Improving Trust in Context-Aware Applications with Intelligibility

Brian Y. Lim
Carnegie Mellon University
byl@cs.cmu.edu

ABSTRACT
Since context-aware applications use implicit sensing and increasingly complex decision making, they may make mistakes or users may misunderstand their actions. This may hinder trust and adoption of context-aware applications. We hypothesize that making these applications intelligible by explaining themselves to users would help counter this lack of trust. The proposed thesis would contribute to context-aware computing by (i) understanding the need to explain these applications to users, (ii) understanding the benefits and trade-offs of providing intelligibility, and (iii) providing toolkit support intelligibility to ultimately improve the trust, adoption of, and sustained use context-aware systems.

Author Keywords
Context-awareness, intelligibility, explanations, toolkits.

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

General Terms
Human Factors.

INTRODUCTION
Context-aware applications [5] make use of sensed inputs and contexts, coupled with intelligent decision-making to automatically and calmly adapt to serve users. However, the implicit nature of context sensing, and the growing complexity of models (e.g., rules, hidden Markov models) underlying context-aware applications make it difficult for users to understand them (e.g., [12]). Lay users may not understand how these applications make their decisions, let alone be aware when decisions are made and actions are taken. This can lead to user frustration, and ultimately a loss of trust in the applications. To counter this, context-aware applications should be intelligible [2, 9, 10, 11] by generating explanations of application behavior.

Thesis Statement
My thesis posits that when users understand context-aware applications, they will be able to appreciate the nuances and idiosyncrasies involved with such complex systems; they will become less suspicious and more trustful of these systems. With this trust comes increased adoption, and increased and improved (knowledgeable) usage of context-aware applications. Hence, we have set out to investigate what information users want to know about how context-aware applications work, how this understanding impacts their interaction with and opinion of these types of applications, and have developed support to make context-aware applications more intelligible.

Expected Contributions
My thesis is organized into three expected contributions:

1) Assessment of user information needs of context-aware applications (demand for intelligibility). So as to provide salient explanations of context-aware applications to users, we should learn the users’ gaps in knowledge, and what they desire to know. My thesis will explore these needs for explanations, and how they vary across applications and usage circumstances. Results from this investigation would be composed into a set of design recommendations on what explanations should be provided to make context-aware applications intelligible.

2) Understanding of benefits of and trade-offs with intelligibility. The second contribution of my thesis seeks to validate the hypothesis that explaining context-aware applications would improve user understanding and trust. We investigate whether users use intelligible context-aware applications more than non-intelligible ones. We expect users to more effectively engage with and control intelligible context-aware applications since they understand better how they work. We also expect there to be trade-offs and issues arising from providing intelligibility. Applications will need to convey more information to explain their actions to users, and this could increase obtrusiveness. Furthermore, context-aware applications that facilitate social interactions (e.g., awareness displays) may compromise the privacy of users, particularly when they expose privacy sensitive information in order to explain themselves. This thesis seeks to uncover some of the concerns and compromises users encounter when context-aware applications are made intelligible. It would also explore the efficacy of providing explanations of context-aware applications.

3) Support for intelligibility in context-aware applications (Intelligibility Toolkit). Armed with a set of requirements for making context-aware applications intelligible, it may still be difficult to implement these requirements. Developers will need to intimately understand the decision making mechanisms (e.g., various machine learning algorithms) they employ and generate the various explanations as recommended. The third contribution facilitates building intelligible context-aware applications by providing automatic explanation generation.
in an intelligibility toolkit. With the toolkit, developers would be able to invoke the generation of explanations for commonly used decision models without worrying about the details of these underlying models. The toolkit can also facilitate further exploration of the design and usage of intelligibility, since it will be easier to compose explanations. Hence this would facilitate the creation of better designed explanations for context-aware applications.

RELATED WORK
Explanations are well researched in fields such as knowledge-based systems (KBS), recommender systems, end-user programming [7, 8], and even specifically for context-aware applications [3, 4, 6, 12]. Explanations have been provided for KBS, expert systems, and task agents for several decades, but they target skilled or novice workers in knowledge intensive domains. My thesis focuses on lay users of context-aware applications performing everyday activities. Recommender systems also make use of explanations to justify their suggestions to users. These tend to explain preference models and collaborative filtering processes, and are content-centric. Context-aware applications tend to use physical sensors of the real environment so interfaces and information that are needed to explain them would be intrinsically different.

