RC23901 (W0603-037) March 3, 2006
Computer Science

IBM Research Report

Towards an Understanding of Decision Complexity in IT Configuration

Bin Lin
Department of Electrical Engineering and Computer Science
Northwestern University

Aaron B. Brown, Joseph L. Hellerstein
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Towards an Understanding of Decision Complexity in IT Configuration

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

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

Joseph L. Hellerstein
IBM T. J. Watson Research Center
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*. Decision complexity is the complexity faced by a non-expert system administrator—the person providing IT support in a small-business environment, who is confronted by decisions during the configuration process, and is a measure of how easy or hard it is to identify the appropriate sequence of configuration actions to perform in order to achieve a specified configuration goal. To identify spots of high decision-making complexity, 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.

Keywords

Decision Complexity, Configuration Complexity, Complexity Metrics, Autonomic Computing

1. 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 [7]), 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 [1], 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, which we summarize in Section 2, 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, a key complexity challenge lies in improving the experience of the non-expert system administrator—the person providing IT support in a small-business environment; the administrator who has expertise in one platform but is working for the first time with a new one; the experienced operator trying to deploy a new piece of technology for the first time; the outsourcer trying to apply ITIL best practices [9] but facing decision points within the prescribed processes. In these cases, a different dimension of

complexity becomes paramount: the complexity of figuring out for the first time what steps to follow and what decisions to make while performing a complex configuration process. We call this complexity **decision complexity**.

We believe that the highest-value opportunities for deploying autonomic computing in IT systems are going to be where we can reduce human decision-making. A decision complexity model and analysis of configuring computing systems will help identify those spots. And it can also help determine how difficult it will be to create the automation or the policies that control the automated decisions.

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. 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.

The remainder of this paper is organized as follows. Section 2 briefly summarizes our previous work in complexity modeling for experts. Section 3 discusses the related work. Section 4 describes our initial hypothesized model for decision complexity that we used to construct the user study, which is in turn described in Sections 5 and 6. The results and analysis of our study data are presented in Section 7. Finally, we describe our next steps in Section 8, and conclude in Section 9.

2. COMPLEXITY MODEL FOR EXPERTS

To provide context for our work on decision complexity, we first summarize our previous work on complexity modeling for experts, as described in [1, 3]. Our previous approach focused on creating a standard framework for modeling and quantifying configuration complexity from the point of view of an expert administrator. The intended use of this model and related metrics was twofold: first, to provide an easy way for system designers to obtain quantitative insight into the sources of complexity in their designs (without the need for costly user studies), and second to serve as the foundation for a competitive complexity benchmark.

The approach we followed is based on process analysis. The input to our expert-level complexity model is a codified record of the actual configuration procedure used to accomplish some administrative task on the IT system under test, captured from actual execution or documentation. This record contains information on the configuration contexts present in the procedure, the detailed sequences of actions performed within those contexts, and the data items and data flow between actions, as managed by the system administrator. The model uses features of that record to extract complexity metrics in three dimensions: (1) execution complexity, reflecting the complexity of actually performing the needed action sequences; (2) parameter complexity, reflecting the complexity of supplying the correct values of all needed information to the configuration actions; and (3) memory complexity, reflecting the burden of parameter management and data item tracking carried by the system administrator. Metrics are calculated across the entire procedure to allow cross-procedure comparison, and are also computed at a per-action level, allowing identification of complexity "hot spots" and targeting of future development focus.

The metrics computed by our expert-level model are all objective scores, based solely on the structure of the procedure record. Likewise, the procedure record reflects the optimal configuration path as identified by an experienced expert, with no mis-steps or decision branches. Thus the results of the analysis are objective and comparable across systems and environments, and they reflect inherent structural complexities present in the configuration procedures, but they do not capture any of the decision complexity in identifying

the right procedure or choosing the correct decision branches within that procedure. Hence the focus of this work is on extending the complexity model to include an initial understanding of the impact of decision complexity.

3. RELATED WORK

Understanding decision complexity would appear to be in the purview of human-computer interaction research and psychology. However, the work in those areas [5, 8, 10, 4, 11] has concentrated on understanding how human beings make decisions in general. And the cognitive or perceptual models in those field are very complex and not practical to be directly borrowed to benchmark complexity. In addition, none of those models were developed under the specific goal of understanding how non-expert system administrators make decisions in performing a complex configuration process.

