Intelligibility and User Control of Context-aware Application Behaviours

Johnson Fong[1, 2]
[1] The University of Queensland
School of Information Technology and Electrical Engineering,
[2] National ICT Australia (NICTA)
Queensland Research Laboratory
jfong@itee.uq.edu.au

ABSTRACT
Context-aware applications adapt their behaviours according to changes in user context and user requirements. Research and experience have shown that such applications will not always behave the way as users expect. This may lead to loss of users' trust and acceptance of these systems. Hence, context-aware applications should (1) be intelligible (e.g., able to explain to users why it decided to behave in a certain way), and (2) allow users to exploit the revealed information and apply appropriate feedback to control the application behaviours according to their individual preferences to achieve a more desirable outcome. Without appropriate mechanisms for explanations and control of application adaptations, the usability of the applications is limited. This paper describes our on going research and development of a conceptual framework that supports intelligibility of model based context-aware applications and user control of their adaptive behaviours. The goal is to improve usability of context-aware applications.

Keywords
context-aware, intelligibility, user preferences, user control, user feedback, personalisation.

1. INTRODUCTION
Context-aware applications use contextual information about users to evaluate whether there is any change to their situation and environment that requires the applications to adapt their behaviour automatically on behalf of users [1]. They have the potential to greatly reduce human interactions with computing devices, thus giving users the impression that computing services have faded into the background. However, one of the main problems with context-aware applications is that the adaptations they automatically performed will not always result in the application behaviours that users expect due to various reasons, such as imperfect sensing and reasoning of/on context information and inaccurately captured user preferences, etc.

Hence, users may feel loss of control over the behaviours of their applications, become dissatisfied and abandon the most useful context-aware applications eventually. To mitigate these problems, (1) context-aware applications should be intelligible: being able to "represent to their users what they know, how they know it, and what they are doing about it" [4]. The users need to understand how the applications operate and how adaptation decisions are made (such as which context information and logic are used, and what rules and models are employed to arrive at certain automated actions). Based on this information, (2) the users may decide to alter the adaptive behaviour and personalise it according to their preferences, to achieve a more desirable outcome. Without appropriate mechanisms for explanations and control of application behaviours, the usability of the applications is limited.

Most of the current middleware solutions for context-aware applications support gathering and management of context information, and evaluation of situations and user preferences, while some are more technology specific focusing on specific problems (e.g., fault-tolerance, user privacy, etc.). However, few middleware solutions address the issue of intelligibility and user control of application behaviours. This paper describes the most relevant work in this area, as well as our ongoing research and development of a conceptual framework that improves user experiences by supporting intelligibility of model based context-aware applications to facilitate user understanding of application actions, and user control of their adaptive behaviours.

More specifically, the research focuses on (1) a model for explicit exposure and explanation of the internal adaptation decision making process of context-aware applications, and (2) a method that allows users with various skill levels in technology to customize the context-aware behaviour dynamically at run time (without the need for reimplementation) according to their individual needs and requirements. The structure of the paper is as follows. Section 2 present related works in the area of intelligibility and user control of application behaviours, Section 3 discusses the research goal and methodology of our work Finally, Section 4 summarises the paper with the results to date, outlines any future work to be completed and potential challenges to the work.

2. RELATED WORK
This section briefly reviews relevant approaches in the area of intelligibility and user control of context-aware application behaviours.

ICPS2010, July 13-15, 2010, Berlin, Germany.

2.1 User Studies

Lim et al. carried out extensive user studies in [5] and [6], and discovered that explanations describing *why* an application behaved a certain way resulted in better understanding and stronger feelings of trust from users. Contrastingly, the users' trust decreases when the explanations of the behaviour have low accuracy.

In addition, separate user studies conducted in [16] found that context-aware applications providing intelligibility types such as *why*, *certainty* and *control* (as described in Table 1) are helpful in improving users' satisfaction, and should be made available for all context-aware applications, while some are more useful for specific contexts (*e.g.*, *why not* for goal-supportive tasks).

