that constitute the lowest level of abstraction. The description of the system at this level is that of dynamic configuration updates being executed on tuned MCs (including buffer pools, the sortheap, etc. – see Figure 2). The configuration updates are based on the current sizes of the MCs, as well as any extra available memory, and are affected by the nature of the queries and other loads against the database.

Finally, the bottom level of the AH describes this subset of DB2 in terms of the resources necessary for it to function. This includes the configuration parameters that

TABLE I. Information elements identified from the Abstraction Hierarchy

Category of Information	**Information Element**	**Source (AH)**
Purpose	Purpose of the automation: to optimize total memory usage	top level
	Purpose of the automation: to optimize allocation of memory among consumers	top level
	Purpose of the automation: to achieve quick convergence to a near-optimal state	top level
Process	Memory allocation optimization (overall and between consumers) is achieved through de-allocation and reallocation of memory	second level
	Quick convergence is achieved by tuning the timing interval and by preventing configuration oscillation	second level
	Memory is reallocated on the basis of respective resource demands (system seconds saved per page of memory) and extra available resources (including holdovers from the previous tuning cycle)	third level
	The tuning interval is changed based on the workload profile (stable vs. fluctuating)	third level
	The process coordinating resources keeps track of historic configuration changes in order to ensure that the system does not oscillate between near-optimal configurations	third level
	(Relevant properties of tuned components, including maximum and minimum sizes, time it takes to resize, etc.)	fourth level
	The workload is based on queries and other activity against the database	fourth level
	Resources are coordinated (and updates made) through automatic configuration changes to the tuned memory consumers	fourth level
	(Conditions necessary for the STMM to be engaged, including resources and configuration parameters)	fifth level
Performance	Current memory consumption	first level
	Optimality of memory allocation	first level
	Memory currently in transition from one MC to another	second level
	Current tuning interval	second level
	Resource demands for each component	third level
	Reassignments based on resource demands	third level
	Stability of workload profile	third level
	Size of individual tuned MCs	fourth level
	Configuration changes issued by the STMM	fourth level
	Workload running against database	fourth level
	Current available memory (including swap)	fifth level
	Current configuration values	fifth level
	Application(s) connected to the database	fifth level
	State of the STMM process	fifth level

need to be set in order for the STMM to operate, i.e., `SELF_TUNING_MEM` must be set to `ON`, and at least two of the tunable MCs must be configured appropriately (e.g., `SHEAPTHRES_SHR=AUTOMATIC` and `PCKCACHESZ=AUTOMATIC`), so that there exist both an MC from which to reclaim memory, and another to which to allocate it.

III.B. Identification of Information Elements

As we stated earlier, once an AH has been developed for an automated tool, the task of identifying specific information elements is a straightforward one. Based on the AH in Figure 2, and the description given in the previous section, the information elements listed in Table I were identified. The level of the AH from which each information element is derived is listed in the third column. As discussed earlier, all purpose-related information is derived from the top level of the AH; process-related information comes from the remaining four (in our case) levels of the AH; performance-related information corresponds to in-the-moment values for all elements of the AH. (For the sake of brevity, Table I only states or implies current values. However, it should be noted that a log of past values for each information element also presents important performance-based information.)

III.C. Application: Documentation and Logging

Since the STMM has no explicit interface (it is enabled, directly and indirectly, through configuration parameters in DB2), it was necessary to communicate information about this automation to database administrators (DBAs) through other channels. Static information regarding the automation's purpose and processes (algorithms) was placed in the DB2 Information Center (http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp). Dynamic performance-based information was logged to a special STMM log, and the availability of this information was also noted in the Information Center.

Some of the information identified through the AH was already present in the Information Center. Purpose-related information, describing the function of the STMM, and some process-related information – specifically, information from the lowest level of the AH, describing how to turn the STMM on or off – were present. The remaining information was added, chiefly in a new page outlining "operational details and limitations" of the STMM. [13]

In addition to the addition of process-related information to the DB2 documentation, we were also able to effect some changes to the STMM log. Although not all of the performance-related information cited in Table I (such as the optimality of the current configuration) can be measured directly, many elements (e.g., resource demands, newly assigned sizes of MCs) can. Many of these were already being logged, although often using memory addresses and internal indexes. Our primary impact on the STMM log was to add labels to several of the information elements being logged, that would connect them to the language used to describe the automation's purpose and process. The ability to associate performance-related information with information at higher levels of abstraction is expected to help DBAs draw inferences between levels, thereby increasing the impact of this information on appropriate DBA trust in the STMM.

