In Proceedings of the 16th IBM Centre for Advanced Studies' Annual International Conference on Computer Science and Software Engineering. (no pagination).

Trust by Design: Information Requirements for Appropriate Trust in Automation

Pierre P. Duez
University of Toronto

Michael J. Zuliani
IBM Canada Ltd.

Greg A. Jamieson
University of Toronto

Abstract

Trust has, since the early stages of IBM's Autonomic Computing (AC) initiative, been recognized as an important factor in the success of new autonomic features. If operators do not trust the new automated tools, they will not use them – no matter how useful or efficient they might be. Despite this stated awareness of trust as a major contributing factor to successful operator adoption of AC functionality (e.g., [11]), no clear process of explicitly designing for operator trust has emerged. The purpose of our research is to develop such a process, to provide a theoretically grounded method for designing for appropriate trust in automation. We define "appropriate trust" as it is described in [6]. By this definition, there are two components to appropriate trust. The first is proper *calibration* of trust, meaning that the operator trusts the automation to the degree of its capability, without over-trust or distrust. The second component is *resolution* of trust: the operator must be sensitive to different or changing conditions (functional or temporal) that might affect the ability of the automation to achieve the operator's goals.

In our research, we have drawn on the extensive review of trust literature by Lee and See [6], who investigated the concept of trust as published from multiple perspectives (e.g., organizational, psychological, and interpersonal). Lee and See have developed a model of trust in automation, based on their review of the literature, which describes the feedback loops that inform one's attitude of trust (or distrust) towards automation. Furthermore, Lee and See identify a continuum of attributional abstraction – information based on which an operator may attribute a sense of trust in an automated tool. Three categories along this continuum are defined: *purpose*-, *process*-, and *performance*-related information are described as being necessary to achieving appropriate trust.

Although they provide these categories of information, Lee and See [6] do not provide a process by which the appropriate information might be identified for a given automated tool. We hypothesized that Work Domain Analysis (WDA; [12]) might serve to provide a clear and definite list. WDA is part of a multi-stage analytic framework, developed for the analysis of complex socio-technical systems. It is a constraint-based, formative analysis, which describes the realm of possible actions, rather than a single prescribed path. The WDA, we reasoned, could be adapted and applied to the problem of design for appropriate trust in automation.

In this paper, we will introduce the model of trust in automation described by [6]. We will also introduce WDA. We will then describe how this analysis can be applied to the question of trust in automation. Finally, we will present a case study from new automation in the IBM® DB2® Version 9.1 for Linux®, UNIX®, and Windows® product (DB2 V9.1), in which we applied WDA to identify specific information requirements for appropriate trust in the Self-Tuning Memory Manager, and used these findings to impact documentation and logging for this new automated functionality.

1 Background/Motivation

With IBM's autonomic computing (AC) initiative, there has been a push to introduce more automation to its middleware software, such as DB2 V9.1, Tivoli® and others. This automation is being introduced to achieve four primary goals – self-configuration, self-optimization, self-protection, and self-healing. The objective of this initiative is to relieve operators of low-level maintenance tasks and allow them to focus on higher-level performance.

Most tasks that are being automated are currently done manually by operators. Although autonomic tools might perform a task more efficiently (and on an ongoing basis, as opposed to discrete efforts by operators), there is always an alternative to engaging these new tools. Operators must *choose* to turn on automation (or at least, not to turn it off). Rather than prescribing the use of automation, we must *convince* operators that it is in their best interests (and in their companies' best interests) to use these new autonomic tools. From a business point of view, the most compelling reason is financial – less operator time spent maintaining systems results in a lower total cost of ownership. However, the possible consequences of over-reliance on automation (that is, use of automation beyond its designed capabilities) might outweigh the potential benefits.

It is important, therefore, to provide sufficient information about the automation to allow operators to *trust* it sufficiently to turn it on, and to be well-enough informed to foresee situations in which the automation might fail (and thereby avoid engaging it under such conditions).