Table 1. Description of Intelligibility Types

Intelligibility Types	Description
Why	*Why* did the application do X?
Why Not	Why did it not do Y?
What Else	*What (else)* is it doing?
How	*How* (under what condition) does it do Y?
What If	*What if* there is a change in conditions, what would happen?
Certainty	Application confidence of its action / decisions
Control	How to change settings / thresholds

Although, these experiments and user studies are useful in understanding the impact of explanations in context-aware applications, formal models and solutions to achieve intelligibility in applications are yet to be designed and developed.

2.2 Middleware and Infrastructures

There are several existing middleware solutions that support the development of context-aware applications. For example, Henricksen provided a generic solution with formal and well defined models capturing of context information, and reasoning on situations and user preferences. While some others are more technology specific focusing on specific problems such as, fault-tolerance, resource discovery, user privacy and security, etc. Examples include, CoBrA [7], Context Toolkit [10], JCAF [8], Cooltown [9], and the PACE middleware [15]. However, they do not offer higher level abstractions that explain application adaptations and the decision making process involved in those adaptations (i.e., what the context and logic were employed) in an accessible manner.

The exceptions to the infrastructures above are extensions to the Context Toolkit [10] and the PACE middleware [15]. Their extension addressed aspects of intelligibility and control in context-aware applications. Newberger and Dey [13] extended the Context Toolkit, to include a new component, the Situation [13] (or Enactor in earlier publications [12]). It provides external access to application logic at the toolkit level through a standard API, and facilitates such access through interface design tools. The Situation allows context-aware application developers to implement specific application logic and expose the design-time and run-time characteristics of that logic. Situation ties together context input and application output in a way that inherently supports external access to internal application logic. It has three subcomponents: references, parameters, and listeners. References acquire context data from widgets, listeners monitor all changes, and parameters allow control. Although this solution is primarily intended for use by application designers, it is easy to see that a similar solution could be developed to allow users to personalise and control context-aware behaviour.

While Dey et al. focused the research on supporting developers for building intelligible context-aware applications, Hardian et al [14] focused on a generic approach for supporting users in scrutinising contextual information in applications that are based on models (i.e., context models, situation models and preference models). They extended the PACE middleware for exposing adaptation decisions on user requests. Their work identified three elements within PACE that influence behaviour of context-aware applications. They are (1) context facts that can be abstracted using situation abstraction to describe the condition or scenarios to which context-aware application should adapt, (2) preference information that accommodates explicit user preferences and (3) choices that link adaptation logic to the preferences for it to be executed by the applications under a particular situation.

These elements are being maintained by various models in the PACE middleware. Hardian et al's approach [14] is able to reveal these internal elements to users, their aim is to provide users with better understanding to the inner workings and inputs of applications. Although their work is able to provide *transparency* to the internal adaptation decisions by revealing elements that influence application behaviours, these elements are not necessary intelligible (i.e., they are exposed in a low-level form that is currently being maintained in the system). Without appropriate mechanisms for explanations, they cannot easily be understood by ordinary users (i.e. users do not have expertise in information technology), because the models that maintain these elements are primarily designed for application developers to interpret and input preferences on behalf of the users. In addition, Hardian et al's approach does not accommodate sufficient feedback mechanism for users to control application behaviours.

3. RESEARCH GOALS AND METHODS

Our research has three components: (1) investigation of the architecture of the PACE middleware, and analysis of various models and elements that influence the adaptive behaviour of context-aware applications; (2) utilisation of results from the investigation to design and develop a model for explicit exposure and explanation of the adaptation decision making process in the applications, and methods that allow users to customize, at run time, the context-aware behaviour according to their individual preferences; (3) implementation of a proof-of-concept prototype as an extension of PACE, and evaluation the effectiveness of the model for improving usability of context-aware applications by performing a user study.

3.1 Analysis of PACE models

In order for users to correct inappropriate application behaviours, the internal states of the applications such as the current context and preference information need to be captured in formal and well defined models at the first place, so that the information can be revealed in a structured manner. Thus, our work needs to be supported by a context-aware infrastructure that offers formal context and preference modelling abstractions, so as to provide a generic solution to context-aware applications (independent of their application types or domains) for improving their intelligibility and control.

