Why and *Why Not* Explanations Improve the Intelligibility of Context-Aware Intelligent Systems

Brian Y. Lim, Anind K. Dey
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
{byl, anind}@cs.cmu.edu

Daniel Avrahami
Intel Research Seattle
1100 NE 45th St., Seattle WA 98105, USA
daniel.avrahami@intel.com

ABSTRACT
Context-aware intelligent systems employ implicit inputs, and make decisions based on complex rules and machine learning models that are rarely clear to users. Such lack of system intelligibility can lead to loss of user trust, satisfaction and acceptance of these systems. However, automatically providing explanations about a system's decision process can help mitigate this problem. In this paper we present results from a controlled study with over 200 participants in which the effectiveness of different types of explanations was examined. Participants were shown examples of a system's operation along with various automatically generated explanations, and then tested on their understanding of the system. We show, for example, that explanations describing why the system *behaved* a certain way resulted in better understanding and stronger feelings of trust. Explanations describing why the system *did not behave* a certain way, resulted in lower understanding yet adequate performance. We discuss implications for the use of our findings in real-world context-aware applications.

Author Keywords
Intelligibility, context-aware, explanations

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Over the past 20 years many attempts have been made to achieve Weiser's vision of ubiquitous computing [26] through continued advancements in context-aware computing [23]. Context-aware intelligent systems adapt and tailor their behavior in response to the user's current situation (or *context*), such as the user's activity, location, and environmental conditions. Most such systems employ complex rules or machine learning models. With the goal of calm computing [27], these systems rely on implicit input often collected without user involvement. Thus users of context-aware applications can have great difficulty reasoning about system behavior [6,7]. Such lack of system intelligibility (in particular if a mismatch between user expectation and system behavior occurs) can lead users to mistrust the system, misuse it, or abandon it altogether [19].

One mechanism to alleviate this lack of intelligibility in intelligent context-aware systems is through automatically generated explanations. This approach has been shown to be effective in other domains such as decision making [12] and recommender systems [15] where providing explanations led to increased trust and acceptance. Commercial applications, such as Amazon's product recommender system or Pandora's music recommender now integrate explanations into their interfaces.

The use of explanations has been studied extensively in the area of Knowledge-Based Systems (for a review, see [14]). Unlike in most context-aware applications, however, knowledge-based systems typically focus on supporting *expert* users trying to gain *expert knowledge* of a domain. In contrast, our aim is to explore how providing explanations to novice end-users can help improve their understanding of novel context-aware systems. We expect that this improvement in understanding would result in improved user trust of the system, and lead to ready and rapid acceptance and adoption of these nascent technologies. Cheverst *et al*. explored the development and initial user experience of a decision-tree, rule-based, personalizable office control application that was transparent and provided explanations to users. They learned many issues relevant to their specific application, but did not investigate the effects specific to explanations or their impact on users, and had only a handful of participants.

In this paper we describe a detailed investigation of a number of mechanisms for improving system intelligibility performed using a controlled lab study. To investigate these intelligibility factors and their effects, we defined a model-based system representing a canonical *intelligent system* underlying a context-aware application, and an interface with which users could learn how the application works. We recruited 211 online participants to interact with our system, where each one received a different type of explanation of the system behavior. Our findings show that explaining *why* a system behaved a certain way, and explaining why a system did *not* behave in a different way

CHI 2009, April 4–9, 2009, Boston, MA, USA.

provided most benefit in terms of objective understanding, and feelings of trust and understanding compared to other intelligibility types.

The paper is organized as follows: We first define a suite of intelligibility explanations derived from questions users may ask of a context-aware system and that can be automatically generated. We then describe an online lab study setup we developed to compare the effectiveness of these intelligibility types in a quick and scalable manner. Next we describe the experimental setup used to expose participants to our system with different types of intelligibility and the metrics we used to measure understanding, and users' perception of trust, and understanding. We present two experiments in which we investigated these factors, elaborating on the results and implications. We end with a discussion of all of our results and plans for future work.

INTELLIGIBILITY

Much work has been conducted on the generation and provision of explanations, particularly in the domain of knowledge-based systems (KBS) [14] and intelligent tutoring systems (ITS) [2]. Gregor and Benbasat [14] present a review of explanations, identifying several constructs often used to generate explanations. In KBS, content type can be classified into a number of categories: *reasoning trace* (providing the line of reasoning per case), *justification* (attaching "deep" domain knowledge), *strategic* (system's problem solving strategy), and *terminology* (term definitions). In our work, we are particularly interested in the types of explanations that will be most helpful to end-users of context-aware applications.

Context-aware systems can form user confusion in a number of ways. For example, such systems may not have familiar interfaces, and users may not understand or know what the system is doing or did. Furthermore, given that such systems are often based on a complex set of rules or machine learning models, users may not understand why the system acted the way it did. Similarly, a user may not understand why the system did not behave in a certain way if this alternative behavior was expected. Thus, our focus in the work presented here is on explanations that can be regarded as reasoning traces.

While a reasoning trace typically addresses the question of *why* and *how* the application did something, there are several other questions that end-users of novel systems may ask. We chose to look into the following questions (adapted from [11]):

1. **What:** What did the system do?
2. **Why:** Why did the system do W?
3. **Why Not:** Why did the system not do X?
4. **What If:** What would the system do if Y happens?
5. **How To:** How can I get the system to do Z, given the current context?

Throughout this paper we will refer to these as our 5 intelligibility questions, and the explanations addressing each of them as intelligibility type explanations.

Norman described two gulfs separating users' goals and information about system state [21]. Explanations that answer questions 1 to 3 address the gulf of evaluation (the separation between the perceived functionality of the system and the user's intentions and expectations), while explanations answering questions 4 and 5 address the gulf of execution (the separation between what can be done with the system and the user's perception of that). With a partial conception of how a system works, users may want to know what would happen if there were some changes to the current inputs or conditions (question 4). Similarly, given certain conditions or contexts, users may want to know what would have to change to achieve a desired outcome (question 5).