1.1 Autonomic Computing and Trust

Trust is ubiquitous, but very ephemeral. It plays a role in interactions at every level – inter-personal, group dynamics, corporate, etc. – yet it is difficult to isolate and observe on its own.

In the context of IBM's AC initiative, trust has been identified as being an important factor to consider in the design of automation. Operator trust in new automated features is necessary, in order to achieve widespread adoption of new automated and autonomic tools. In describing usability issues relevant to an autonomic system, Telford et al. stated that "for an autonomic solution to be accepted [by operators with different backgrounds and skill sets], user trust must be addressed with at least as much care as the autonomic solution itself" [11, p. 575]. In a discussion regarding security and identity in the context of interacting with autonomic systems, Chess et al. [4] noted that initial adoption of new autonomic features – a necessary precursor to interaction with autonomic systems – depends on operator trust in the new automation: "the human decision makers who opt to use an autonomic system in the first place must have good reason to trust that the system as a whole will serve their purposes" [4, p. 110].

Despite an awareness of trust as an important factor, and one that warrants specific attention at design time, there has been little progress in developing a method for designing explicitly *for* trust in automated tools.

There are, generally speaking, two approaches to designing for trust in automation. The first is to design the automation itself – its control mechanisms and its scope – in such a way as to achieve trust "naturally". For example, there has been discussion of control models such as Policy-Based Administration [2] that would give operators control over the automation in a manner that might engender trust.

It should be pointed out that the main goal of PBA is not explicitly trust, but a more general ease-of-use consideration. Although the control analogy might be more familiar, and allow for greater control over automated behaviour, mistrust in the underlying automated mechanism might still limit operator trust in new automation based on this design approach. PBA is also limited (in the context of designing for trust) by the automated functionality to which it can be applied: some autonomic tools are too low-level, or too atomic in their configurability, to lend themselves well to this control model.

In the broader literature, models such as that proposed in [7] attempt to achieve a similar purpose. They state that through analysis of the work surrounding the automation, one can determine the appropriate degree of autonomy and initiative to be developed into the automation in order to achieve an appropriate degree of reliance.

We feel that the model proposed by Parasuraman et al. does not lend itself well to the problems faced by IBM for reasons similar to those that limit the applicability of PBA: we cannot define an exhaustive set of use cases. In their introduction, [7] cite examples from

anaesthesiology, military (air defence), and securities. Each of these domains provides sufficient context to allow for proper analysis of the tasks that will interact with automation. Furthermore, there is a reasonable expectation of proper training, resulting in a more uniform profile of operators interacting with the automation. Commercial software such as DB2 UDB or Tivoli, on the other hand, is used in an extremely wide variety of software and business environments, and we expect a larger variety of skills and expectations in operators interacting with the automation in these tools.

Our research, therefore, proposes another approach: *given* an automated tool, we are investigating the information that needs to be presented to the operator in order to engender appropriate trust *in that tool*. Much of the theoretical basis for our work comes from research [6] on information requirements for appropriate trust in automation. The focus of our research is thus not on the *design* of the automation, but on the *information* about that tool that is available to the operator. Thus, it can be applied to any automation in a wide variety of contexts. It can be used in conjunction with models pertaining to the design of automation, and is not constrained to any single automation control model.

1.2 Trust in Automation

Lee and See have conducted an extensive review of trust literature, from "organizational, sociological, interpersonal, psychological, and neurological perspectives" [6, p. 50]. They synthesized results from these fields into a coherent model of trust in automation, in the form of an action/perception feedback loop. In this loop, an operator's attitude of trust informs possible reliance action. This attitude is, in turn, informed by beliefs developed through observation of (or feedback from) the automation, as well as external factors (including the reputation of an automated tool – or the company that developed it – and individual predisposition to trust). Lee and See [6] also discussed the information necessary for appropriate trust.