For example, the traditional normative models of decision making prescribe that people assign either an objective or subjective value to an option and then factor in the opinion's probability [5, 11]. It is almost impossible to measure such perceptual value and probability in the real world including IT configuration, not to mention that research has shown a variety of ways in which people deviate from the normative models. For another example, the Prospect Theory [8], which provides a general theory of decision making that explains how people's reasoning deviates from normative models, models people's decisions by a descriptive $\pi(p)$ function, which represents the subjective perception of probabilities [11]. Obviously, it is not very practical to calculate such functions in the real world.

4. 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 1, 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.

For example, one IT procedure we studied involved the installation of a secured web portal software stack, including a portal server, directory server, application middleware server, and a security infrastructure. The procedure contained several decisions concerning software version selection, feature selection (e.g., should the portal support SSL-based secure access), configuration alternatives (e.g., authentication mechanisms), and sequencing.

In this procedure, guidance was provided in the form of product manuals, a step-by-step "how-to"-style guide [6], and on-screen prompts. The procedure involved several constraints, such as incompatibilities between different versions of the constituent software products, different feature sets across different software versions, and resource consumption requirements. Each of the several decision points in the process (for example, choosing which security protocol to use) resulted in consequences relative to the original goal—either performance or functionality implications in the resulting portal installation, or the ability to achieve the goal state at all. An example of the latter style of consequences is a case where certain product versions could not be co-located on the same machine. If the decision was made to co-locate the incompatible versions, the procedure resulted in a non-working system.

Of the guidance, constraints, and consequences factors, guidance is of particular interest because it is the major source of information that user will consult with in making a decision. Analogous to work in the HCI area [4], we further define the formulation of a guidance system in table 2. The definition is based on what a good guidance system should provide.

In both tables 1 and 2, we give examples in the IT configuration domain to show the ground on which we build the model. For example, in our portal case study, the "how-to" guide provided global information guidance about the structure of the entire task; specific dialog boxes in the install wizards for the portal's components provided short-term goal-oriented guidance for configuring each separate component. There was little explicit position information except what could be gleaned from matching screenshots in the how-to guide with the on-screen display. Confounding information was 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.

5. 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 configura-

Table 1: 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 2: 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 than the one being used
Position information	Information for 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

tion 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 3, 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 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.

6. USER STUDY DESIGN

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.

6.1 Experiment and test cases

We designed 3 experiments for our user study. Each user was assigned an experiment randomly after he logged in. Each experiment consists of 6 sequential test cases and 1 warm-up test case in the beginning. We have 10 possible test cases (not including the warm-up) in total, which we carefully designed and believe will help us find out the answers to the questions that we discussed in previous section 4.

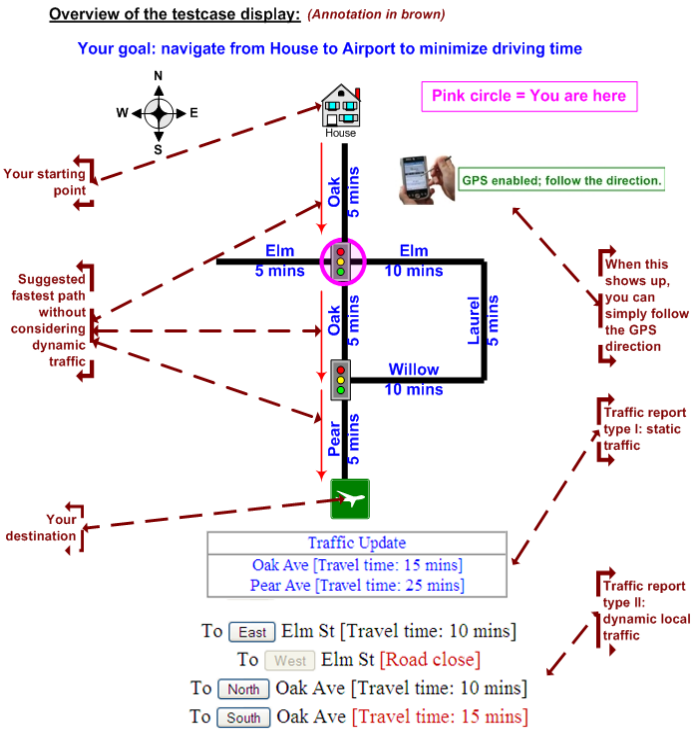

Figure 1: The screen-shot of a running test case.