Some research has looked into providing explanations for these questions. The Whyline [16] and Crystal application framework 20] provide explanations that answer Why and Why Not intelligibility questions for novice programmers and desktop users respectively, but the effectiveness of these explanation types were not compared. In KBS, several expert systems (*e.g.*, MYCIN [9]) provided reasoning trace explanations to support Why, Why Not and How To questions, but the comparative benefits of these were not compared. Instead, follow-up research in KBS has focused on providing justification explanations for the reasoning trace, and explaining problem-solving strategy used. Since context-aware systems are not primarily about inferring decisions from deep knowledge bases, we keep our focus on reasoning trace explanations.

This paper deals with providing and comparing the value of explanations that address four of these intelligibility questions to investigate which of these explanations benefit users more. We label these intelligibility types: Why, Why Not, What If, and How To. Since the system we developed to evaluate the value of explanations, already explicitly shows the inputs and output of the system (see next Section on Intelligibility Testing Infrastructure), Question 1 of the five intelligibility questions (*What* the system did) could not be investigated in this study.

Hypotheses

We hypothesize that different types of explanations would result in changes in users' user experience: *understanding* of the system and *perceptions* of trust and understanding of the system. We will now present our hypotheses about each of these intelligibility questions.

Why explanations will support users in tracing the causes of system behavior and should lead to a better understanding of this behavior. So, we expect:

H1: *Why* explanations will improve user experience over having no explanations (*None*).

Why Not explanations should have similar benefits to *Why* explanations; however, users' ability to apply *Why Not*

explanations may not be as straightforward. There may be multiple reasons why a certain outcome did not happen; while a why explanation may be a single reasoning trace (or at least a small number of possible traces), a why not explanation is likely to contain multiple traces. Given this complexity, users would require more cognitive effort to understand how to apply the knowledge, and may do so poorly. As such, we expect:

H2: *Why Not* explanations will **(a)** improve user experience over having no explanations (*None*), but **(b)** will not perform as well as *Why* explanations.

Explanations for How To and What If questions would have to be interactive and dynamic, as they depend on example scenarios that users define themselves. Receiving these explanations should be better than receiving none at all. However, given that novice end-users are unlikely to be familiar with a novel system, they may choose poor examples to learn from, and learn less effectively than the Why explanations. So we expect:

H3: *How To* or *What If* explanations will **(a)** improve user experience over having no explanations (*None*), but **(b)** will not perform as well as *Why* explanations.

To test these hypotheses, we created a test-bed that allows simulating different types of intelligent systems and testing different explanation types. We describe this testing infrastructure next.

INTELLIGIBILITY TESTING INFRASTRUCTURE

We developed a generalizable web interface that can be applied to various application domains to study the effect of the various mechanisms for providing intelligibility. Users interact with a schematic, functional *intelligible* system that could underlie a context-aware application: it accepts a set of inputs (*e.g.* Temperature, Humidity), and uses a model (for example, a decision-tree), to produce a single output (*e.g.*, Rain Likely, or Rain Unlikely). Users are shown different instances of inputs and outputs and can be given various forms of explanations (or no explanations) depending on what intelligibility type is being studied. To users who do not receive explanations, the system appears as a black box (only inputs and the output are visible).

This infrastructure allows us to efficiently and rapidly investigate different intelligibility factors in a controlled fashion and closely measure their effects; further, the online nature of the infrastructure allowed us to collect data from over two hundred participants. The design also has the advantage of being generalizable to a variety of different domains simply by relabeling its inputs and outputs to represent scenarios for those domains.

System Implementation

The web interface was developed using the Google Web Toolkit [13]. We leverage Amazon's Mechanical Turk infrastructure [1] to recruit and manage participants and manage study payments by embedding our study interface in the Mechanical Turk task interface. Users found our study through the listings of *Human Intelligence Tasks*

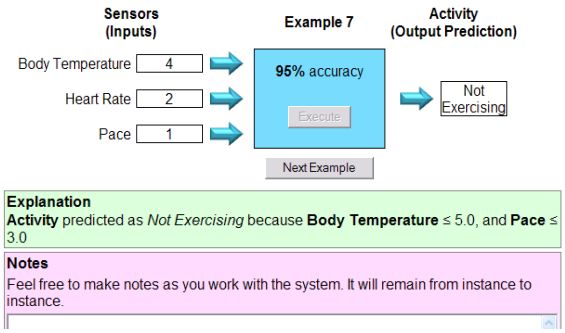

Figure 1. Screenshot of the interface for our intelligibility testing infrastructure.

(HITs), and after accepting our HIT, they participated in the study and interacted with the system.

The user encounters several examples of system inputs and output (see Figure 1). He first sees the input values listed and has to click the "Execute" button so the system 'generates' the output. When he is done studying the example, he clicks the "Next Example" button to move on. Depending on the explanation condition the user is in, he may receive an explanation about the shown example (our system also supports explanations only being shown upon user request, or On Demand).

We modeled our testing infrastructure on typical sensor-based context-aware systems that make decisions based on the input values of multiple sensors. Many of these sensors produce numeric values and the applications change their behaviors based on threshold values of the sensors. For example, a physical activity recognition system could look at heart rate and walking pace. To keep our experiments and the task reasonably simple for participants we restricted the system to three input sensors that produce numeric values, we used inequality-based rules to define the output value, and constrained the output to belonging to one of two classes. In Experiment 1, for example, we defined two inequality rules that consider two inputs at a time (see Figure 2). Since we did not want the lack of domain knowledge (*e.g.*, that the body temperature can rise from 36.8 to 38.3°C when weight lifting) to affect users' understanding of the system, so the inputs use an arbitrary scale of integer values: Body Temperature from 1 to 10, and Heart Rate and Pace from 1 to 5.