1.2.1 Appropriate Trust

There are two dimensions relevant to appropriate trust [6]. The first is *calibration* of trust, the degree to which an operator's trust in an automated tool's ability corresponds with the capability of that automation to achieve the operator's goal. Poor calibration can result in either over- or under-trust. The former involves relying on an automated tool beyond its designed-for capability, such as relying on a word processor's spell-checking functionality (which may not find typos that are nonetheless valid words). The latter involves turning off, or second-guessing, automation that is designed to achieve (and capable of achieving) a desired goal, such as disabling the DB2 UDB Query Optimizer in circumstances where it would otherwise result in improved performance.

The other dimension of appropriate trust is *resolution* of trust. This refers to the operator's sensitivity to the changing capability of the automation over time and amid varying functions and goals. While an operator's sense of trust might be properly calibrated under normal operating conditions, a change in the system (such as the failure of a component) might have a pronounced effect on the automation's ability to achieve its goals.

1.2.2 Information Requirements: Levels of Attributional Abstraction

In [6], Lee and See identified a continuum of attributional abstraction along which information about the automation lies. They described information at three levels of abstraction – purpose-, process- and performance-related information – that lie along this continuum, based on which operators might attribute a sense of trust in the automation.

Purpose-related information describes the purpose for which the automation was designed. This includes the specific problem that the automation might have been designed to solve, as well as lower-level objectives that the automation was designed to meet. This information provides the operator with an understanding of how well the current demands and goals of her work align with the purpose of the automation. Although using the automation at cross-purposes might sometimes work (e.g., relying on a side-effect of running the automation), an understanding of the divergence of goals ensures that the operator does not rely on artifacts of automation performance that might not be reproducible. Having a proper understanding of the purpose of the automation promotes an appropriately calibrated set of

expectations.

Process-related information pertains to the way in which automation achieves its goals. This includes a description of internal processes and algorithms, as well as information about the resources (including time) required by the automation in order to execute effectively. Just as one might have more faith in a co-worker's work if one understands the process by which he executes his tasks, understanding the processes involved in an automated tool allows an operator to develop a formal understanding, and formal expectations, of the automation's performance. Such an understanding can allow an operator to predict conditions under which the automation might be less reliable in achieving its goals, and encourage appropriate functional resolution of trust.

Finally, performance-related information describes the effect of the automation's actions. This includes the automation's ability to achieve its stated goals, as well as intermediate and peripheral state information. Understanding how the automation affects the system (and how the system can affect the automation) allows the operator to predict automated performance in the future, under similar circumstances and can serve to reinforce formal and informal operator models of automation performance and ability.

Inferences can also be drawn between information at varying levels of attributional abstraction. For instance, information regarding the processes governing an automated tool can serve to add depth to an operator's understanding of the purpose of the automation. Additionally, predictions made about automation's performance, based on knowledge of the internal algorithms and other process-related information, can be verified (or corrected) by performance-related information once the automation has run. This type of feedback can provide the operator with a better understanding of the automation, leading to more finely calibrated trust in this tool.

Although [6] provides a model of trust in automation within which to work, and describes the information requirements necessary for appropriate trust in automation, they do not provide a method by which these requirements might be satisfied. That is, they do not describe any sort of method that would yield specific information for a given automated tool. We have, therefore, turned to Work Domain Analysis [12] as a possible framework with which to determine exact information requirements for new automated tools.

2 Work Domain Analysis

The work domain analysis (WDA; [12]) is a constraint-based, formative analysis of a system. It uses the Abstraction Hierarchy (AH; [9]) to describe a system at different levels of abstraction. Rasmussen [8] identified several meaningful levels of abstraction at which a system might be described; each of these levels can be useful, depending on the use being made of the system. Elements at adjacent levels of abstraction are linked via *functional means-ends* relationships: lower-level representations of system elements describe the means by which a higher-level function (or goal) is implemented. In other words, by descending a level, we answer the question of *how* a function is implemented; by ascending a level, we answer the question of *why* an element is part of the system. By describing the system at these levels of abstraction, and by linking these levels, the AH demonstrates how the higher-level goal (or goals) of the system is achieved using the lower-level resources.