We evaluated several infrastructures for their context and preference modelling and reasoning approaches in our critical literature survey. The results show that the PACE infrastructure, proposed by Henricksen et al [15], is one of the few solutions that provide a rich and comprehensive set of middleware models, and therefore, we employ it as the basis of our research. As the starting point of our research, we analysed the middleware models from Henricksen et al's framework aimed to find out (1) which models or elements influence the application behaviours and (2) where the difficulties and challenges lie in exposing them in a way that is acceptable and understandable for users.

Preliminary analysis of the PACE infrastructure discovered four *middleware models* that are of relevance to application adaptation. The first model is the CML based *context model* for capturing context information (context facts) used in a context-aware application. The Context Modelling Language [2], or CML as it is known, provides a graphical notation that assists designers with the tasks of exploring and specifying the context requirements of applications (i.e., modelling context information fact types that are of interest to applications). Figure 1 depicts a modified version of a CML context model for a smart home application presented in [11].

The second model is the *situation abstractions* for describing high-level context information derived by context reasoning on context facts in the context model. Context facts are used to create another level of context abstractions known as situations, as context-aware applications adapt to particular situations, not to changes in single context information fact. We base our research on situation models that express situations using a variant of predicate logic (a combination of context facts supported by universal quantifiers and/or existential quantifiers) as defined in Henricksen et al's framework. An example of a situation definition `PersonHasFallen(person)` required by the smart home application is shown in Figure 2.

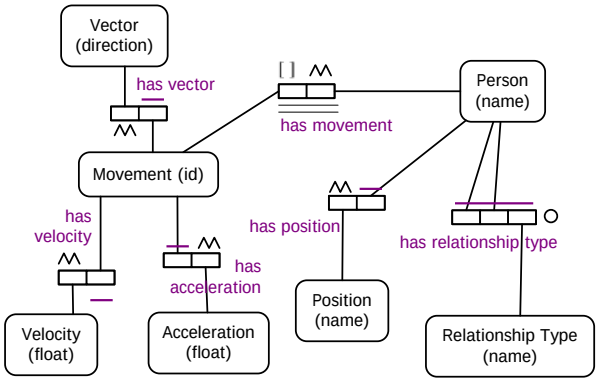

Figure 1. Portion of a CML context model of a smart home application [11].

The third model is the *preference model* for representing user preferences. The preference model builds on the situation abstraction. Preferences work based on a scoring mechanism - users can assign a score in a particular context situation. The scores assigned by user's preferences can be dynamically combined with system's preferences, to support decision making on how a context-aware application should adapt.

```
PersonHasFallen(person):
        ∃ Movement
        • HasAcceleration [movement, acceleration]
                • acceleration > 5.8 ms²
        • PersonHasMovement
                [Person, movement, t₁, t₂]
                • (t₁ ≥ (timenow() - 5 sec) )
        • MovementHasVector[movement, vector]
                • vector = "Downward"
```

Figure 2. An example situation required by the smart home application

The scope describes the context in which the preference applies in terms of situations. A preference is considered applicable within a given context only if the scope expression is true. The scoring expression assigns a score to a preference and increasing scores represent increasing desirability. Each preference is linked to several possible adaptations with a set of candidate choices. Depending on what context the preference is evaluated against, an appropriate adaptation will be executed. The *candidate choice* is the fourth middleware model that is of relevance to application adaptations. Choices are for linking preferences with adaptation logics through a set of valuations and their binding as shown in Figure 3. Figure 3 illustrates a simplified version (ignoring details such as triggers and `selectbest()` function call, etc.) of the procedures that the smart home application is required to take, in order to decide which action is to be executed when the occupant `Mary` has fallen. Initially, step (1) from Figure 3 shows each binding in the candidate choice 1 (i.e., `Dr Joe` and `Mary`) is substituted into the each situation variable (i.e., `Occupant` and `ContactPerson`) of preference P1. Using the choice bindings, step (2) shows all the situations in that preference P1 is then evaluated against the current context held in the context model. (3) When all the situations in P1 return true, choice 1 scores a rating of one. Above procedures are then repeated for all preferences P1 to P5, averaging all the ratings that choice 1 has scored. Finally, the last procedure is repeated for all choices 1 to 3, the choices are then compared to determine which one has the highest rating and the adaptation associated with that choice is executed.

