A Framework to Support Intelligibility in Pervasive Applications

Johnson Fong, Jadwiga Indulska and Ricky Robinson
The University of Queensland
School of Information Technology and Electrical Engineering
National ICT Australia (NICTA)
Queensland Research Laboratory, Australia
{jfong, jaga}@itee.uq.edu.au, ricky.robinson@nicta.com.au

Abstract— Adaptations of context-aware applications do not always result in behaviours that users expect, due to imperfect sensing of context information and variability in human preferences, etc. This can negatively impact the user experience of applications and compromise the trust users have in them. In order to gain user acceptance it is critical for applications to support intelligibility, so they are capable of justifying their adaptive actions and explaining the decision process of adaptations to their users. Based on these intelligible explanations, users should be able to modify application settings/thresholds to correct any undesirable behaviour. This paper presents a model-based developmental framework that supports intelligibility and user control of context-aware applications. It identifies and exposes the internal middleware models which influence adaptation decisions, and facilitates generations of explanations regarding evaluations of the models. These middleware models include preference models defined using Defeasible Logic, situation abstractions specified using Hidden Markov Models and First Order Logic, and context models developed using Context Modelling Language. The framework also takes into account users' expertise in technology when providing explanations and control to application behaviours.

Keywords –context-aware; intelligibility; framework; infrastructure; context modelling

I. INTRODUCTION

Context-aware applications that do not behave as user expect can lead to loss of user trust, satisfaction and acceptance to these systems. Consider the following scenario:

"Mary is an elderly woman who has been diagnosed with epilepsy. She is currently living by herself in a smart home that provides context-aware services designed to assist older persons in maximizing independence and maintaining a high quality of life. One of its context-aware services provides medical assistance moments before and during an epileptic seizure [16]. Measuring heart rate variability, physical activity and brain signals, this service predicts seizures, warns her and forbids her from performing risky activities (e.g., taking a bath) moments in advance. It also contacts nearby relatives or healthcare professionals automatically during a seizure. Mary was satisfied with the service until recently she felt somewhat overwhelmed by the amount of warnings about forthcoming seizures. She is annoyed because the anticipations were mostly either false-positives (i.e., no occurrence of seizures after the warnings) or false-negatives (i.e., missed seizures).

Hence, this raised a few questions in her mind, such as why is it so sensitive sometimes and how certain are the predictions, etc. Apart from questioning the inference of the situation, she also has some opinions regarding her user preferences. She reckons that even though if she is about to have a seizure, she still doesn't want the service to alert her daughter unless the seizure has really occurred. However, since she does not know how the smart home works, she cannot change its behaviour. She ended up misusing it and abandoning the useful smart home services, simply because she no longer trusts it."

To mitigate these issues context-aware applications should be *intelligible*, i.e., able to reveal their inner workings to users and help them to understand how the applications operate, thereby increasing users' trust in them. In this paper, we describe an architecture of a framework that supports intelligibility of context-aware applications by identifying and exposing the internal middleware models which are used to arrive at adaptation decisions, and renders them understandable to non-technical users. The framework is an extension of the PACE middleware [18], which provides a rich and comprehensive set of programming and middleware models. The framework includes: user preference models defined using Defeasible Logic (DL) [9], an extension of situation abstractions to include Hidden Markov Models (HMM) in addition to a variant of First Order Logic (FOL) [2] used in PACE, and an extension to the Context Modelling Language (CML) [18]. The framework takes into account various levels of user expertise in technology when providing feedback/explanations regarding the application behaviours.

The remainder of the paper is organised as follows. Section 2 introduces DL user preferences and a method [5] adopted by the framework for explaining the reasoning outcomes. Section 3 presents a technique by [6, 19] for verbalising FOL situations. Section 4 and 5 describes an extension to the situation abstraction for classifications of HMMs and an algorithm from [7] for explaining situation inferences. Section 6 discusses an approach by [8] for verbalising CML context models. Section 7 identifies key requirements for the framework to support intelligibility, and provides an overview of its architecture including a user expertise model. Section 8 summaries the paper.