Burns and Hajdukiewicz [3] provide a more detailed description of the AH. They describe examples of, and techniques to create, Abstraction Hierarchies.

2.1 Deriving Information Requirements from a WDA

The AH can be applied to the discovery of specific information requirements by conducting a work domain analysis scoped to the automation. Because our research applies to the determination of information requirements for appropriate trust in (and adoption of) new automated functionality, we are considering automation with which operators are not familiar, and with which there is an explicit initial interaction. Thus, the automation might be perceived by operators as a (sub) system unto itself (nested within the larger context of the software within which it operates), and operators will infer and develop models of the functioning of the automation as an explicit entity.

Having developed an AH of an automated tool, the identification of specific information requirements is straightforward. The highest level of abstraction in the AH, as described above, provides information regarding the *purpose* of the

automation – corresponding to the first category of information. The other levels of abstraction describe how the automation implements its purpose (or purposes). In other words, these describe the *processes* by which it operates. This includes the bottom level of the AH, which – in this analysis – describes the resources necessary for the automation to be engaged, including resources consumed, configuration settings, etc. In a discussion of the levels of abstraction for physical systems, [8] described the lowest level as representing "the physical conditions for purposeful function of a system" [8, p. 10]. Applying this to software systems, one can generalize this to a representation of the resources necessary for purposeful function of a system (in our case, of the automation).

The *performance* of the automation is not only measured in terms of the automation's ability to satisfy its highest-level purpose, but also in terms of the current (and past) values of internal functions and processes. Thus, the current and historical values of the properties of the elements of the AH provide information for the third category of attributional abstraction.

3 Case Study: Self-Tuning Memory Management

In DB2 V9.1, IBM introduced the Self-Tuning Memory Manager (STMM; [10]). With the STMM, memory is tuned in such a way as to eliminate memory bottlenecks when running queries. Memory is allocated among tuned memory consumers (MCs) in such a way as to provide additional pages of memory where they are needed, by de-allocating memory from MCs whose need is not as great. Potential tuned MCs include: buffer pools, the package cache, locking memory (the locklist), sort memory, and total database shared memory.

The STMM does not have an explicit interface; rather, it is engaged through the SELF_TUNING_MEM configuration parameter, and MCs are assigned to automatic tuning by setting their respective sizes to `AUTOMATIC`. The challenges posed by this automation were, therefore, twofold: first, we needed to determine the information requirements to engender appropriate DBA trust in this new automation. Secondly, we needed to determine how best to communicate this information to potential users of the automation.

The following sections outline a partial application of the design method that we have outlined. We conducted a WDA of the STMM in order to develop a list of information requirements to support appropriate trust in automation. We then collaborated with STMM developers to implement the information requirements identified here, in order to engender more appropriate initial trust in this new automated tool.

3.1 Work Domain Analysis of the STMM

The WDA of the STMM – in the form of an AH – requires a more detailed explanation of how the STMM works. The STMM has three stated goals: to achieve near-optimal memory usage, to optimize the allocation of memory among the MCs, and to converge to this near-optimal configuration in an expedient manner.

Tuned MCs keep a running model of the impact (in terms of execution) that the addition or removal of a single additional page of memory would have on their performance. This metric, "system seconds saved per page of memory" (SS/PM) can be compared directly between any two memory consumers. For consumers for which the impact is not symmetrical, for example, where the addition of memory would have a small impact, but the removal of memory would severely slow down the system, two separate figures are calculated. A high SS/PM indicates that the allocation of additional memory to that MC would have a strong positive effect on performance (or, in the case of removal, that shrinking an MC would be strongly detrimental). A low SS/PM indicates that little benefit would occur through the allocation of additional memory to the MC, or that performance would not be severely impacted if memory was taken away. SS/PM can thus be seen as a measure of priority: higher SS/PM indicates that additional memory will have a greater positive impact on performance if assigned to this consumer; low SS/PM indicates that removal of memory would not have a strong negative impact.