From the analysis, it is evident that the current middleware models are not easy to understand even when they are fully exposed (depicted in Figure 1 to 3) and explained accordingly. It is not obvious for users to see what the final adaptation will be given the current context, situations and user preferences. In this smart home example, assuming Mary is unconscious after the fall and it is currently outside working hours, the final action the application takes in this case is to contact `Mary`'s next of kin, `Ms Helen` (i.e., `choice 2`) rather than one of the doctors. However, it is not entirely clear for users as to how or why this has happened. The underlying reason is that choice 2 has been given the highest rating with an average score of 1, where choice 1 and 3 have both scored an average rating of 0.9.

While the middleware models have high transparency (i.e., able to expose their internal states to users), the lack of intelligibility in the revealed information is still causing difficulties for users when they attempt to exploit the exposed information (i.e., trace inappropriate behaviours back to incorrect context or preference information, and correct this information). One of the fundamental reasons for the lack of intelligibility in the models is caused by the cumbersome procedures in order to evaluate a set of preferences by substituting choice bindings into each of the situations and calculate an overall rating for a particular choice.

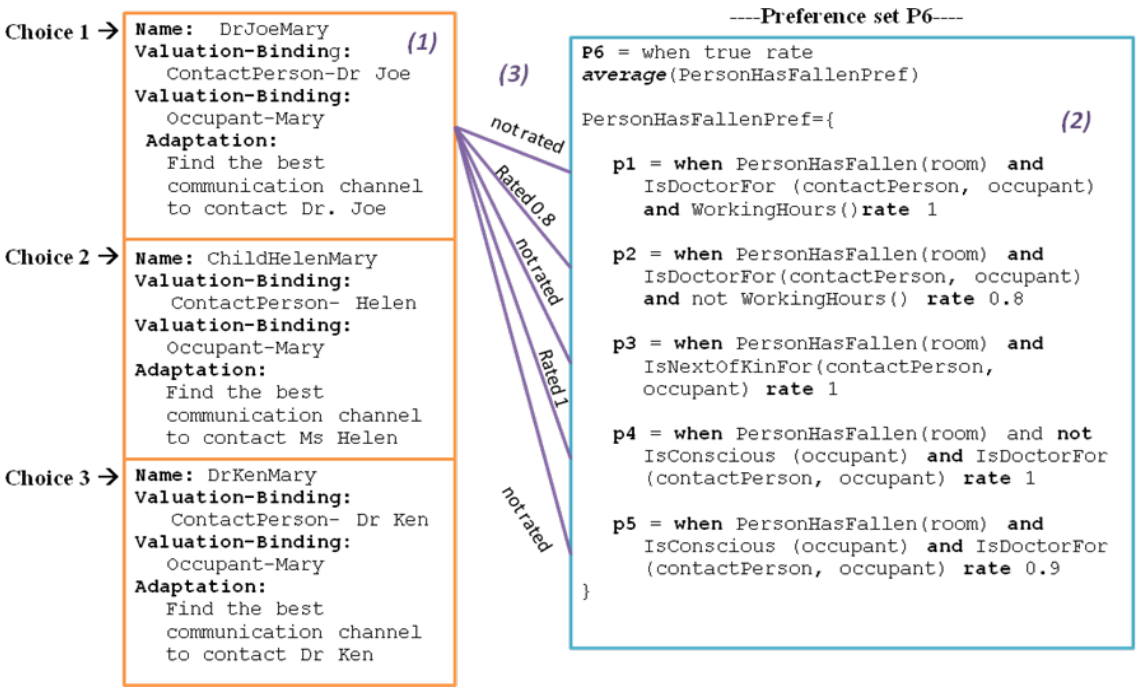

Figure 3. Choices and Preference model

For example, when all preferences have evaluated against one particular choice and given it a set of ratings, a utility function (e.g., `average (P)`, `wgtaverage (<Pa,...,Pn>, <W1,...,Wn>)`, `override (P1,P2)`, and `as (P)`, etc.) has to be executed on the combination of ratings for the choice[15]. The application can decide which adaptation to perform only when all existing choices have been evaluated and the final set of ratings are calculated by whatever the function is associated with the preferences.

3.2 Design of models for intelligibility and user control

Toward the second objective, we hypothesize that the lack of intelligibility in context-aware applications can be addressed by the following proposals. Firstly, in this research (1) we advocate for an intelligible preference model in terms of "if-then-else" decision rules derived from Henricksen et al's middleware models [15]. It is expected to have a higher intelligibility than the existing preference model, as it represents a closer mapping to users' cognition of preference representations. And secondly, (2) based on Hardian et al's approach [14], we design and implement a user feedback system for (i) user scrutiny of the middleware models with explanations for users with various expertises in I.T., and (ii) user feedback on application behaviours by changing their preferences, and/or introducing new context fact types and situations.