II. EXPLANATION OF DEFEASIBLE PREFERENCES

The preference model in PACE has been redesigned to support intelligibility. It employs DL [9] for modelling user preferences which capture requirements on how applications

978-1-4673-5077-8/13/$31.00 ©2013 IEEE

should behave in particular situations and are used in adaptation decisions, capturing requirements on how users would prefer the applications to behave under certain situations. DL is a computationally efficient rule based non-monotonic (sceptical) formalism for reasoning with incomplete and inconsistent information. Figure 1 illustrates a defeasible preference `EpilepticSeizure_Pref` based on the smart home scenario that demonstrates the use of a superiority relation. An evaluation of the preference can be initiated by PACE situation based triggering technique [2] upon a detection of a seizure.

```
EpilepticSeizure_Pref(Occupant):
r1: ⇒forbidRiskyActivities(Driving, Cooking, Bath)
r2: ⇒alert (CareCenter)
r3: Cooking()⇒TurnGas(Off)
r4: WaterTemp(Hot)⇒TurnGas(off)
r5: Minor_Seizure()⇒remindMedication(Epileptic)
r6: Minor_Seizure()⇒~alert (CareCenter)
r6>r2
```

Figure 1. An example of a defeasible preference with a superiority relation.

Overall the rules in the preference are somewhat self-explanatory: Defeasible rules r1 to r4 indicate that when the elderly is experiencing a seizure, care centre should be contacted and risky activities should be forbidden. If she is cooking or using any hot water, the gas should be turned off automatically. But r5 and r6 suggest that if it is a minor seizure (e.g., some eye fluttering or smelling odours that do not exist, etc.), then it is not necessary to alert care centre and just remind her about medication. Hence, a superiority relation is employed to allow the conclusion of r2 to be overridden by r6. The framework also facilitates automatic generation of explanations regarding evaluations of user preferences, by adopting a trace tree pruning method from Antoniou and Governatori [5] that extracts meaningful traces from the reasoning of defeasible preferences. The traces are then used to generate readable explanation phrases by retrieving descriptions of corresponding atoms and variables, and concatenating them with appropriate keywords using a string concatenation method.

Explanation Types	Explanations
What is it doing?	forbidRiskyActivities () and remindMedication() - **Forbid user from driving, cooking and taking bath until her state return to normal** and **remind user of epileptic medication**
Why?	forbidRiskyActivities () and remindMedication() - Because **Seizure is detected** and **Seizure is minor**.
Why Not perform?	
alert (CareCenter)	Do not alert Care Center - Although **Seizure is detected,** the action is blocked when **Seizure is minor**.
How To perform?	
TurnGas(Off)	Turn Off Gas – **User is cooking** or **Water temperature is too hot**
User Control	Novice user may edit the preference via the Feedback Interface. Advance user may modify …

Figure 2. Explanations generated from an evaluation of the preference

The aim is to generate some of the explanation types recommended by Lim and Dey [10] as shown in Figure 2, which explains to users the reasoning outcome of the preference `EpilepticSeizure_Pref` under particular situations. Details of the explanation generation mechanism and syntax for defining defeasible preferences are available in [3, 5].

III. VERBALISATION OF FIRST ORDER LOGIC SITUATIONS

Situations are abstractions of the events occurring in the real world and are derived from context information (facts) captured in context models. They can be employed together with the defeasible preferences for users to describe circumstances/ conditions to which an application should adapt. Situations are specified using a variant of FOL proposed by [2]. The section discusses a technique from Kaljurand and Fuch et al. [6, 19] to support FOL intelligibility by verbalising FOL situations into a Controlled Natural Language (CNL), known as Attempto Controlled English (ACE) at The University of Zurich [11]. It is a subset of standard English with a domain-specific vocabulary and a restricted grammar. The restriction is to reduce ambiguity and vagueness inherent in full natural language, so that it is easier to understand by non-technical users. The aim is to allow users to be able to scrutinise the situations and appreciate their meaning, so as to develop a mental model of the application that is crucial to its intelligibility. Figure 3 and Figure 4 illustrate an example mapping of FOL situations to ACE queries that conform to ACE grammar and syntax defined in [11].

```
GlycosuriaDetected (Person):
∃ person, space • LocatesIn [person, space] ∧
DetectsSubstance [space, unusualSubstance] ∧
space="Lavatory" ∧ unusualSubstance = "Glucose"

Is there a Person and a Lavatory such that the Person locatesIn
  the Lavatory, and the Lavatory detects some Glucose?
```