Table 3: 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

Table 4 summarizes the test cases we used in the study. We also carefully selected the set of test cases to be included in each experiment so that we can maximize our data set. The major parameters we built into our test cases are:

- **Traffic:** we have two types of traffic update, representing constraints in our complexity model. **Static update** presents the global traffic updates to the user in the beginning of the test case, while the **dynamic update** only discloses the local traffic to the user when he arrives at the traffic-related intersection or road. This is the equivalent of listening to a traffic report versus running into a traffic jam, and in the IT domain is analogous to prespecified versus unexpected constraints (such as version compatibility). For dynamic update, we further design two types of update: **road close** and **travel time update**. The former is analogous to the constraints in the IT configuration domain that eliminate the viability of one installation path, and cause user to undo and look for a new path, while the latter is an analogy to those constraints that only change the resulting performance of an otherwise-viable configuration.

- **Expert path:** an expert path is the suggested route for user without considering the traffic. It is analogous to the previous experience a user or expert brings to configure the same system, or the information presented in a "how-to" or step-by-step walkthrough guide.

- **GPS:** similar to the advanced Global Position System people use when driving in the real world, it is analogous to an omniscient expert that directs people during a configuration procedure, which we believe requires the least mental effort from the user in making decisions.

- **Position indicator:** a pink circle on the map indicates current location of the user. It is analogous to the position information defined in Table 2, i.e. the feedback information in IT context, which provides feedback on the current state of the system and the effect of the previous action.

- **Path differences:** different length of routes from the starting point to the destination reflects different consequences resulted from user's decisions. To study the impact of consequences on the decision complexity, we vary the path difference for different maps so that some maps have small path differences among all possible routes, while some maps have big path differences.

6.2 Perspective of the user

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. Each test case uses a slightly different map to avoid learning effects; however, all maps are topographically equivalent with the same set of decision points. The optimal path differs across test cases, but note that only one path is optimal in each map. This scenario is roughly equivalent to the IT configuration problem of being given a new system to install/configure and a set of documentation laying out possible system- and resource-dependent sequences of configuration actions. Just as the user has to work out the optimal path through the map, the IT administrator has to make the configuration decisions at each branch point in the IT setup process, based on the state of the system and the visible paths ahead.

To maximize the quality of our data, we requested users not to multi-task or walk away from the system while a test case was in progress. In some test cases, users may have encountered traffic or other obstructions that changed the travel time for certain road segments or rendered them impassable. Users may also have received different levels of guidance that may have helped them to identify the right path. 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.

- What is your gender? (Male / Female)
- Do you have formal or informal training in mathematics, computer science and/or engineering? (Yes / No)
- How long have you been driving? (specify years)
- How often do you drive a car? (Every day / A few times a week / A few times a month / Rarely / Never–do not drive)
- Do you use online map services like Mapquest, Yahoo Maps, Google Maps, etc when you need to drive to an unfamiliar destination? (Always / Frequently / Occasionally / Never)
- How would you rate your proficiency with map-reading and navigating based on maps? (Excellent / Very good / Good / Mediocre / Poor)

Table 4: Summary of test cases; a × means the parameter is not presented while a check means the opposite.

No	Pos indicator	Traffic type	Update type	Path diff	Expert path	GPS
1	√	×	×		×	×
2	√	static	travel time		×	×
3	√	dynamic	road close		×	×
4	√	dynamic	travel time		×	×
5	√	×	×		√	×
6	√	dynamic	travel time		√	×
7	√	dynamic	travel time		×	√
8	√	dynamic	road close	bigger	×	×
9	√	dynamic	travel time	bigger	×	×
10	×	×	×		×	×

At the end of the set of test cases, we ask the user to rank the test cases according to difficulty on a scale of 1 (easiest) to 6 (most difficult). Note that as the user proceeds through the experiment, he has the opportunity to input a reminder at the end of each test case to help him remember which one is which when he gets to the end of the experiment.