As discussed in the analysis, explanations of preference information are very challenging with the existing middleware models. Untrained users have significant difficulties understanding the complicated procedures for preferences and choices evaluation, as the models were originally designed by application developers and their main goal was to externalise context and preference evaluation from the application to middleware. Therefore, adaptations performed by the applications often represent the perspective of the developers, causing usability issues in applications. Hence, we propose an intelligible preference model that is designed with the users in mind, by removing unintelligible aspects of the existing Henricksen et al's models such as the utility function in the preference model (`wgtaverage` and `override`, etc.), and extending the choices to enhance their expressiveness. An example of the proposed model and its set of choices are shown in Figure 4 and 5, The model extends Henricksen et al's middleware models (in particular the preferences model and candidate choices) for it to be automatically mapped and expressed in terms of "if-then-else" decision rules, as shown in Figure 6.

It is expected to be more intelligible than the existing Henricksen et al's preference model, as "if-then-else" expression speaks the language of most users (i.e., understandable by users). We will evaluate to which extent the model meets requirements such as expressiveness, support for conflict resolution, externalisation of context/situation evaluation, simple and user friendly composition of preferences, etc. There may be some losses with regard to these requirements compared with the existing Henricksen et al's model. We will evaluate the extent of the losses created by a tradeoff between meeting all the requirements and intelligibility of context-aware applications.

Figure 6 shows an "if-then-else" mapping of the proposed preference model with preferences set `PersonHasFallenPref` (Figure 4) and `Choice Set 1` (Figure 5). The preferences being expressed are similar to the user preferences shown in Figure 3; both lead to the same adaptation under the same situations and assumptions from the pervious smart home example. One of the differences between the two preference models is that, preferences represented using the proposed model, users can immediately see the why Mary's daughter is being contacted (i.e., preference P2 is selected) rather

than her doctors when she has fallen. This is simply because Mary's fall occurred outside the working hours, and a lower weighting is assigned to P3 probably because Mary initially felt it may not be appropriate to contact the doctor outside working hours. Hence, P1 and P3 are not selected, and thus the less desirable adaptation associated with P2 is performed. By employing the intelligible preference model, Mary can now understand the adaptation decision process, and together with a user feedback system that is being proposed and discussed below, Mary can also modify the preferences according to her own needs and requirements, thereby effectively bringing the application behaviours under her control. This would be very difficult to achieve with the existing preference model and choices.

PersonHasFallen(room):	Weight	Choice
P1 = **WHEN** IsDoctorFor (ContactPerson, Occupant) **AND** ***WorkingHr()*** **AND** ~Isconscious(Occupant)	**1**	Set **1**
P2 = **WHEN** **IsNextOfKinr** **(ContactPerson, Occupant)**	**1**	Set **1**
P3 = **WHEN** IsDoctorFor (ContactPerson, Occupant) **AND** ~WorkingHr()	**0.9**	Set **1**

Figure 4. Preference set `PersonHasFallenPref` containing a set of preferences for the proposed intelligible preference model.