Figure 3. A mapping of a situation `GlycosuriaDetected` to an ACE query.

```
CanUseChannel(person, channel):
∀device•Requires[channel,device]•∃person•LocatesNear
 [person,device] ∧ PermittedToUse [person, device]

For every Device that a Channel Requires, is there a
 Person LocatesNear the Device, and PermittedToUse the
 Device?
```

Figure 4. A mapping of a FOL situation `CanUseChannel` to an ACE query

The technique for mapping the situations to ACE can be summarised as firstly, converting the FOL into its prenex normal form by equivalence transformations. ACE phrases are deterministically generated via Discourse Representation Structures (DRS) [12]. Thus, the second step is to map the normal form to DRS, which is a logical representation of the information in the phrases. DRS act as an interlingua between the FOL situations and the ACE phrases. Example of a DSR can be found in [12]. Then given a valid DSR representation, a FOL situation can be verbalised into ACE. The verbalisation begins with a depth-first traversal of a

DRS tree, which contains discourse referents of the domain (i.e., quantified variables representing the objects of a discourse), and conditions for the referents. Each referents and conditions are verbalised depending on their locations in the tree structure, whether the referents have already been verbalised before, and complexity of the conditions whether they are built from other DRS or simple logical atoms. Finally, in order to verbalise a situation with the correct grammar, a lexicon, such a WordNet from Princeton, is needed to guide the morphological analysis/synthesis of some FOL predicates and object variables.

IV. Inference of Hidden Markov Models

Various types of situations require different types of situation models and FOL used in PACE does not cater for all types of situations, particularly for recognitions of certain human activities. HMM is one of the most widely used approaches for activity recognitions (e.g., physical, domestic activity and gaze [13], etc.) in context-aware computing. Not only can it be used to estimate the parameters, decode sequences and perform classifications efficiently using various inference algorithms, but is also rich enough to handle a large number of real-world applications that take into account of sequential data. It inherently caters for dynamic sequencing, thus, supporting temporality.

This section describes an extension of PACE to support reasoning with HMM, firstly, by capturing HMM parameters ($a_{ij}, b_j(O)$, π_i) to handle the Decoding and Estimation problem. Parameters are stored as matrixes in a situation model. We illustrate the use of the situation model through an example based on the seizure detection scenario. Figure 5 shows a graphical view of the model capturing parameters of an HMM `SeizureDetectionHMM`.

The parameters are Baum-Welch estimated based on training observations, which are EEG signals extracted from epileptic patients [14, 20]. The parameters reveal some characteristics of the disease: The initial probabilities indicate that there are typically three states (namely the *interictal*, *preictal* and *seizure*) in the brain of an epileptic patient [16]. The transition probabilities suggest that the interictal state is the state in which the brain functions normally while the seizure state is the state in which the brain functions abnormally. The preictal state is a special state before seizure onset, and its presence and duration may indicate how far the brain is away from normality. Regarding the observations, each hidden state generates an observation that has two features, the Teager Energy and Frequency Range of EGG brain wave. The emission probabilities are represented as Gaussian Mixture Densities, where each mixture is a multivariate Gaussian (two-dimensional) with different mean vector and covariance matrix. Given the HMM situation model, epileptic states can be Viterbi decoded from a set of observation sequences using the learned parameters.

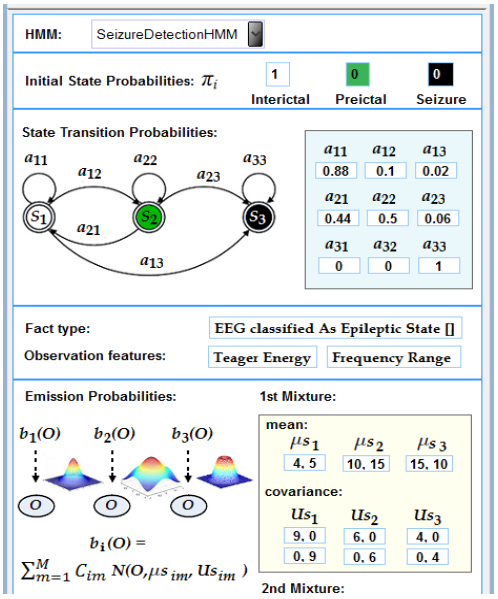


Figure 5. A situation model for `SeizureDetectionHMM`.