At regular intervals, the STMM process polls all tuned MCs for their current SS/PM, and compares them. It determines which MCs are experiencing the greatest need for additional memory, and which can most afford to shrink. Once this decision has been made, the STMM

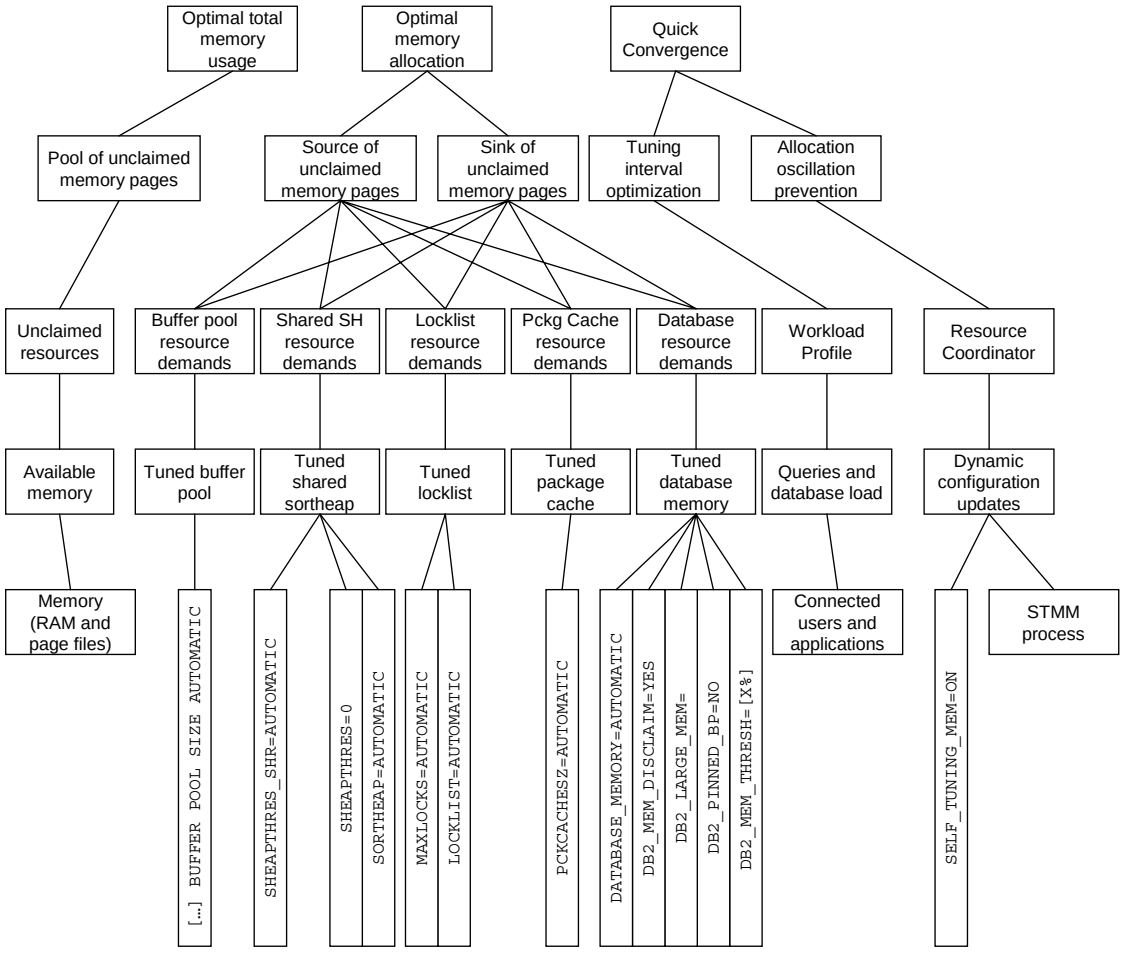

Figure 1: An Abstraction Hierarchy of Self-Tuning Memory Management

first shrinks the MCs from which memory is being de-allocated, and then (once the memory has been freed) reallocates it to the MCs for which there is an urgent need. In cases where this process does not complete in a single tuning cycle, these unallocated pages of memory are tracked, and are assigned to the neediest MCs during the next tuning cycle.

The time interval between tuning cycles is also subject to change. The STMM tracks the workload profile against the database over the course of several tuning cycles. Periodically, the STMM determines whether the workload has consistent memory characteristics – that is, whether the demands placed on MCs from the workload is consistent over time. Depending on this consistency of memory characteristics, the STMM might decrease the time between tuning intervals, in order to achieve more rapid convergence. Conversely, if the memory profile is less constant (or if no workload is being executed against the database), the STMM will increase the time between tuning intervals, in order to avoid tuning towards transient workload profiles.

In addition to optimizing the tuning interval, the STMM also guards against configuration oscillation. It does this initially by tracking which MCs have been grown and which have been shrunk over recent tuning intervals. Once several cycles have passed, it uses a MIMO (multiple-input, multiple-output) process [5] to prevent configuration oscillation.

Figure 1 represents an AH of the STMM. As described earlier, the top level of the AH repre-

sents the goals that the STMM is designed to achieve: near-optimal total memory usage, near-optimal allocation of memory among the MCs, and quick convergence. Although this level does not explain *how* these goals are achieved, it is a complete description of the STMM at a high level of abstraction.

Looking at how the STMM implements these goals (but still maintaining a relatively high level of abstraction in the description), this automation can be described as a system that allocates memory pages. Thus, the first two goals are achieved by properly managing memory pages that become allocated (the source of unallocated pages), either by growing the pool of unallocated memory or by assigning it elsewhere (the sink of unallocated pages). The third goal, quick convergence, is achieved by dynamically tuning the time interval between tunings, and by preventing configuration oscillation. This level of the AH provides a high-level view of the processes involved in the STMM.

