Towards an Understanding of Decision Complexity in IT Configuration

Bin Lin
Department of Electrical Engineering and Computer Science
Northwestern University
Email: binlin@cs.northwestern.edu

Aaron B. Brown
IBM T. J. Watson Research Center
Email: abbrown@us.ibm.com

Joseph L. Hellerstein
IBM T. J. Watson Research Center
Email: hellers@us.ibm.com

***Abstract*— There exist many opportunities for deploying autonomic computing in an IT environment. The highest-value opportunities are going to be where we can reduce human *decision-making complexity* for systems administrators. To identify these opportunities, we need a model of decision complexity for configuring and operating computing systems. This paper extends previous work on models and metrics for IT configuration complexity by adding the concept of decision complexity. As the first step towards a complete model of decision complexity, we describe an extensive user study of decision making in a carefully-mapped analogous domain (route planning), and illustrate how the results of that study suggest an initial model of decision complexity applicable to IT configuration. The model identifies the key factors affecting decision complexity and highlights several interesting results, including the fact that decision complexity has significantly different impacts on user-perceived difficulty than on objective measures like time and error rate. We also describe some of the implications of our decision complexity model for system designers seeking to automate the decision-making and reduce the configuration complexity of their systems.**

I. Introduction

One of the most important benefits of autonomic computing is that it reduces the complexity of managing an IT environment. Visible complexity—of setting configuration knobs, installing and updating software, diagnosing and repairing problems, and so on—is a challenge for IT. It hinders penetration of new technology, drastically increases the cost of IT system operation and administration (which today dwarfs the cost of the IT systems themselves [1]), and makes the systems that we build hard to comprehend, diagnose, and repair. Autonomic computing technology offers the promise of reducing this complexity, but only if applied at the right points in the IT environment. To find these points, we need a way of identifying the high-value automation opportunities, namely the points of highest visible management complexity in the IT environment.

In previous work [2], we argued that complexity can be tackled quantitatively, with a framework that allows system designers to assess the sources of complexity and directly measure the effectiveness of potential complexity improvements. Such a framework provides the key enabler for identifying the high-value autonomic computing deployment opportunities. In previous work, we also introduced an initial approach to quantifying the complexity of IT configuration and management tasks, based on a model of the sources of configuration complexity and a set of metrics derived from that model [3]. This approach focuses on complexity as perceived by expert users–for example, experienced system administrators who have long-term experience with the systems they are managing—and is based on a structural analysis of the configuration or administration task itself, assuming all decisions are known and made correctly.

While this expert-focused approach is proving its value in practical application within IBM, the fact remains that its expert-only perspective limits the complexity insights that it can provide. In particular, the expert-centric focus overlooks some of the highest-value opportunities for reducing complexity via autonomic computing, where autonomic technology can be applied to reduce the decision-making burden on less-experienced system administrators or to simply reduce the need for expert administrators in the first place. If we are to find these opportunities through complexity analysis, we need to understand *decision complexity* and build a decision complexity model that applies to the process of configuring and maintaining computing systems.

However, quantifying decision complexity is not straightforward. Unlike the expert-only case, we cannot simply analyze a "gold standard" procedure for complexity. Instead, we must understand how configuration decisions are made, what factors influence those decisions, and how those factors contribute to both perceived difficulty as well as objectively-measured quantities like time and error rate. And, since our goal is ultimately to be able to easily quantify points of high complexity, we must build and use this understanding pragmatically, without having to resort to complex cognitive or perceptual modeling.

We quickly realized that the only way to make progress towards these goals was to formulate an initial model of decision complexity and move rapidly to collect real data to test that model and provide insight into factors that affect decision complexity. We designed and conducted an extensive user study to produce data relating hypothesized decision complexity factors to measured user perception ratings, task time, and error rate. Because of the difficulties of conducting a controlled study on actual IT tasks with a large population of practicing system administrators, we collected data in an alternative, more accessible domain—route planning—with an experiment carefully designed to connect features of decision-making in the route planning domain with analogous features in the IT configuration domain.

Analysis of our study data reveals several interesting results.

TABLE III

ROUTE PLANNING DOMAIN BASED ON THE MODEL

Factors	**Route planning domain**
Constraints	Traffic
Guidance (Global info)	Map, Expert path
Guidance (Goal-oriented info)	GPS
Guidance (Position info)	Current position indicator
Consequence	Reach the destination or not

We found that task time was primarily affected by the number and type of constraints controlling the key decisions, as well as secondarily by the presence of short-term goal-related guidance. User-perceived difficulty was affected primarily by the short-term goal-related guidance factor, with a secondary effect from the presence of status feedback and only minor effects from constraints. Error rate was affected by short-term goal-related guidance and position guidance. The contrasts in these results suggest the hypothesis that decision complexity has multiple influences, and that system designers can optimize differently to minimize time, error rate, and perceived difficulty, respectively.