As machine learning algorithms are popular in context-aware applications, our system also uses machine learning. Among the myriad of machine learning algorithms, decision trees and Naïve Bayes lend themselves to be more explainable and transparent, while others are black-box algorithms that are not readily interpretable (*e.g.*, Support Vector Machines and Neural Networks) [22]. We chose to

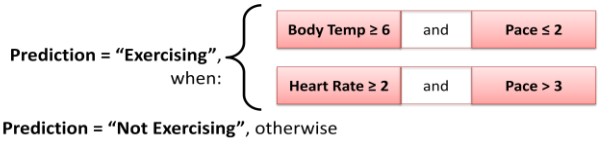

Figure 2. Inequality-based rules for physical activity domain.

start our investigation using the more simple decision trees with inequality rules because they are popular among context-aware systems (e.g. [5, 8]), and are easier to explain, especially to end-users who may not understand the probabilistic concepts that underlie Naïve Bayes algorithms. Using an implementation of the C4.5 Decision-Tree algorithm [27], our system learns the inequality rules from the complete dataset of inputs (250 instances from the permutations of all inputs) and outputs and models a decision tree (see Figure 3) that is used to determine the output value.

Decision Tree Explanations

While the decision tree is able to classify the output value given input values, we had to extend it to expose how the model is able to derive its output. The decision tree model lends itself nicely to providing explanations to the four intelligibility type questions. Table 1 describes how the explanations were implemented.

METHOD: EXPERIMENTS

Given the different factors we wanted to investigate and the flexibility of our testing infrastructure, we were able to independently test different intelligibility elements in a series of experiments. We made the tradeoff of conducting controlled, yet simple, experiments with a large number of subjects that we could generalize from, over studying more realistic, yet more complex situations. We ran Experiment 1 to explore providing different types of intelligibility explanations (Why, Why Not, and the control condition with no explanations). The system was presented in the context of the domain of activity recognition of exercising as described above. However, due to participants' prior knowledge of the domain, our results were difficult to interpret. So, we decided to subsequently run experiments with an abstract domain. Experiment 2 compares explanations provided to address each of the four intelligibility type questions (Why, Why Not, How To, and What If) individually to investigate which are more effective in helping users gain an understanding of how our intelligent system works compared to not having explanations (None).

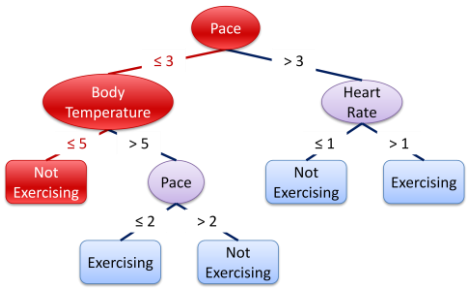

Figure 3. Visualization of the learned decision tree model used in Experiment 2.

Study Procedure

Our study consists of four sections. The first section (Learning) allows participants to interact with and learn how the system works. Two subsequent sections test the participants' understanding of the system (Fill-in-the-Blanks Test and Reasoning Test), and a final section (Survey) that asks users to explain how the system works (to evaluate the degree to which participants have learned about the system's logic) and to report their perceptions of the explanations and system in terms of understandability, trust and usefulness.

Learning Section

In the Learning section, participants are shown 24 examples with inputs and output values (see Figure 1). These examples were chosen from all possible input instances, to have an even distributed over all branches in the decision tree, and they appear in the same order to all participants. Examples were arranged in ascending order of Body Temperature, then of Heart Rate, then of Pace. Participants have to spend at least 8 seconds per example (controlled by disabling the Next Example button). Explanations are provided depending on the experimental condition. If participants receive explanations, they will receive them *automatically* when executing each example. It is important to note that explanations are only provided during the

Why: Walk the decision tree to trace a path of decision boundaries and values that match the instance being looked at. Return a list of inequalities that satisfies the decision trace of the instance (*e.g.*, "Output classified as Not Exercising, because Body Temperature≤5 and Pace ≤3"; see Figure 2).
Why Not: Walk the whole tree initially to store in memory all the traces that can be made. Walk the tree to find the why-trace, and find differing boundary conditions on all other traces that return the alternative output. A why-not trace would contain the boundary conditions that match the why trace and boundary conditions where it is different (*e.g.*, "Output *not* classified as Exercising, because Pace≤3, but *not* Body Temperature>5"). A full Why Not explanation would return the differences for each trace that produces the alternative output. However, so as not to overwhelm the user, we use a heuristic to return the differences of just one why-not trace, the one with the fewest differences from the why trace. Note that while this technique is suitable for small trees, it is not scalable to large trees, and heuristics should be used to look at subsets of traces.
How To: Take user specified output value, and values of any inputs that were specified. Iterate through all traces of the tree to find traces that end with the specified output value and has branches that satisfy the specified input values. If any trace is found, it identifies the satisfying boundary conditions for the unspecified inputs and returns them. Note that if there is a trace, there will only be one, since an instance can only satisfy one trace in the tree. If there are no boundary conditions for the unspecified inputs, then these inputs can take any value. If no trace is found, then there are no values for the unspecified inputs, given values of the specified inputs, to produce the desired output value.
What If: Take user's inputs and puts it through the model to classify the output. Return the output value, but since this is a simulation, do not take any action based on this output value.

Table 1. Algorithms for generating different types of intelligibility explanations from a decision tree model.

Learning section. Participants are told that their task is to learn how the system works and are encouraged to take notes using a dedicated text box that persists throughout the Learning section. At the end of the Learning section, users are told to spend some time studying their notes as those are not available during the rest of the study.