Switching in turn to a description of how these high-level processes are implemented, we can describe the STMM as a system wherein a resource coordinator acts based on a workload profile, balancing several resource demands and tracking unclaimed resources at the same time. This level provides sufficient specificity to see where the resource demands are coming from, and gives information regarding how tuning interval optimization and allocation oscillation prevention are effected.

The fourth level describes the system in terms of tuned components. At this level, there is some association with the functions they provide in the greater system. Again, this is a self-contained description of the STMM: this automated tool provides dynamic configuration updates, based on the profile of queries against the database, to the tuned MCs (buffer pools, shared sort heaps, etc.) while at the same time tracking available memory.

The final level describes the resources necessary for the STMM process to function. In a physical system, this would represent the equipment that constitutes the system. In a software system, this has been interpreted as the configuration parameters (to achieve a tuned buffer pool, for instance, one needs to configure that buffer pool's size to `AUTOMATIC`) and the resources (STMM process, memory and an application applying a load to the database) that drive it.

We see, thus, that every level represents a description of what the STMM is, albeit at different levels of abstraction. Each transition down a level explains how the functions are implemented, whereas transitions upward explain how a function or resource is relevant at higher levels (and ultimately, how they are relevant to the goals of the automation).

3.2 Information Requirements for Trust in the STMM

As described above, a WDA was used to determine the information that should be presented to an operator in order to engender appropriate trust in the STMM. The top row of the AH provides *purpose*-related information; the remaining rows represents the *process*-related information. Finally, the current (and historical) values of the functions described represent *performance*-related information. Table 1 summarizes the information content identified by the AH; the level of the AH from which each information element has been derived is shown in the third column.