6.3 Implementation

We implemented our on-line user study using a JAVA Servlet-based architecture with server-side collection of data, including timings. The web pages are dynamically generated based on the data submitted by the user. The experiment server records user navigation sections (i.e. decision points) as well as the real time he takes to complete each test case. The server also compares the user's path with the optimal path for each map.

We used XML-based experiment configuration files so that we can not only design various test cases and experiments using a standard data format, but also finely control each parameter of the study by simply modifying the corresponding XML file.

We used JPEG images to represent the steps in the experiment. In the beginning of the experiment and after each navigation action, a JPEG image was presented to the user. In our experiments, these were images of the route map with the appropriate information presented to the user (such as their current position, or the suggested expert-supplied path). The implementation consists of approximately 3100 lines of JAVA and 211 JPEG files.

One of our goals in implementing the user study is to design a general framework so that it can be easily exploited for similar experiments. The core of our JAVA Servlet is a general user-driven decision engine which can present information, react and record all according to external XML-based configurations. By supplying different sets of JPEG images, along with corresponding XML files, our experiment framework should be adaptable to explore many other aspects of IT administration and complexity. For example, the map images could be replaced by screen-shots of actual configuration dialogs (with corresponding XML files). We discuss this possibility later in Section 8 as a possible next step in validating our results in a more directly-IT-relevant context.

6.4 Two-stage User Study

Our user study consisted of two stages. In the first stage, 37 users from IBM T.J Watson Lab participated. In the second stage, we revised the order of test cases in each experiment based on the analysis of the user data from the first stage. Note that we did not change the content of the test cases. 23 users from IBM Almaden Lab, University of California, Berkeley, and Harvard University participated.

In both stages, we advertised for participants via emails. The duration of the study for each user was around 30 minutes. The 10 participants who did the best at the experiments were automatically entered into a random drawing; two won a $50 gift certificate each.

7. RESULTS AND ANALYSIS

7.1 Metrics

We use three metrics to evaluate the study.

The *AvgTimePerStep* is the average time that users spent in one step in one test case. Note that we have different number of steps in different test cases. The *UserRating* is the average rank specified by users. Recall that users were asked to rank test cases from 1 to 6 in term of difficulty, where 6 indicates the most complex/difficult test case and 1 indicates the easiest one. If they felt that two (or more) test cases have approximately the same level of difficulty, they may give them the same rank. The *ErrorRate* is the percent of users who failed to find the optimal path for a test case. Note that for each test case, we have only one optimal path.

7.2 Qualitative results

To reduce the variation across users, for each user we normalized his AvgTimePerStep based on test case 7 (see Table 4), where we provided GPS turn-by-turn guidance. This test case involves no decision making at all on the user's part, and thus reflects each user's baseline speed of navigating through the user interface of the study; in all cases each user spent the least amount of time in test case 7,

Figure 2 shows that most parts of the trends for UserRating and normalized AvgTimePerStep are tracked, except for test case 8, which users felt was difficult but in which they only spent a small amount of time. In figure 3, we see similar tracking between ErrorRate and normalized AvgTimePerStep, except that in test case 10, where all users who did that test case spent more time due to the lack of the position indicator. Interestingly all users were able to find the

Table 5: Summary of complexity factors

Model factors	Constraints	Guidance (global)	Guidance (goal)	Guidance (position)	Consequences
Test case factors	Test case no	Test case order	Found optimal or not		
Background factors	Gender	CS & math background	Driving frequency	Online-map usage	Proficiency with map & reading routing

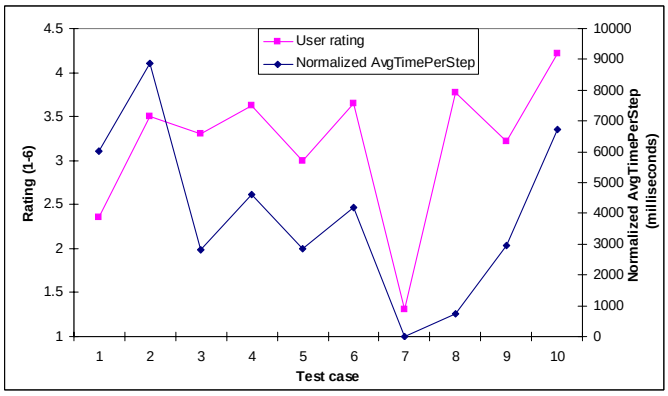

Figure 2: User rating and time; Avg Std for time over all test cases: 4368 milliseconds