Fill-in-the-Blanks Test Section

This section tests users on their ability to accurately specify a valid set of inputs or output; they are given a single blank in one of the inputs or the output, and are given the rest of the inputs/output. There are 15 test cases, three with blank *Body Temperature*, three with blank *Heart Rate*, four with blank input *Pace*, and 5 with blank output. These test cases different from the earlier examples, and are randomly ordered, but in the same order for all participants. On seeing each test case, users have to fill in the missing input or output with a value that makes the test case correct. If an input is missing, they should provide a value that causes the given output value to be produced; if the output is missing, they provide a value that would be produced with the given input values. After providing the missing value, they are also asked to provide a reason for their response. Participants are not given any explanations during this test and, are not given the answer or told whether they are correct after they finish.

Reasoning Test Section

This section shows users three complete examples, and, for each example, asked to give reasons why the output was generated, and why the alternative output was not. These test case examples are different from what users have encountered before, and are randomly ordered, but are in the same order for all participants. To see if improved understanding can lead to improved trust, users are also asked how much they trust that the output of the system is correct for each example. Participants are not given any explanations during the test and, are not given the answer after they finish.

Survey Section

The final Survey section is used to collect self-report information from users. Users provide a more detailed description of how they think the system works overall (*i.e.*, an elicitation of their mental models), and are asked several Likert-scale questions to obtain an understanding of how users feel about using our system, including whether they trusted and understood the system and explanations.

Measures

In order to see what types of intelligibility explanations would help users better understand the system, and whether this improved understanding would lead to better task performance, improved perception of the system, and improved trust in the system output, a number of measures were collected.

Task performance was measured in terms of task completion time, and the Fill-in-the-Blanks Test inputs and output answer correctness. Task completion time was measured with two metrics: total learning time in the Learning section, and average time to complete each Fill-in-the-Blanks Test question.

Guess/Unintelligible	No reason given, guessed, or reason incoherent
Some Logic	Some math/logic rules, probability, or citing past experience
Inequality	Correct Type of rules which are inequalities of inputs with fixed numbers
Partially Correct	Some, but not all, of the correct rules, or extra ones
Fully correct	All correct rules, with no extra unnecessary ones

Table 2. Grading rubric for coding free-form reasons given by participants. Mental Models were coded using this same rubric.

User understanding is measured by the correctness and detail of the reasons participants provide when they give their answers (in the Fill-in-the-Blanks Test), explain examples (in the Reasoning Test), or give an overall description of how the system works (mental model in the survey). The reasons given for each answer in the Fill-in-the-Blanks Test were coded using a rubric (see Table 2) to determine how much the participant understands about how the system works. Reasons are coded as Guess/Unintelligible if participants wrote they were guessing, did not write anything, or wrote something not interpretable. Reasons are graded as Some Logic if participants provided some rules or probability statement or cited past experience (*e.g.*, saying they saw something similar before) that were not inequalities with fixed numeric boundaries. This includes cases such as "Body Temperature>Heart Rate". Reasons are coded as Inequality if participants specified an inequality of at least one of the inputs with a fixed numeric boundary (*e.g.*, Body Temperature>7). Reasons are coded as Partially Correct if participants provided only one rule with the correct input, boundary value, and relation. Reasons are coded as Fully Correct if participants get only all the sufficient rules correct, and did not list any extra ones. Each reason was coded with only a single grade (*i.e.*, the highest appropriate grade).

There are two inequality rules (*e.g.*, Pace>3, and Heart Rate>1) for each test case or example, so answer reasons for the Fill-in-the-Blanks Test have two components. We measure how many of these components participants learn using three coding metrics that count (i) the number of inputs the participant mentions as relevant in the reasons, (ii) the number of correct rules described, and (iii) the number of extraneous rules mentioned (0 or 1).

The reasons for the why and why not questions that participants provided in the Reasoning Test were coded using a rubric similar to Table 1. We also recorded, on a five-point Likert-scale the participant's level of *trust* of the correctness of the outputs for each example in the Reasoning Test.

In the survey, we asked participants to describe their overall *understanding* of how the system works. This mental model understanding is coded in a similar manner to why reasons, but not applied to specific examples.

We did a factor analysis on the 16 Likert-scale questions of system and explanation *perceptions* in the survey and derived 6 factors. 10 questions on the system are grouped into 3 factors: Understood System, Found System Confusing, Liked System/Found it useful. 6 questions on explanations are grouped into 3 factors: Explanations Difficult, Explanations Useful, Understood Explanations.

EXPERIMENT 1

Our first experiment focused on providing answers to hypotheses H1, and H2; whether Why explanations would lead to improved user understanding, trust, perception, and performance more than having no explanations, and H2 regarding providing Why Not explanations being better than no explanations, but not as good as Why explanations. We chose the domain of activity recognition of exercise, which users would have a reasonable understanding of. Mapping to the generalized abstract system described earlier, the system takes on the role of a wearable device that can measure the wearer's Body Temperature, Heart Rate, and walking/running Pace, and classify whether the wearer is exercising or not (Figure 2). The first rule can be satisfied during strength training (*e.g.*, weight lifting) that does not require much walking about, but can raise body temperature, while the second rule can be satisfied by running.

Participants in the no explanation (None) condition did not receive any explanations, and could only execute each example and move on. Participants in the Why condition receive Why explanations automatically along with the output value when they execute each example by clicking the "Execute" button. Participants in the Why Not condition receive a Why Not explanation in place of a Why explanation.

Participants

53 participants were recruited. There were 18 participants in the None condition, 18 in the Why condition, and 17 in the Why Not condition. 39 of the participants completed the demographics survey showing 20 were female and 19 males with ages ranging from 18 to 57 (M=29.8), and education levels ranging from less than high school to post-graduate degrees. We removed from the analysis any responses of participants who took fewer than 15 minutes (one participant in the None condition) or longer than 50 minutes to complete the four sections. This was done to filter out participants who just click through the steps without thinking, and to leave out participants who may be distracted while performing the task and take too long. On average, participants took 34 minutes to complete the study. Participants were each given $3 for completing the study ($1 base and a $2 bonus to motivate performance). A further $2 was offered to a few participants who participated in interviews conducted soon (up to a few days) after completing the task.