We have created a model from our study results that relates decision complexity in the route-planning domain to some of the factors discussed above. Because of the construction of our experiment, we believe that this model should apply to decision complexity in the IT configuration complexity domain as well, and that it can be used to extract some initial guidance for system designers seeking to reduce complexity. However, there is still a clear need for further extension and validation of the model in actual IT contexts. We describe some thoughts and future work on how we intend to accomplish that validation. These are the next steps to continue the exploration of this crucial aspect of complexity analysis and can take us closer to a quantitative framework that can automatically identify points of high complexity and thus target high-value opportunities for deployment of autonomic technology.

II. MODEL AND HYPOTHESIS

To understand decision complexity, we initially approached it with an attempt to build a low-level model that could capture and compute every aspect of a human-driven configuration procedure. We then realized that such a model requires a detailed understanding of human cognitive processes. This approach is too complex for practical use, so we decided to re-approach the problem from a high level, to understand what factors influence decision making, and how those factors contribute to decision complexity.

To address these questions, we formulated an abstract high-level model. As shown in table I, the three major factors we consider in our model are *constraints*, *guidance* and *consequences*. We choose these factors based on results from the HCI literature [4] as well as our own assessment of real IT configuration procedures, where the user is given various types of guidance and needs to make different decisions while facing various constraints. The decisions made by the user then generate different consequences in term of a specific user goal.

Of the guidance, constraints, and consequences factors, guidance is of particular interest because it is the major source of information that users will rely on in making a decision. Analogous to work in the HCI area [4], we further define the formulation of a guidance system in table II. The definition is based on what a good guidance system should provide.

In both tables I and II, we give examples in the IT configuration domain to show the ground on which we build the model. A specific example is the installation of a secured portal site, where a "how-to" guide provides global information guidance about the structure of the entire task; specific dialog boxes in the install wizards for the portal's components provide short-term goal-oriented guidance for configuring each separate component; little explicit position information is provided except what can be gleaned from matching screenshots in the how-to guide with the on-screen display; and confounding information is present in the standalone documentation for each product component of the overall portal stack.

As stated above, our goal in constructing the 3-facet model of guidance, constraints, and consequences is to obtain a high-level understanding of the forces involved in creating decision complexity for IT operational procedures. Thus with the key factors identified, the next step is to validate their impact on decision complexity, and to begin to quantify their relative effects. If we can do this, we can provide a high-level framework for assessing decisions in IT processes and for providing guidance to system designers seeking to reduce decision complexity.

III. APPROACH

To validate our model, ideally we should conduct a user study where users perform a real IT configuration procedure. However we face some obvious difficulties here. First it is challenging to obtain a large set of users with a consistent level of IT experience, especially those with system administration training. Second, it is difficult to finely tune a real IT configuration procedure to validate each component of our model in a controlled, reproducible environment that allows data collection from large numbers of users.

Facing these challenges, we searched for an alternative domain that would allow us to carefully control its elements, and that offered similar characteristics to the IT configuration domain, so that a model built on it could be mapped back to IT configuration domain. We ended up settling on the domain of *route planning*.

In route planning, users navigate a set of interconnected paths to arrive at a prespecified destination within certain limits of time and distance traveled. As they navigate, they make multiple decisions based on information available to them at the time. If they are unfamiliar with the map, the users are effectively non-experts, and thus face decision complexity at each branch point. As shown in table III, the route planning domain contains examples for all factors that we define in our model. In addition, it is familiar to ordinary users with or without an IT administration background, so user training is unnecessary. Using this domain, we can conduct a user study to learn how people make decisions in the context of performing a prescribed procedure, which in our case is navigating a car from one point to another, and extrapolate the results back to the IT configuration domain. While the mapping is clearly

TABLE I

HIGH-LEVEL MODEL OF DECISION MAKING

Factors	Definition	Configuration analogy (examples)
Constraints	Constraining conditions that restrict users to avoid or make certain decisions	compatibility between software products, capabilities of a machine
Guidance	Guiding information on decisions	documentation, previous configuration experience
Consequence	Results from the decision	functionality, performance

TABLE II

SUB-FACTORS WITHIN GUIDANCE

Sub-factors of Guidance	Definition	Configuration analogy (examples)
Global information	Providing an overview of the situation across a set of short-term goals.	A "Redbook" describing the options for combining multiple software products into a solution
Short-term goal-oriented information	Information needed for a particular short-term goal, or goal of current interest is co-located and directly answers the major decision.	A configuration wizard, such as a database tuning wizard
Confounding information	Extraneous or misleading info not related to goals are not presented.	A manual providing application configuration instructions for a different OS platform
Position information	Information identifying relative order of current decision across a set of decisions is provided	Feedback on results of last configuration action; a task-level progress bar