Cassens et al. [3] provided a theoretical framework for explaining context-aware systems using activity theory and semionics. They propose problem frames to capture explanations for transparency, justification, relevance, conceptualization, and learning. My thesis adds to this work by providing a technical contribution via a toolkit, and an HCI contribution by investigating the impact (pros/cons) of intelligibility on usability.

There are a few toolkits and frameworks that have been built to support explanations in context-aware applications. PersonisAD [1] provides an accretion / resolution mechanism that can propagate evidence when contextual rules are satisfied. Enactors [6] similarly expose rules from lower level contexts, but also provide support for exposing the logic to users (developers and designers). These frameworks provide basic reasoning traces, while my thesis considers explanations in a more diverse form. Specifically, we draw from the question-type (Why and Why Not) explanations of Ko et al.’s Whyline [7], and extend them to support more question types (e.g., How To, What If, Control). The Whyline reduces the programmer’s ignorant speculation about program error (or misconception) by allowing them to ask questions explicitly about the program output. However, given the implicit nature of context-aware applications, and lack of programming know-how, lay users may ask a wider range of questions.

HCI research of explanations — specific needs for, interaction with, and impact of explanations — have been well researched in other fields (e.g., KBS, recommender systems), but little prior work has been done specifically for context-aware systems (exceptions: e.g., [4, 8, 12]). My thesis would extend this work by exploring the use of intelligibility over a broader range of context-aware applications. We would investigate how users use intelligibility differently across different dimensions (e.g., application, function, situation, and behavior).

METHODOLOGICAL APPROACH
The general approach to test my main hypothesis is to build prototypes of varying degrees of fidelity and run user studies (online, in the lab, in the field) to investigate the need for and impact of intelligibility.

1) Assessing demand for intelligibility. We first survey and interview a large number of participants regarding their information needs for a range of context-aware applications across a variety of circumstances. This allows us to investigate needs more generally. We run the study by presenting scenarios and mock-ups to online participants and subsequently survey and interview them. With the broad understanding, we can focus on situations and applications where intelligibility would have a more important impact. To gain a more realistic understanding of the need for intelligibility, we develop and deploy intelligible and non-intelligible versions of context-aware applications for users to engage with and articulate their information needs. Some applications are deployed over short periods of time for lab studies, and others are deployed longitudinally for long-term studies.

2) Investigating benefits and trade-offs of intelligibility. Similar to the investigation of user information needs, we employ intelligible context-aware application prototypes that span a range of fidelity (low-fi to high-fi) to study how participants use the applications. We study whether they gain any benefit of understanding and trust of the applications, of task performance, and of satisfaction, etc. We also study how other factors (such as privacy, and obtrusiveness) interact with the provision of intelligibility. Some factors are studied as dependent variables (e.g., privacy concerns, controllability) and others are manipulated as independent variables (e.g., obtrusiveness, application reliability). We run comparative studies of intelligible and non-intelligible versions of the applications so that we can have a baseline comparison. We also run case studies by analyzing how participants use deployed intelligible applications, particularly in the long run.

3) Intelligibility Toolkit. Having elicited the user needs for intelligibility, we have a list of explanation types that context-aware applications should provide. We seek to facilitate the provision of these explanations from context-aware applications through the development of an intelligibility toolkit. The toolkit would not only make it easier to implement explanations, but would also promote and facilitate the design of appropriate explanations. With this support, it also becomes easier to develop intelligible context-aware applications that would also facilitate subsequent investigations of intelligibility. This, in turn, would feed into the further improvement of the
In the presented application, we could predict missing inputs and outputs in 15 test cases, and by asking them to explain how they think the application reasons. We also measured their level of trust in the output of the application. We found that participants who received Why and Why Not explanations better understood and trusted the application than those who received How To and What If explanations.

Assessing Demand for Intelligibility
This study [10] investigated which explanations users of context-aware applications wanted, so that by providing these explanations, we could maximize user satisfaction. We presented 860 online participants with video scenarios of four prototypical context-aware applications under various circumstances along the dimensions of application behavior appropriateness, situation criticality, goal-supportiveness, recommendation, and number of external dependencies. We elicited and subsequently solicited (validation) what information participants wanted to know under the various circumstances and extracted 11 types of explanations of interest. We also found how the demands for the explanations varied with circumstances (e.g., explanations of all types are highly desired for critical situations, and Why Not explanations are highly desired for goal-supportive applications such as reminders). We presented our results as design recommendations of when context-aware applications should provide each of the explanations (see Table 1).