Results

To analyze participants' ability to apply their understanding, the number of correct answers per participant was summed and a Tukey HSD pair-wise test was performed. The number of correct answers was the dependent measure. The analysis showed significant differences in accuracy between intelligibility types ($F[2,84]=8.85$, $p<.001$; see Figure 4). To analyze participants' ability to formalize their understanding, their reasons were coded using the rubric in Table 1 and dummy variables were generated indicating: Inequality or better (0 or 1), Partially or Fully Correct (0 or 1), and Fully Correct (0 or 1). The analyses were done with the reason coding as the dependent measure and with condition as a fixed effect. Participants were modeled as a random effect and nested within condition. A Tukey HSD pair-wise test of the occurrences of each coded score shows that providing explanations leads to more correct answers than not providing any (contrast of None with Why and Why Not: $F[1,50]=15.1$, $p<.001$). However, there was no significant difference in the number of correct answers between Why and Why Not type explanations.

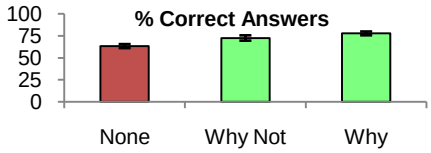

Figure 4. Participants receiving explanations (in the Learning section) answered significantly more questions correctly in the Fill-in-the-Blanks section.

Figure 5. Percent of reasons coded as *Inequality*, *Partially Correct*, or *Fully Correct* in the Reasoning Test section.

Using the grading rubric in Table 1 on the Why reasons provided in the Fill-in-the-Blanks Test, we found that participants in the Why and Why Not conditions were able to produce more Partially Correct reasons compared to those in the None condition ($F[1,50]=27.4$, $p<.001$) (see Figure 5). Participants in the Why condition produced more Fully Correct reasons compared to None and Why Not ($F[1,50]=10.8$, $p<.002$). There were no significant differences between Why and Why Not. A similar pattern was found in the Reasoning Test section Participants in the Why condition had a higher level of trust than those in None ($F[1,49]=8.98$, $p<.005$), while those in the Why Not condition did not. The survey measures on overall mental model or perceptions of the system and explanations did not reveal significant differences.

Discussion and Implications

The generally poor trust in the system could be due to occasional examples that follow the system rules, but may not be 'natural' (*e.g.*, high Body Temperature and low pace predicted as "Not Exercising"). The answer and reason results indicated that providing explanations lead to better understanding and trust of the system with less

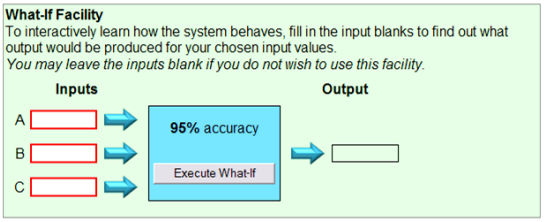

Figure 6. Participants in the What If condition view this facility. They enter values for the inputs A, B, and C, and the system simulates what the output would be.

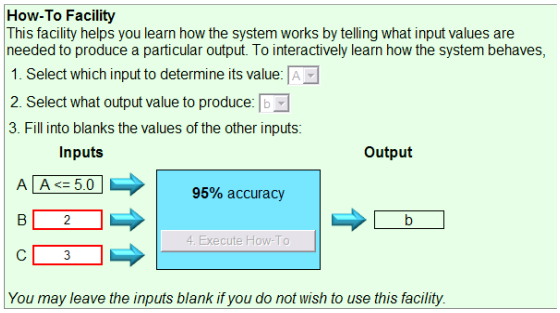

Figure 7. Participants in the How To condition view this facility. They specify two of the input values and an output value, and the system responds with the possible values of the remaining input.

disagreement about the system output. However, in their provided why reasons, several participants alluded to the domain of physical activity and physiology to explain how the inputs (Body Temperature, Heart Rate, and Pace) should relate to whether the device wearer was "Exercising" (*e.g., "moving & high [body temperature], looks like running so I upped the [heart rate]"*). Furthermore, most responses specified the inputs as "high" or "low" rather than specifying numeric boundaries (*e.g., "heart rate is low, so must be a high pace along with high body temperature to predict exercising"*). This suggests that having prior knowledge would lessen participants' effort to be precise about their understanding. To mitigate the effects of prior knowledge, and to support more generalizability to other domains, we decided to anonymize the inputs and outputs with an abstract system.

EXPERIMENT 2: INTELLIGIBILITY TYPE

Our second experiment focused on comparing the effectiveness of different explanations types for each of the 4 intelligibility questions. Using the algorithms described in Table 1, we isolate these explanations for each condition.

Method

This experiment followed the procedure of Experiment 1. For the None, Why, and Why Not conditions, participants see the same interface as in Experiment 1, but with the inputs obfuscated as *A*, *B*, and *C*, and the output values relabeled to *a* and *b*.

Participants in the What If condition receive a What If interaction facility (see Figure 6) instead of an explanation to let them see the output given their choice of inputs. Participants in the How To condition received an interactive facility (see Figure 7) to determine how to get the system to produce a chosen output value. To control for the number of examples encountered, participants in the What If and How To conditions only get 12 complete examples (the even-numbered examples of other conditions), and can invoke their respective intelligibility facilities 12 times to see a total of 24 examples (similar to the other conditions). For each condition, the explanations or explanation facilities will always appear as each example is executed.