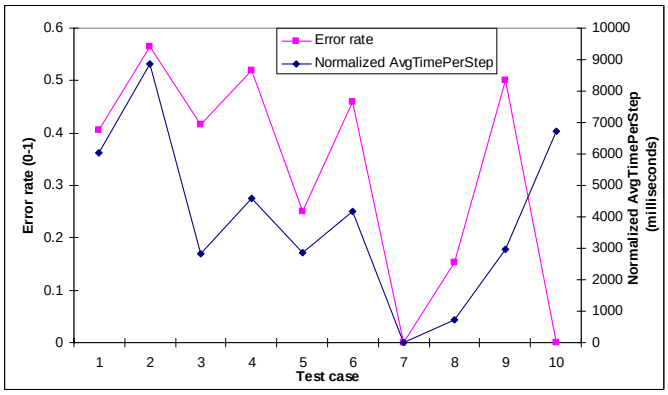

Figure 3: Error rate and time; Avg Std for time over all test cases: 4368 milliseconds

optimal path in this test case. One possible reason for this is that when there was no position indicator, users had to become more careful in each step and spent more time in tracking their movement and planning their routes. As a result, the ErrorRate was greatly reduced.

Overall, this result confirms that decision complexity has different impacts on:

- User-perceived difficulty
- Objective measures (time and error rate)

Figure 2 and 3 bring out some interesting discussion. However we can not draw quantitative conclusions from them because the variation of AvgTimePerStep for each test case is very large across all users. The average standard deviation of AvgTimePerStep over all test cases is 4368 milliseconds, almost half of the maximum AvgTimePerStep.

In an attempt to gain more insight into the data, we further analyzed the data in two steps.

- Step I: general statistical analysis; treat each test case measured as an independent data point, with the goal to identify factors that explain the most variance.
- Step II: pair-wise per-user test case comparisons; get more insight into specific effects of factor values, with the goal to remove inter-user variance.

We discuss the results of this analysis in the next section.

7.3 Quantitative results

Table 5 lists all factors that we identified within the study. The first row lists all factors that we propose in our initial model. We call them *complexity model factors*. The second row includes those test case related factors. The third row shows all background related factors.

7.3.1 Time

Table 6: Baseline analysis of variability for time

Factor	Sum Squares
Test case #	32.778
Driving years	17.637
Online-map usage	7.260
Residual	45.192

Table 7: Analysis of complexity factors for time

Factor	Sum Squares
Constraints	16.764
Guidance (goal)	11.397
Consequence	1.939

As step I, we conduct an ANOVA (Analysis of Variance) test on AvgTimePerStep using a linear-space regression model. To see how much variance that we can explain, we first include test case number, test case order, and all background related factors (e.g. gender, driving years) in the ANOVA. Since test case number subsumes all of the factors we explicitly altered during the experiment, we believe that the variance that can be explained by test case number should be a superset of what can be explained by our model factors. Table 6 is the summary of the ANOVA. We only list those factors which have significant impact on *Sum of Squares*. As we can see, the maximum variability that can be explained by model factors (those we explicitly varied in the experiment) is 32.778. Interestingly, the length of driving years contributes 17.637 to the Sum of Squares, indicating experience is a significant factor. Other factors are not listed due to their tiny impact. The residual, we believe, comes from random per-user effects that can't be explained by either model factors or user background.

Based on this baseline analysis, we then do an ANOVA test on our model factors (i.e. constraints, levels of guidance, consequence) to identify those factors that explain the most variance. We know from our earlier analysis that at most 32.778 of the sum of squares variance can be explained by these factors. Table 7 indicates that constraints and short-term goal related guidance have the most impact on time, followed by a small amount of affect from consequences. Other factors have very little impact and are not listed. Note that constraints and guidance together explain 96% of the total variance explainable by model factors.

From this Step I data, we can conclude that the user's decision time is primarily influenced by the presence of constraints, along with goal-directed guidance such as step-by-step instructions. The impact of visible consequences is also present, though at a lower level. The regression fit data confirms this analysis, showing increased predicted step time when constraints are present, and decreased time when goal-directed guidance is provided or consequences are more visible.