IV. DISCUSSION

While empirical studies measuring the impact of information at three levels of attributional abstraction have not yet been completed, we expect the addition of purpose-, process- and performance-related information to have a positive impact on the appropriateness of operator trust in this new automation. Due to the myriad of external factors affecting trust in (and reliance on) automation, providing operators with the appropriate information is only one part of the solution. Depending on the domain in question, many of the external factors (such as the organizational structure, or the opportunity for operators to engage in exploratory behaviour) are known, or can be designed for (or prescribed). In situations where this is not the case (such as with DB2, a commercial product that is used in a wide variety of social and technological environments), the opportunity for intervention for the explicit purpose of engendering appropriate trust is limited. In these cases especially, it is important to leverage the opportunities that exist.

The focus of our ongoing research is primarily on the information requirements for appropriate initial trust in automation. If an operator decides to engage the automation, that interaction between operator and automation can have a more profound impact on a continually-evolving attitude of trust (or distrust) in the automation than that afforded by this information.

The case study presented in this paper was from the domain of relational databases, and entirely software-based work domain. However, the method and the results should be generalizable to physical, process-control domains (such as nuclear power plants), where appropriate reliance upon (and thus, trust in) automation plays a crucial role in maintaining operator safety. Furthermore, in domains where there exists an explicit interface to monitor and control the automation, Ecological Interface Design can be applied to leverage the results of the AH. This can increase operators' awareness and understanding of the behaviour the automation, thereby engendering a more appropriate degree of trust.

V. CONCLUSION

The Abstraction Hierarchy (AH) is a powerful tool for modeling a complex system. This established form of analysis is central to such design and analysis frameworks as Ecological Interface Design [9] and Cognitive Work Analysis [7]. Using the Abstraction Hierarchy, we have developed a method for identifying specific information elements that satisfy the requirements analysis implied by Lee and See's model of trust in automation. [1] Although the impact of information identified by this method on appropriate operator trust has not yet been tested empirically, we expect that the comprehensive purpose-, process- and performance-related information will engender a more finely-tuned sense of trust in automated tools.

ACKNOWLEDGEMENTS

This research was funded by grants from the IBM Center for Advanced Studies (CAS) and the National Science and Engineering Research Council of Canada.

REFERENCES

[1] J. D. LEE and K. A. SEE, "Trust in Automation: Designing for Appropriate Reliance," *Human Factors*, **46**, 50. (2004).

[2] J. D. LEE and N. MORAY, "Trust, control strategies and allocation of function in human-machine systems," *Ergonomics*, **35**, 1243. (1996).

[3] R. PARASURAMAN, T. B. SHERIDAN and C. D. WICKENS, "A Model for Types and Levels of Human Interaction with Automation," *IEEE Transactions on Systems Man and Cybernetics – Part A: Systems and Humans*, **30**, 286.

[4] B. BARBER, *The Logic and Limits of Trust*, New Brunswick, NJ: Rutgers University Press (1983).

[5] M. DEUTSCH, "Trust and Suspicion," *Journal of Conflict Resolution*, **2**, 265. (1958).

[6] R. C. MAYER, J. H. DAVIS and F. D. SCHOORMAN, "An integrative model of organizational trust," *Academy of Management Review*, **20**, 709. (1995).

[7] K. J. VICENTE, *Cognitive Work Analysis: Towards safe, productive, and healthy computer-based work*, Mahwah, NJ: Erlbaum (1999).

[8] J. RASMUSSEN, *On the structure of knowledge - A morphology of mental models in a man-machine system context*, Roskilde, Denmark: Risø National Laboratory, Electronics Department, Tech. Rep. Riso-M-2192, (1979).

[9] C. M. BURNS and J. R. HAJDUKIEWICZ, *Ecological Interface Design*, New York, NY: CRC Press, (2004).

[10] N. NAIKAR, R. HOPCROFT and A. MOYLAN, *Work Domain Analysis: Theoretical Concepts and Methodology*, Melbourne, Victoria: DSTO Air Operations Division Tehc. Rep. DSTO-TR-1665, (2005).

[11] A. J. STORM, C. GARCIA-ARELLANO, S. S. LIGHTSTONE, Y. DIAO and M. SURENDRA, "Adaptive self-tuning memory in DB2," *Proc. of the 32nd VLDB Conference*, Seoul, Korea, September 12-15, (2006).

[12] Y. DIAO, J. HELLERSTEIN, A. J. STORM, M. SURENDRA, S. S. LIGHTSTONE, S. S. PAREKH, and C. GARCIA-ARELLANO, "Using MIMO Linear Control for Load Balancing in Computing Systems," *Proc. Of the American Control Conference*, Boston, MA, June (2004).

Trademarks