Participants

158 participants were recruited. There were 26-37 participants in each of the 5 conditions: None (n=31); Why (n=30); Why Not (n=31); How To (n=29); What If (n=37). 115 participants completed the demographics survey. 65 were female, ages ranged from 18 to 72 (M=31.9), and education levels ranged from less than high school to post-graduate degrees. On average, participants took 33 minutes to complete the study (similar to Experiment 1, they were required to complete the study within 15 to 50 minutes). Compensation was identical to Experiment 1.

Results

We analyzed the results by using the Tukey HSD pairwise test, looking for differences between groups for our previously described metrics. Compared to participants in the None, What If and How To conditions, participants in the Why and Why Not conditions had *more correct* answers in the Fill-in-the-Blanks tests (see Figure 8), provided *better reasons* (see Figures 9 and 10), and reported having a *better understanding* of the system (see Figure 11a). Participants in the Why and Why Not conditions had an accuracy of 80.0% and 74.2%, respectively, compared to 61.7% for the None condition ($F[1,152]=51.6$, $p<.001$; see Figure 8). More of their answer reasons were coded as at least Inequality type rules (Inequality: $F[1,153]=198$, $p<.001$), Partially Correct ($F[1,153]=195$, $p<.001$) and Fully Correct ($F[1,153]=108$, $p<.001$) (see Figure 9). Finally, the self-reports of understanding for Why and Why Not were 3.14 and 2.79, respectively (see Figure 11a).

Participants in the Why condition further distinguished themselves from Why Not by giving more Fully Correct reasons (contrast of Why with Why Not: $F[1,153]=23.2$, $p<.001$), and trusting the system output more (contrast of Why with None: $F[1,153]=8.26$, $p<.001$ vs. contrast of Why Not with None: p=n.s.) with means of 3.26, 3.0 and 2.46 for Why, Why Not and None, respectively (see Figure 11b). However, these participants also took the longest to answer each Fill-in-the-Blanks test case (M=26.3 seconds, compared to M=22.0 and M=17.0 for Why Not and None, respectively) (contrast of Why with None: $F[1,145]=9.32$, $p<.003$ vs. contrast of Why Not with None: p=n.s.).

Surprisingly, participants in the Why Not condition were *not* statistically better at providing Why Not reasons than Why reasons. While participants in the What If condition were indistinguishable from those in the None condition across all of our metrics, we did find that participants in the How To condition were able to understand the types of

rules used in the system better than participants in the None condition (answer reasons coded as Inequality or better: F[1,153]=15.6, p<.001).

Surprisingly, participants in the Why Not condition were *not* statistically better at providing Why Not reasons than Why reasons. While participants in the What If condition were indistinguishable from those in the None condition across all of our metrics, we did find that participants in the How To condition were able to understand the types of rules used in the system better than participants in the None condition (answer reasons coded as Inequality or better: F[1,153]=15.6, p<.001).

To identify why participants in the Why Not condition understood less about the rules than Why, we coded the quality of answer reasons on the number of inputs and rules mentioned. Participants in the Why condition provided more correct rules (M=1.19 *vs.* M=0.79; F[1,59]=6.16, p<.02) and fewer extraneous rules (M=0.11 *vs.* M=0.23; F[1,59]=8.276, p<.006) than Why Not.

Discussion and Implications

The results in Experiment 2 validate those in Experiment 1 with a more generalized abstract domain, while not suffering from confounds due to prior domain knowledge. The Why and Why Not explanations improved participants' understanding, increased their trust in the system, and their task performance. Examining the user reasons, we found that automatically generated Why explanations allowed users to more precisely understand how the system functions for individual instances compared to Why Not explanations. This is in spite of the Why Not explanations being logically equivalent to Why explanations since flipping the *not*'s in the former can derive the latter. Moreover, we found that the Why Not participants tended to provide fewer correct rules (more participants could only provide one correct rule instead of two) for the answer reason, or provide extraneous inputs and rules that the system did not consider for the respective test cases, as compared to the Why participants. These indicate that Why Not participants tended to learn only part of the reasoning trace, and did not associate the two rules together, but treated them separately. This failure in rule conjunction could be due to the inclusion of negative wording (*i.e.* "but" and "not") in the Why Not explanation. The mental effort to understand the Why Not explanation and create such a rule conjunction is certainly more than those in the Why condition had to expend, which could explain the differences we observed.

Neither the How To nor What If explanations showed much benefit over not having explanations. Participants receiving What If explanations did not optimize their selection of examples, with some users even selecting input values out of range (*e.g.*, A=100). Given the abstract and mathematical nature of the experimental setup, without any reasoning trace (unlike Why, Why Not, How To), almost none of these participants proposed inequality rules as reasons, similar to those in the None condition. However, as with the

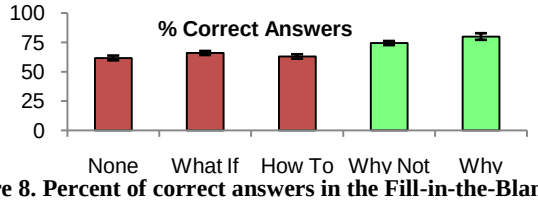

Figure 8. Percent of correct answers in the Fill-in-the-Blanks test section, by condition. Different colors (left three vs. right two) indicate statistically significant differences.

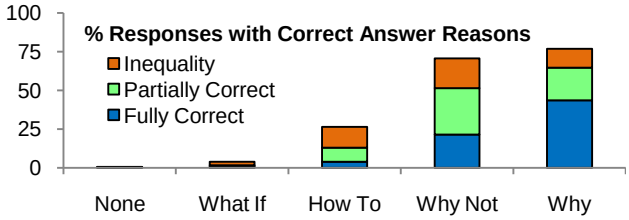

Figure 9. Percent of reasons coded as *Inequality*, *Partially Correct*, or *Fully Correct* in the Fill-in-the-Blanks Test section for each condition.