Next, in step II, we aim to remove inter-user variance and get more insight into specific effects of factor values. Table 8 summarizes our pair-test analysis, providing 95% confidence intervals. In these tests, we compared the results of a pair of test cases from a single user, to determine a per-user effect of factor differences between the test cases. We then averaged across users to test for a significant cross-population effect. Note that we only list those results which allow us to discount the null hypothesis, that two test cases have no difference, with $> 95\%$ confidence. This result confirms what we found in step I, i.e. constraints and guidance (goal) are two major factors influencing task time. We further discover that statically-presented constraints (like our static traffic) actually increase time compared to dynamic constraints, likely due to the user's need to assess the relevance of the global information at each step of the procedure.

7.3.2 Rating

Similar to our analysis for time, we first do an ANOVA test on UserRating using test case number, test case order, and all background related factors. From table 9, we can see that the maximum variability that can be explained by the model factors is 51.671. The length of driving years again has some impact although the impact is small compared to that in the time case.

We then feed the model factors into the ANOVA test. Different from what we found in the time case, here short-term goal related guidance is now the top 1 influential factor, followed by position guidance. Constraints however only have small impact on the user's rating.

The results also show that a third factor, Log(order), impacts UserRating, although at a much lower level than Guidance. The Order factor refers to the sequence in which the user was shown the various test cases; the presence of the Log(order) term in the ANOVA implies that there is a bias to users' rating, with higher ratings given later in the sequence.

Table 11 is the summary for step II - pair-wise test. Although it does not statistically show the impact of guidance (goal), it confirms the impact of position guidance and constraints providing 95% confidence intervals.

Table 9: Baseline analysis of variability for rating

Factor	Sum Squares
Test case #	51.671
Driving years	7.125
Residual	67.087

Table 10: Analysis of complexity factors for rating

Factor	Sum Squares
Guidance (goal)	42.272
Guidance (position)	6.278
Log(order)	2.071
Constraints	1.683

7.3.3 Error rate

The analysis of ErrorRate is different from time and rating because we only have one data point per test case, i.e. error rate averaged across all users who finished that test case. So it is hard do any further statistical analysis. However from figure 3, we can still draw two conclusions. First, all users were able to find the optimal path in test case 7, where we provided GPS turn-by-turn guidance. So we can conclude that providing short-term goal related guidance will reduce error rate. Second, the error rate in test case 10 is also zero, where the position guidance was not provided. So the conclusion is that error rate will be reduced when guidance (position) is not present, although perceived difficulty and time both increase, illustrating the tradeoffs between different forms of decision complexity.

7.4 Summary

The contrasts and complexity in the above 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.

Table 8: Pair-wise test for time

	1st Study	95% CT	2nd Study	95% CT
Constraints	static traffic > dynamic (road close) static traffic > without traffic	(0.78, 1.07) (0.73, 1.01)	static traffic > dynamic (road close) static > dynamic (travel time update)	(1, 1) (0.54, 1.13)
Guidance (goal)	without expert path > with expert path	(0.53, 0.89)		

Table 11: Pair-wise test for rating

	1st Study	95% CT	2nd Study	95% CT
Guidance (pos)	without pos indicator > with pos indicator	(0.51, 1.05)	without pos indicator > with pos indicator	(0.51, 1.05)
Constraints	static traffic > dynamic traffic (road close)	(0.54, 1.13)		

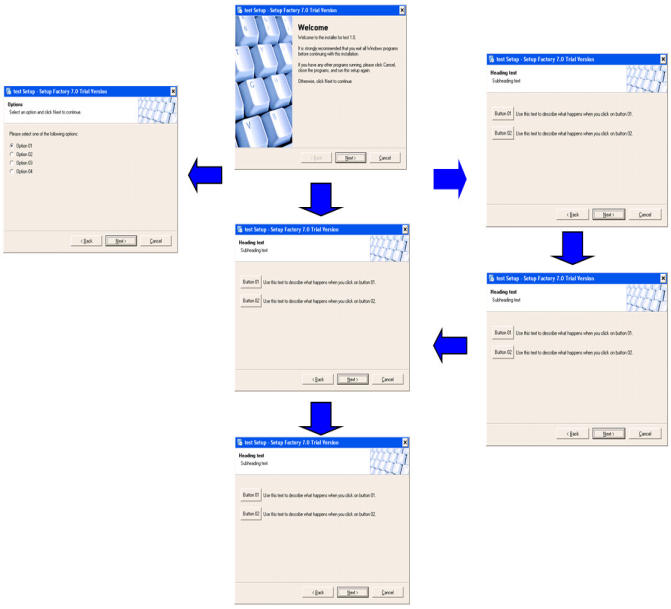

Figure 4: Mapping

- 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.