Choice Name	Valuation-Binding	Adaptation	*Weight*
C1	ContactPerson-Dr Joe Occupant-Mary	Contact Dr Joe	***1***
C2	ContactPerson-Dr Ken Occupant-Mary	Contact Dr Ken	***0.9***
C3	ContactPerson-Ms Helen Occupant-Mary	Contact Ms Helen	***1***

Figure 5. `Choice Set 1` containing a set of choices for the proposed intelligible preference model.

```
As PersonHasFallen(room),

    IF IsDoctorFor (Dr Joe, Mary) AND WorkingHr()
       AND Not Isconscious (Mary) THEN contact
       Dr Joe

    IF IsNextOfKin (Ms Helen, Mary) THEN contact
       Ms Helen

    ELSE IF IsDoctorFor (Dr Joe, Mary) AND
            Not WorkingHr()THEN contact Dr Joe
```

Figure 6. "if-then-else" expression derived from the preferences and choices in Figure 4 and 5.

Regarding our second proposal, we are designing and developing a user feedback system that enables users to modify or add new context, situation and preference information of context-aware applications to control their behaviour. In order for users to make changes to the information, users must first be presented with the information, and be able to understand and interpret it correctly. Thus, the research methods for the user feedback system will initially involve an investigation of how the middleware information can be displayed to the users in an intelligible manner. For example, textual explanations may be used for revealing situations, perceptual cues (i.e., visualisation) for exposing the current context model and its context facts, and a combination of both textual and graphical explanations can be used for representing preferences. Techniques for retrieving the information from middleware, generating explanations and presenting it to users are designed based on Hardian et al's approach [14].

After the investigation when users have access to the middleware information and capable of interpreting it accurately, we will then research how users can provide feedback and make changes to the information (i.e., modifying a particular preference, context fact or situation, and linking them with various existing adaptive actions using choice candidates) to control the application's adaptation. The research methods will involve designing a user friendly graphical tool that is intuitive for users to add/modify context facts and formulate situations with the context information displayed using the representation techniques derived from the initial investigation. We will also research a method for providing guidance to users, guiding them through the process of modifying or formulating new context facts and situations, and showing what the resulted adaptation will be after the modification they have made, so they can be assured that the final adapted application behaviour will meet their expectations. Furthermore, we will research a user model for the feedback system to take into account the user's level of technology expertises (e.g., amateur and experienced). Experienced users possess higher computer literacy than amateurs, and to an extend they are able to understand the languages used in working with the middleware models. They do not require detailed explanations and guidance, and should be allowed greater flexibility and options when modifying middleware information to control application behaviour.

3.3 Usability Study and Evaluation

Towards the final objective, we will evaluate the effectiveness of the proposed intelligible preference model and user feedback system by carrying out a user study to investigate whether the framework improves usability of context-aware applications. The first step is to develop two applications with similar functionalities, one will adopt the proposed preference model during the development, and the other one will adopt the classical model. Both applications will be placed under the same scenario where they do not behave as users expected and require overriding of adaptations. The user of the application that adopts the proposed intelligible model should experience fewer difficulties with minimal frustration in changing the application behaviours compare to the other application.

4. RESULTS AND CONCLUSION

Adaptivity in context-aware computing applications is a double-edged sword. Application adaptations will not always behave as users expected due to variability in human preferences and imperfect sensing of context information, etc. causing annoyance and hinder user objectives. Our research tackles this challenge by proposing an intelligible preference model and a user feedback system that enable users to understand the internal decision process for application adaptation, including explanations for context, preference and choice evaluations. The works also allow untrained users to modify situation and preference information

according to their individual requirements to gain control of application behaviour.