3.3 Application: Changes to DB2 Documentation and Logging

The model developed in [6] describes the "display" as being the point of feedback that we can affect, to provide the requisite information. This need not, however, refer strictly to a monitor or an interface. The STMM was developed in such a way as not to have (or need) its own interface. Instead, it is enabled by configuring the tuned memory consumers to `AUTOMATIC` size. Since there is no graphicaluser interface specifically for the STMM, other means of providing feedback about the performance of the automation were required. The documentation for the STMM was updated, and changes were made to the logs generated by the STMM, in order to include and clarify relevant information. The changes made to the STMM took place before a CTA was conducted; therefore, they only reflect the results of the WDA.

Through meetings with the STMM information developer and development owner, we were able to add purpose- and process-related information to the documentation. Some of the informa-

tion identified through the AH wasalready present – most notably, the top and bottom levels of abstraction (purpose and resources, respectively). The supplementary information was added to an Infor

mation Center page [1] outlining "operational details and limitations" of the STMM. This page also provided information on determining performance-related information – both through the use of DB2 V9.1 snapshots and by consulting logs.

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

Table 1: Information requirements derived from a Work Domain Analysis of the STMM

The information in, and information format of, the STMM log (stmmlog) was updated as well. Of the performance-related information that can readily be collected (including the size and demands of memory consumers), much was already being logged. The primary modifications to the STMM were the addition of labels for the logged values that could be read by DBAs, and that were expressed in terms that connected back to process- and performance-related information, to facilitate interpolations between these levels.

We were not able to incorporate all of the information elements described above: some performance-related information is difficult for a DBMS to measure internally. Most notably, direct measures are not taken of time-to-convergence and the degree of optimality of memory allocation and usage.

In addition to the specific contributions to documentation and logging, the semi-regular collaboration with the development team had a side benefit of raising the team's awareness of trust in automation issues, and a potential systematic approach for addressing these issues.

4 Discussion

While the empirical studies have not yet been completed, we hope that the addition of purpose-process- and performance-related information to the STMM documentation and logging will have a positive impact on initial DBA trust in this new automated tool. By developing an Abstraction Hierarchy scoped to the context of an automated tool, we have a description of a system that provides information at three levels of attributional abstraction – information that is predicted to engender a more appropriate degree of operator trust in automation. Thus, we have developed a method for eliciting these information requirements that can be generalized to any automated tool (in DB2 or elsewhere).

The focus of our research is on the identification of the information requirements necessary for appropriate initial operator trust in automation. The impact of the form that that information takes (e.g., documentation vs. GUI) is beyond the scope of this research. In future applications, there might be an opportunity to present this information in a richer format, such as an interface providing real-time feedback, possibly allowing interaction with the automation.

However, in this case study, there is no such explicit interface for the automation. There is no graphical user interface for STMM; in modifying the documentation and stmmlog, we have provided the information through the channels available to us. From a practical point of view, this might limit the impact that the information that we have identified will have on initial DBA trust in this new automation.

Ongoing empirical studies are investigating the impact of information identified through this method on initial operator trust in new automated tools. Although our research is only concerned with initial trust, the presentation of purpose-, process- and performance-related information can continue to have a positive impact on ongoing operator trust in automation. If the operator's expectations of the automation correspond to its abilities and performance, then positive reinforcement of those expectations and beliefs can result in a better-calibrated sense of trust in the automated tool.

5 Conclusion

In this paper, we described research in designing for trust in automation. Our focus is on the development of a method for determining information requirements that will engender appropriate operator trust in automation. This approach is complementary to other approaches, such as Policy-Based Administration, that focus on the design of automation itself. Applying the model developed in [6], we propose the application of Work Domain Analysis (WDA) to identify information at three levels of attributional abstraction.