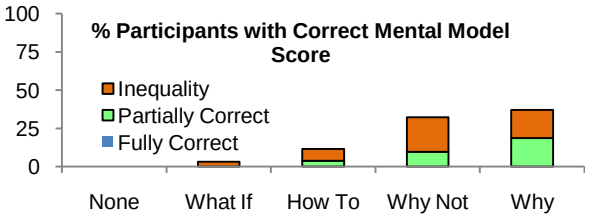

Figure 10. Overall understanding of the system was similar to the understanding in-situ of individual examples, but responses were less precise (fewer correct descriptions).

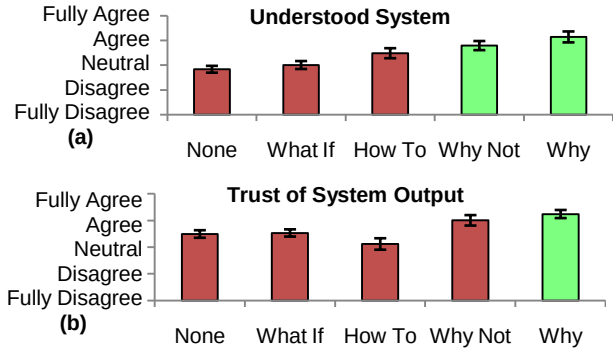

Figure 11. Self-reports of (a) understanding and (b) trust, by condition. Different colors indicate significant differences.

effect of domain knowledge (in Experiment 1), participants who did not receive reasoning traces did consider the inequality rules, but just not correctly (see Figure 9).

Our results suggest that developers should provide Why explanations as the primary form of explanation and Why Not as a secondary form, if provided. Our results may suggest the ineffectiveness of How To and What If explanations, but these intelligibility types may be more useful for other types of tasks, particularly those relating to figuring out how to *execute* certain system functionality, rather than interpreting or *evaluating*.

GENERAL DISCUSSION

We now discuss the findings of our two experiments and their implications for real world context-aware systems.

Impact of Prior Knowledge

We found in Experiment 1 that participants formed less accurate and precise mental models of the system, compared to those in Experiment 2. This could be due to participants applying their prior knowledge of exercising to understanding how the system works and not paying careful attention to the explanations, as evidenced by the reasons they provided. This persistence of mental model was also shown in [24] where participants received explanations, over time, of how an interruptibility system worked. As many real context-aware applications are based on common everyday activities, users may have strong prior knowledge of the domains although weak understanding of the applications, and may also not diligently learn from the provided explanations. One way to address this could be to learn from the knowledge-based systems community, and provide deeper *justification* [14] explanations to help users understand why the system behavior may be different from typical everyday understanding. To support users that may not have this prior knowledge, perhaps explanations that provide this knowledge can be used.

From the Lab to the Real World

Our intelligibility experiments form a necessary first step in exploring this rich area, but they differ from real applications in that users would have different goals when asking either of the intelligibility type questions. In reality, users would ask Why questions when they lack an understanding of how the application works, but Why Not questions when they expect certain results that the application did not produce. This distinction in user expectations and goals was not present in our lab study. Thus, even if Why Not explanations are found to be less effective than Why explanations for real systems, users may prefer Why Not explanations to bridge gaps in their understanding and improve trust and acceptance.

In order to investigate how our findings can be applied in a real-world setting, we are currently developing a context-aware extension for the AOL Instant Messenger (AIM) that uses predictions of buddy responsiveness to instant messages (based on [5]). In a forthcoming longitudinal deployment we plan to investigate how explanations affect usability and acceptability.

Implications for Context-Aware Applications

While our intelligibility test infrastructure has some characteristics of context-aware systems, real context-aware applications are more complex and several issues would have to be handled regarding the provision of intelligibility type explanations. Firstly, applications that use decision tree models tend to have much larger trees learned from possibly hundreds of features, and it would not be scalable to generate explanations from them. For example, a tree of depth 13 could lead to the Why traces that have over 10 inequality relations. The explanations returned would be too long for users to assimilate and remember. One way to deal with the larger tree size is to just provide subsets of reasons in the explanations. For example, the Why trace could just provide the top 5 inequality relations ranked by how much each relation affects the prediction accuracy. Providing subsets of explanations would provide users with only partial understanding of each application behavior instance, and users may have to interact with the system longer before understanding the system better. One way to reduce overall learning time may be to start new users with higher-detail explanations, then progress to less detail the more they interact with the system.

While our setup dealt with decision tree learners, the Naïve Bayes classifier is another popular learner used in context-aware applications. Even though they are not as intuitive as decision trees, Naïve Bayes models can be interpretable, and there are several visualizations to explain them (*e.g.*, nomograms [7]). However, some learners (*e.g.*, Support Vector Machines, Neural Networks) are considered black-boxes and are not inherently interpretable. Fortunately, there have been some attempts to make them explainable using decision trees or rules (*e.g.*, [3]). We can then use the same techniques to provide explanations for systems based on decision tree models.

Another issue with real systems is that users may not like to receive explanations *all the time*, but *on demand* instead, because the former may be too obtrusive. We would like to run a future study to compare if users can still benefit sufficiently from explanations if they get to choose when and how often they can receive explanations, and if explicit effort in asking for explanations can improve learning.

Our results suggest the effectiveness and importance of providing Why and Why Not explanations over How To and What If. The former two deal with Norman's gulf of evaluation, while the latter two deal with the gulf of execution [21]. While we feel that this dichotomy should remain true for informative context-aware systems (*e.g.*, applications to determine interruptibility of others to inform onlookers [5, 24]), systems that are more pro-active (*e.g.*, applications that send notifications based on the user's interruptibility) may benefit more with the How To and What If explanations. With those explanations, users would be better informed of how they can carry out their tasks.