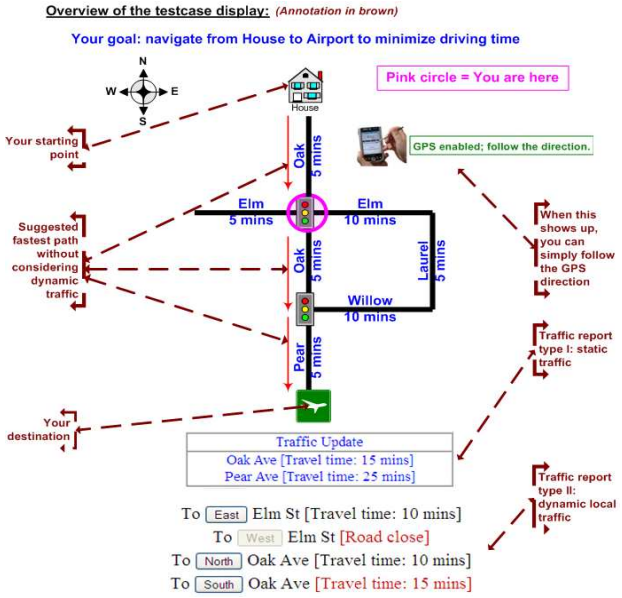

Fig. 1. The screen-shot of a running test case.

not perfect, we believe that it is sufficient to provide a high-level understanding of how our model factors affect decision complexity, giving us an initial step towards the goal.

IV. USER STUDY

We designed an on-line user study that could be taken by participants over the web. The study included multiple experiments with different test cases. Each test case varied the levels of our key factors (guidance, constraints, consequences) and measured the user's time, correctness, and reported difficulty ranking. The detailed design and implementation of our user study can be found in [5]. In each test case, the user is presented with a map consisting of a series of road segments and intersections. Each road segment is marked with a travel time. The pink circle indicates current position of the user in the map. The goal is to navigate a path from the stating point (home) to the airport in the minimum amount of driving time, using the navigation buttons at the bottom of the interface. Figure 1 shows an introductory page, with all possible components annotated. This is what the user saw after logging in and before starting the experiment. Note that not all components showed up in each test case. In the beginning of the experiment, we ask the user about his or her background. At the end of the set of test cases, we ask the user to rank the test cases according to difficulty.

V. SUMMARY OF RESULTS

We discuss both qualitative and quantitative results for our user study in [5]. We summary high-level results below.

Our results suggest the hypothesis that decision complexity has multiple influences on time, error rate, and user-perceived difficulty, and suggests some rough approaches for reducing complexity along these dimensions.

Depending on its goal, optimization for lower complexity will have a different focus. The examples below illustrate possible design approaches for reducing complexity.

- In the IT configuration domain, an installation procedure with easily-located clear info (e.g. wizard-based prompts) for the next step will reduce both task time and user-perceived complexity, though it is unclear how much it will affect error rate.
- A procedure with feedback on the current state of the system and the effect of the previous action (e.g. message windows following a button press) will reduce user-perceived complexity, but is unlikely to improve task time or error rate.
- A procedure that automatically adapts to different software and hardware versions to reduce compatibility constraints will reduce task time, and may also cause a small reduction in perceived complexity.
- Omitting positional feedback (i.e., by not showing users the effects of their actions) may, counterintuitively, increase user accuracy, but at the cost of significantly higher perceived complexity and task time.

VI. NEXT STEPS

A natural next step following this study will be to extend and validate the model in the IT configuration domain through a controlled user study. Again we are facing the challenge of choosing a real scenario, which we can tailor to test various factors of our model. We propose to use a simulated installation process (Figure 2), where the user has a specific installation goal to achieve and has to go through various

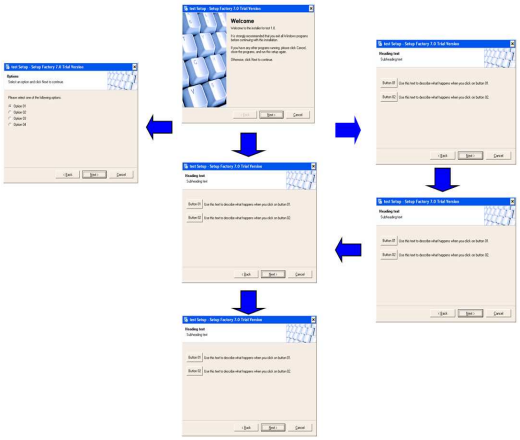

Fig. 2. Mapping

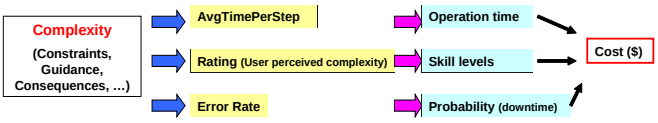

Fig. 3. Steps

decision steps based on provided information (wizard, message windows, buttons...) and choose the right path. For example, the installation process might be to install the web portal software stack mentioned earlier, with the requisite decisions concerning product versions and deployment topology. This approach has the following advantages:

- it is close to a real IT installation process and thus will be familiar to most IT-trained people
- we will have full control over the process
- we can borrow the framework from our route-planning study (on-line experiment engine, test case design etc)

In fact, as described earlier, there exists a mapping between the route-planning domain and the installation domain. For example, the traffic in driving can be seen as analogous to compatibility between software or to machine capacity limits.

Extrapolating from our earlier results, we can hypothesize that the quality of guidance provided—in terms of overall global configuration flow as well as step-by-step goal-directed guidance—will dominate an IT administrator's perception of decision complexity, whereas the degree of compatibility and software configuration sequencing constraints will dominate the decision time in the installation/configuration process. However, as next steps we need to validate this hypothesis with concrete data from follow-on user studies in the IT domain.

After validating and refining the model in the actual IT context, the next step to take it further is to start producing mappings from the model-based measures to higher-level measures that speak directly to aspects of IT administration cost. As figure 3 shows, the idea is to calibrate or map the model measures to higher-level measures such as the time it takes to perform a configuration procedure, the skill level required, and the probability of success at various skill levels. This calibration will almost certainly require the integration of decision complexity with the base complexity measures we developed in previous work [3]. It will additionally require either an extensive user study with trained IT administrators of different skill levels performing real (but controlled) IT tasks, or the collection of a corpus of field data from practicing system administrators performing configuration tasks on production IT environments.

Once we have completed the above calibration to metrics such as time, skill, and error rate for specific configuration procedures, we will then be able to recursively apply our complexity analysis to the collections of IT configuration and administration tasks performed in large IT shops. Here, we will use documented IT management processes to guide the analysis; these may be the aforementioned ITIL best practices [6] or other multi-role IT processes formally-documented in swimlane format, as described in [7]. Ultimately, our hope is to be able to use such processes to guide an evaluation framework, or benchmark, that can analyze each key process activity for complexity and produce a prediction of the cost incurred by the process (in terms of labor cost and downtime cost). While this is a lofty goal that will not be reached overnight, its realization would provide a tremendous asset in helping to quickly target complexity with technology like autonomic computing, and thus to simplify current IT infrastructures and ensure that the new ones we build have the least complexity possible.

VII. Conclusions

This paper takes the first step towards developing a model of decision complexity in the context of configuring computing systems, as a starting point to target optimal deployment of autonomic functionalty. When fully fleshed-out, this model will help identify those points in an IT environment where replacing manual decision-making with autonomic decision-making will have the most value and impact. Our initial model, user study, and analysis shows that decision complexity has significantly different impacts on user-perceived difficulty than on objective measures like time and error rate. Based on our results we have also extracted some basic guidance for reducing complexity. Our next steps are to validate the model in real IT contexts, and extend to future work on mapping measures through the model to higher-level measures. Ultimately, we believe this path will bring us to quantitative tools that will identify focal points for deployment of autonomic technology, and drive the creation of less complex, more easily managed IT infrastructures.

References

[1] J. Humphreys and V. Turner, "On-demand enterprises and utility computing: A current market assessment and outlook," IDC, Tech. Rep. 31513, July 2004.

[2] A. B. Brown and J. L. Hellerstein, "An approach to benchmarking configuration complexity," in *Proceedings of the 11th ACM SIGOPS European Workshop*, September 2004.

[3] A. B. Brown, A. Keller, and J. L. Hellerstein, "A model of configuration complexity and its application to a change management system," in *Proceedings of the Ninth IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*, May 2005.

[4] J. G. H. Christopher D. Wickens, *Engineering Psychology and Human Performance*, 3rd ed. Prentice Hall, September 21 1999.

[5] B. Lin, A. B. Brown, and J. L. Hellerstein, "Towards an understanding of decision complexity in it configuration," IBM T.J. Watson Research Center, Tech. Rep. RC23901, March 2006.

[6] U. Office of Government Commerce, *Best Practice for Service Support*, ser. IT Infrastructure Library Series. Stationery Office, June 2000.

[7] A. B. Brown and J. L. Hellerstein, "Reducing the cost of it operations–is automation always the answer?" in *In Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS 2005)*, June 2005.