In our case study, a WDA has been applied to the Self-Tuning Memory Manager (STMM). An Abstraction Hierarchy of STMM was developed and used to generate a table of information about the STMM that is expected to engender appropriate trust in this new automation. Working with the STMM information developer and development owner, we were able to effect changes to documentation and logging, adding most of the information identified through the WDA. This interaction has had a side-benefit of raising awareness among developers of the issues involved in designing for trust in automation.

Although we have not yet conducted empirical studies to determine the impact of this

information on operator trust in (and adoption of) the STMM, we expect that the addition of purpose-, process-, and performance-related information will allow operators a better understanding of the STMM, and will improve both the calibration and resolution of their trust in this automation. We intend to continue this research, through empirical studies evaluating the impact of this information on initial operator trust in automation.

Acknowledgments

The authors would like to thank Sam Lightstone, and Adam Storm and the STMM development team for help, feedback and patience through our analysis of STMM.

This research has been funded through an IBM CAS Fellowship.

About the Authors

Pierre P. Duez is a Ph.D. candidate in the Department of Mechanical and Industrial Engineering at the University of Toronto. He is a member of the Cognitive Engineering Laboratory, studying human-automation interaction. He can be reached at duez@mie.utoronto.ca.

Michael Zuliani is a usability specialist at the IBM Toronto Software Lab. While working at IBM, he has studied database administrators and designed user-interfaces for DB2 for Linux, UNIX and Windows. He can be reached by e-mail at zuliani@ca.ibm.com.

Greg A. Jamieson is Assistant Professor of Mechanical and Industrial Engineering at the University of Toronto where he co-directs the Cognitive Engineering Laboratory. He conducts research in the areas of human-automation interaction and human-machine interface design. He can be reached at jamieson@mie.utoronto.ca.

References

[1] Anonymous (2006, Self tuning memory operational details and limitations. *2006(05/26)*, pp. 1.

[2] Anonymous "An Introduction to Policy for Autonomic Computing," vol. 2006, 03/22/2005. 2005.

[3] C. M. Burns. & J. R. Hajdukiewicz, *Ecological Interface Design*. New York, NY: CRC Press, 2004.

[4] D. M. Chess, C. C. Palmer and S. R. White. (2003, Security in an autonomic computing environment. *IBM Systems J.* [[Online]]. *42(1)*, pp. 107-118. Available: https://www.research.ibm.com/journal/sj/421/chess.pdf; https://www.research.ibm.com/journal/sj/421/chess.html

[5] Y. Diao, J. Hellerstein, A. Storm, M. Surendra, S. S. Lightstone, S. S. Parekh, & C. Garcia-Arellano, Using MIMO Linear Control for Load Balancing in Computing Systems. *ACC 2004 - American Control Conference*. June 2004.

[6] J. D. Lee and K. A. See, "Trust in Automation: Designing for Appropriate Reliance," *Human Factors*, vol. 46, pp. 50-80, 2004.

[7] R. Parasuraman, T. B. Sheridan and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst. Man. Cybern. A. Syst. Hum.*, vol. 30, pp. 286-297, May. 2000.

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

[9] J. Rasmussen, "The role of hierarchical knowledge representation in decision making and system management." *IEEE Trans. Sys. , Man, Cybern.*, vol. SMC-15, pp. 234-243, 1985.

[10] A. J. Storm, C. Garcia-Arellano, S. S. Lightstone, Y. Diao and M. Surendra, "Adaptive self-tuning memory in DB2," in *Proceedings of the 32Nd VLDB Conference*, 2006,

[11] R. Telford, R. Horman, S. Lightstone, N. Markov, S. O'Connell and G. Lohman. (2003, Usability and design considerations for an autonomic relational database man-

agement system. *IBM Systems J.* [[Online]]. *42(4)*, pp. 568-581. Available: https://www.research.ibm.com/journal/sj/424/telford.pdf; https://www.research.ibm.com/journal/sj/424/telford.html

[12] K. J. Vicente, *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 1999, pp. 392.

Trademarks