FUTURE WORK

The design of the generalized web-based intelligibility system lends itself to be easily modified to investigate other factors for providing explanations. We are interested in investigating the effects of varying the number of times participants receive explanations and whether the benefits would plateau or drop after a certain amount. For on demand invocations of explanations, we are interested in exploring how user behavior and perceptions change when varying the cost of asking for explanations, or the cost of the user making mistakes during task performance. Given that our results suggest that Why explanations are more effective than Why Not, we would like to investigate whether users would also prefer Why explanations over Why Not, in a within-subject study. Furthermore, though our results do not show the effectiveness of How To and

What If explanations, we believe that tasks more oriented toward explicit user action may benefit more from these intelligibility types of explanations. We plan to run another study with tasks focusing on user action rather than understanding to explore the effects of these explanations.

Visualizations are a popular medium to provide insight to data or how a system works, and are employed in knowledge-based systems. In particular, for decision trees, showing the tree and tracing the Why trace would provide a quickly interpretable overview of the 5 intelligibility questions: What the system did, Why it did it, Why Not do something else, How it could do something, and What it would do If an input condition changes. However, some technical knowledge is required to properly utilize the tree visualization. We would like to conduct a study exploring such a "Complete" explanation with subjects having variable technical knowledge.

Finally, we plan to build a toolkit that will allow developers to easily make their intelligent context-aware applications intelligible (drawing on lessons from Situations [10], and PersonisAD [4]).

CONCLUSIONS

We have described a large controlled study comparing the provision of explanations addressing four intelligibility type questions (*why*, *why not*, *how to*, and *what if*). We developed a web-based infrastructure that provides a functional input-output interface of an intelligent system prototype that provides different types of explanations. Our findings suggest that providing reasoning trace explanations for context-aware applications to novice users, and in particular Why explanations, can improve user's understanding and trust in the system. This work is a necessary first step into understanding the impact of explanations in context-aware applications.

ACKNOWLEDGEMENTS

We also thank Niki Kittur, Noboru Matsuda, Vincent Aleven, Sara Kiesler, and Andy Ko for their insight and feedback. This work was funded in part by the National Science Foundation under grant 0746428.

REFERENCES

1. Amazon Mechanical Turk. http://www.mturk.com. Retrieved 16 Sep 2008.
2. Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences*, 4(2), 167-207.
3. Andrews, R., Diederich, J. & Tickle, A. (1995). A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge Based Systems*. 8: 373–389.
4. Assad, M., Carmichael, D.J., Kay, J. & Kummerfeld, B. (2007). PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. *Pervasive 2007*, 55-72.
5. Avrahami, D. and Hudson, S.E. (2006). Responsiveness in Instant Messaging: Predictive Models Supporting Inter-Personal Communication. *CHI'06*, 731-740.
6. Bellotti, V. & Edwards, W.K. (2001). Intelligibility and Accountability: Human Considerations in Context-Aware Systems, *Human-Computer Interaction*, 16(2-4): 193-212.
7. Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A. & Lopes, C. (2002). Making sense of sensing: Five questions for designers and researchers. *CHI'02*, 415-422.
8. Cheverst, K., Byun, H.E., Fitton, D., Sas, C., Kray, C. and Villar, N. (2005). Exploring issues of user model transparency and proactive behavior in an office environment control system. *User Modeling and User-Adapted Interaction: Journal of Personalization Research*, 15(3-4), 235-273.
9. Davis, R., Buchanan, B., and Shortliffe, E. (1977). Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, 8(1): 15–45.
10. Dey, A.K. & Newberger, A. (2009). Support for contxt-aware intelligibility and control. To appear in *CHI'09*.
11. Dourish, P., Adler, A. & Smith, B.C. (1996). Organising User Interfaces Around Reflective Accounts. *Reflection'96*, 235-244.
12. Dzindolet, M., Peterson, S., Pomranky, S. Pierce, L. & Beck, H. (2003). The role of trust in automation reliance, *Int'l Journal of Human-Computer Studies*, 58(6): 697-718.
13. Google Web Toolkit. http://code.google.com/webtoolkit/. Retrieved 16 Sep 2008.
14. Gregor, S. & Benbasat, I. (1999). Explanations From Intelligent Systems: Theoretical Foundations and Implications for Practice. *MIS Quarterly* 23(4): 497–530.
15. Herlocker, J., Konstan, J. & Riedl, J. (2000). Explaining collaborative filtering recommendations. *CSCW 2000*, 241-250.
16. Ko, A.J. & Myers, B.A. (2004). Designing the WhyLine: A debugging interface for asking questions about programm failures. *CHI 2004*, 151-158.
17. Mozina M., Demsar, J., Kattan, M., & Zupan, B. (2004). Nomograms for Visualization of Naive Bayesian Classifier. *PKDD 2004*, 337-348.
18. McGuinness, D.L., Glass, A., Wolverton, M. & Pinheiro da Silva, P. (2007). A Categorization of Explanation Questions for Task Processing Systems. *AAAI Workshop on Explanation-Aware Computing*.
19. Muir, B. (1994). Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics*, 37(11): 1905–1922.
20. Myers, B.A., Weitzman, D., Ko, A.J. & Chau, D.H. (2006). Answering Why and Why Not Questions in User Interfaces. *CHI 2006*, 397-406.
21. Norman, Donald A. (1988). The Design of Everyday Things. New York, Doubleday.
22. Nugent, C., & Cunningham, P. (2005). A case-based explanation system for black-box systems. *Artificial Intelligence Review*, *24*(2): 163–178.
23. Schilit, B.N., Adams, N.I. & Want, R. (1994). Context-aware computing applications. *1st International Workshop on Mobile Computing Systems and Applications*, 85–90.
24. Tullio, J., Dey, A.K., Fogarty, J. & Chalecki, J. (2007). How it works: A field study of non-technical users interacting with an intelligent system. *CHI 2007*, 31-40.
25. Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.
26. Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3): 94-104.
27. Weiser, M. & Brown, J.S., (1995). Designing Calm Technology. *PowerGrid Journal v1.01*.