8. 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 4), where the user has a specific installation goal to achieve and has to go through various 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. The global map is analogous to an installation/configuration manual or to a flowchart of the overall process. Likewise, the driving time per road segment can be mapped to the number of features achieved per installation step.

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.

Making use of our current general framework as discussed in section 6.3, we can expect that conducting these next user studies would be straight-forwarded in terms of implementation.

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 5 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.

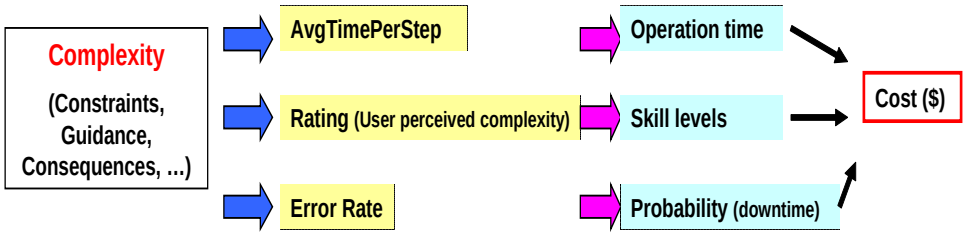

Figure 5: Steps

Here, we will use documented IT management processes to guide the analysis; these may be the aforementioned ITIL best practices [9] or other multi-role IT processes formally-documented in swimlane format, as described in [2]. 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.

9. CONCLUSIONS

Motivated by the need for a quantitative framework for targeting deployment of AC technology, this paper takes the first step towards developing a model of decision complexity in the context of configuring computing systems. 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 model includes three factors: constraints, levels of guidance and consequences. Based on the model, we conduct a carefully controlled user study in an analogous route-planning domain. We discuss both qualitative and quantitative results. We reveal the important fact that decision complexity has significantly different impacts on user-perceived difficulty than on objective measures like time and error rate. And we identify the key factors affecting decision complexity, which we use to extract some basic guidance for reducing complexity. We also propose our next step on validating the model in real IT contexts. And we describe our future work on mapping measures through the model to higher-level measures, which we believe will ultimately 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.

10. ACKNOWLEDGEMENTS

We thank John C. Thomas, Christopher Ward, Melissa J. Buco and Heiko Ludwig for very useful suggestions. We are also grateful to the participants in our study.

11. REFERENCES

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

[2] Brown, A. B., and Hellerstein, J. L. 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).

[3] Brown, A. B., Keller, A., and Hellerstein, J. L. 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] Christopher D. Wickens, J. G. H. *Engineering Psychology and Human Performance*, 3rd ed. Prentice Hall, September 21 1999.

[5] David E Bell, Howard Raiffa, A. T., Ed. *Decision Making: Descriptive, Normative and Prescriptive Interactions*. Cambridge University Press, October 19 1988.

[6] Galic, M., Halliday, A., Hatzikyriacos, A., Munaro, M., Parepalli, S., and Yang, D. *A Secure Portal Using WebSphere Portal V5 and Tivoli Access Manager V4.1*. Redbooks. IBM, December 2003.

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

[8] Kahneman, D., and Tversky, A. Prospect theory: An analysis of decision under risk. *Econometrica 47*, 2 (1979), 263–91. available at http://ideas.repec.org/a/ecm/emetrp/v47y1979i2p263-91.html.

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

[10] Plous, S. *The Psychology of Judgment and Decision Making*, 1st ed. McGraw-Hill, January 1 1993.

[11] Robert J. Sternberg, T. B.-Z. *Complex Cognition: The Psychology of Human Thought*. Oxford University Press, Feb 2001.