Intelligibility Toolkit for Context-Aware Applications
With a design framework in place from [10], this work makes a technical contribution by facilitating the provision of 8 explanation types (Input, Output, What, Why, Why Not, How To, What If, Certainty) generated from commonly used decision models in context-aware applications (rules, decision tree, naive Bayes, hidden Markov models) [11]. The Intelligibility Toolkit (see Figure 1) extends the Enactor framework by providing more types of explanations and supporting machine learning classifiers other than rules. We validate the toolkit with three demonstration applications showing how the explanations can be generated from various decision models.

REMAINING RESEARCH
The Intelligibility Toolkit is pivotal to the remaining research that we intend to conduct regarding intelligibility.
in context-aware applications. It allows us to more easily develop and design intelligibility features into various context-aware applications. The remaining work for the thesis regards validating the need for and impact of intelligibility through more realistic intelligible prototypes, and the investigation of secondary factors that interact with the provision of intelligibility. See Table 2 for the plan of completing the remaining research.

**Real-world use of Intelligibility**

While the previous user studies were informative to elicit user need for intelligibility and give us a glimpse of what users could learn from intelligible context-aware applications, letting users use a working intelligible application would allow us to study how they actually use the intelligibility features. This would give us stronger ecological validity. We intend to conduct a study with a working prototype that participants can use, and run think-aloud studies to learn usage patterns and rationale for use.

**Longitudinal Intelligibility**

We are also interested in the long-term impact and use of intelligibility. We expect intelligibility to help users learn how context-aware applications work and that users would ask for more explanations earlier in their interactions with a new context-aware application, and that their use would taper off in time. We also expect users to ask for explanations for exceptional cases, and a longitudinal study would help reveal those critical incidences and also provide stronger ecological validity, since users would have more realistic (vs. contrived) goals for using explanations.

**Intelligibility and Control**

Intelligibility can help users learn to trust context-aware applications, but users should also use their improved understanding to better control these applications. However, there are various control models for users to manipulate context-aware applications and this may influence what kinds of explanations are best to facilitate effective control. We intend to investigate how intelligibility can help users more knowledgeably and effectively control and configure context-aware applications.

**Intelligibility and Reliability**

We hypothesize that intelligibility will help increase user trust and adoption for context-aware applications that seem less reliable (e.g., due to misunderstanding, or poor initial training of the system). This study would investigate how providing intelligibility helps to improve user trust for at least two versions of an application with higher and lower performance accuracy. An alternative hypothesis for the impact of intelligibility of less reliable applications is that the explanations may also be so severely compromised that this would hurt user trust even more.

**Intelligibility and Privacy**

The exposure of the inner workings of context-aware applications lead to interesting implications for privacy, particularly for social awareness applications. For example, users may not want contacts to know that they are available for a phone call because they are not at the office. We intend to investigate how much privacy users are willing to compromise for improved understanding.

**CONCLUSION**

We hypothesize that the addition of intelligibility would promote trust and adoption of context-aware applications. My thesis would test this hypothesis by (i) accessing this demand by eliciting user need for intelligibility, (ii) measuring the efficacy and issues of providing intelligibility, and (iii) providing technical support for making context-aware applications intelligible. Preliminary work indicates positive results towards the hypothesis, while further work would validate it with higher fidelity prototypes across a wide variety of domains.

**REFERENCES**


<table>
<thead>
<tr>
<th>Project</th>
<th>Duration</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-world use</td>
<td>~4 mths</td>
<td>Jun 2010</td>
<td>Sep 2010</td>
</tr>
<tr>
<td>Privacy</td>
<td>~3 mths</td>
<td>Nov 2010</td>
<td>Dec 2010</td>
</tr>
<tr>
<td>Longitudinal</td>
<td>~6 mths</td>
<td>Nov 2010</td>
<td>Apr 2011</td>
</tr>
<tr>
<td>Control</td>
<td>~4 mths</td>
<td>May 2011</td>
<td>Aug 2011</td>
</tr>
<tr>
<td>Reliability</td>
<td>~3 mths</td>
<td>Jul 2011</td>
<td>Sep 2011</td>
</tr>
<tr>
<td>Thesis Writing</td>
<td>~6 mths</td>
<td>Aug 2011</td>
<td>Feb 2012</td>
</tr>
</tbody>
</table>

Table 2. Plan for completion. Estimated schedule for remaining research. Timeline includes planning, development, user study, and analysis.