Results to date include a full analysis of Henricksen et al's middleware models [15], and a preliminary investigation and design of the intelligible preference model. Future work will involve carrying out proofs on any requirement discrepancies (i.e. expressiveness, conflict resolutions, etc., as mention in section 3.2) between the two models, designing and developing a user feedback system, and a usability evaluation of applications which will require users to override unwanted application behaviour using our proposed. We also aim to uncover some answers for the following questions. Will the attempt to ask users to modify preferences place an undue burden on users (i.e., will it hurt usability and performance unacceptably)? When does the cost outweigh the benefit, and how to strike such balance? Can user preferences be inferred or does it have to be explicitly provided, and what are the tradeoffs for both occasions?

5. ACKNOWLEDGMENTS

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program; and the Queensland Government.

6. REFERENCES

[1] Bettini,C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A. and Riboni, D., "A Survey of Context Modelling and Reasoning Techniques", *Pervasive and Mobile Computing*, Volume 6, 2010

[2] Indulska, J. and Robinson, R., "Modelling Weiser's "Sal" Scenario with CML", the *Sixth Workshop on Context Modelling and Reasoning* (CoMoRea'09) affiliated with the IEEE PerCom'09 conference, Galveston, US, March 2009, IEEE Press.

[3] Hu, P., Indulska, J. and Robinson, R., "An Autonomic Context Management System for Pervasive Computing", *Proc. of the IEEE International Conference on Pervasive Computing and Communications* (PerCom'08), Hong Kong, March 2008.

[4] Bellotti, V. and Edwards, K., "Intelligibility and accountability: Human considerations in context-aware systems", *Human-Computer Interaction*, 16 (2-4), 2001, pp.193-212.

[5] Lim, B. and Dey, A., "Assessing the impact of provision types and accuracy on intelligibility for context-aware applications", *CHI 2010*, ACM, Atlanta, Georgia, USA, 2010

[6] Lim, B. Y., Dey, A. and Avrahami, D. "Why and why not explanations improve the intelligibility of context-aware intelligent systems", *CHI 09: Proceedings of the 27th international conference on Human factors in computing systems*, 2119-2128, Boston, USA, 2009

[7] Chen, H., Finin, T. and Joshi, A. "Semantic Web in the Context Broker Architecture", In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, IEEE Computer Society, 2004

[8] Bardram, J., "The Java Context-Awareness Framework (JCAF) - A service infrastructure and programming framework for context-aware applications". *Pervasive 2004*, 98-115

[9] Caswell, D., and Debaty, P. "Creating Web representations for places". *HUC'00*, 114-126

[10] Dey, A.K., Abowd, G.D. and Salber, D., "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", *Human-Computer Interaction*, 16(2-4): 97-166, 2001.

[11] Indulska, J., Henricksen, K., Mcfadden, T. and Mascaro, P., "Towards a Common Context Model for Virtual Community Applications", *Proc. of 2nd Int. Conference on Smart Homes and Health Telematics (ICOST)*, Volume 14 of Assistive Technology Research Series. IOS Press, pp. 154-161, 2004

[12] Newberger, A. and Dey, A., "Designer support for context monitoring and control". *Technical Report IRB-TR-03-017*, Intel Research, June 15 2003

[13] Dey, Anind K. and Newberger, Alan, "Support for context-aware intelligibility and control", *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pp.859--868, Boston, USA, 2009.

[14] Hardian, B., Indulska, J. and Henricksen, K. "Exposing Contextual Information for Balancing Software Autonomy and User Control in Context-Aware Systems", in *Proc. of the Workshop on Context-Aware Pervasive Communities: Infrastructures, Services and Applications*, affiliated with the Pervasive'2008 conference, Sydney, May 2008.

[15] Henricksen, K. and Indulska, J. "Developing Context-Aware Pervasive Computing Applications: Models and Approach", *Pervasive and Mobile Computing*, Volume 2, 2005.

[16] Lim, B., Dey. A, "Assessing Demand for Intelligibility in Context-Aware Applications", *Ubicomp'09: the 11th international Conference on Ubiquitous Computing*, New York, pp. 195-204, 2009