To maintain reasoning efficiency, the situation model also provides direct linkages to a fact type (`EEGclassifiedAsEpilepticState`) that holds the inferred states as well as the observations. Figure 6 gives an example of the fact type, which has specific timeliness constraints for representing the temporal aspects of HMM sequential data, and quality indicators for modelling certainty of inferred states and their duration. Using this information associated with inferred states, higher level situations can be derived by employing the existing FOL situation modelling approach, which has highly expressive primitives for evaluating and combining a range of abstracted contexts as illustrated in Figure 7. It shows a situation `CriticalCondition` comprising the state information represented in the HMM context fact and a FOL situation. It indicates that an occupant is in a critical condition when there is a seizure lasting over four minutes or the occupant is not conscious.

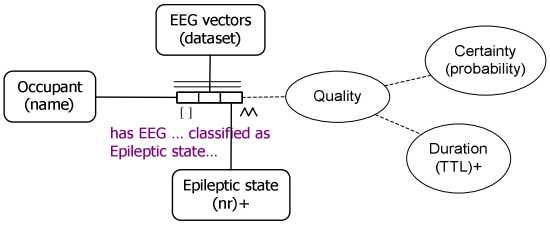


Figure 6. An HMM fact type capturing EEG vectors and decoded epileptic states

```
CriticalCondition(Occupant):
(∃ Duration, EEGvectors, EpilepticState :
eegClassifiedAsEpilepticStates[] ∧ EEGvectors = Dataset1 ∧
EpilepticState = 3 ∧ TTL > 4 mins) ⍰ ~conscious(Occupant)
```

Figure 7. Higher level situation comprised of HMM facts and a FOL situation.

V. EXPLANATION OF HMM INFERENCES

Having introduced the HMM situation model and described how inferred states can be used to form higher level situations, we are set to tackle the intelligibility issues where generating explanations of the inferred state is crucial for the user acceptance of applications. The framework adopts an approach described in Lim et al. [7] and Poulin et al. [21]. It calculates a sum of evidence that indicates how much support each feature of an observation contributes to an inference of a state, why the application did not infer the other state values and how certain the application is regarding the classification of a state. To illustrate how each of these explanation types is achieved, we present an example in Figure 8 that explains the states inferred by the `SeizureDetectionHMM` using EEG dataset from [14, 20].

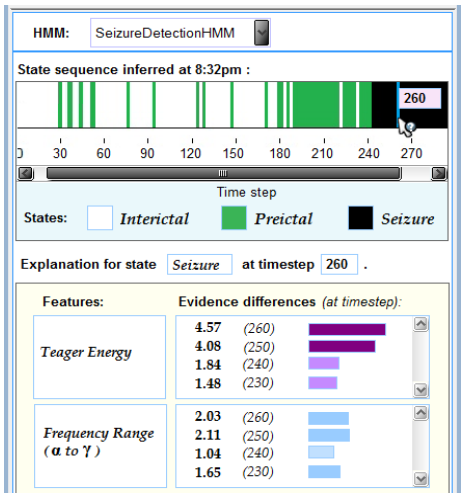


Figure 8. An Explanation of an HMM inference using evidences due to feature vectors.

The figure shows the inferred state sequence commenced with an interictal state and concluded with a seizure state. It enables users to understand *why* a *seizure* state is being inferred at time step 260, which is due to the strong evidence of Teager Energy. At time step 260, the evidence of Teager Energy is valued at 4.57, indicating that it is relatively *certain* regarding the current state being classified as *seizure* compared to the evidence value (1.84) 20 seconds before. Although the evidence due to the Frequency Range is relatively weak, together with the energy evidence it has sufficient confidence to infer the state at time step 260 is *seizure*. The explanations shown in Figure 8 are generated using the evidence due to the *feature vector* $f(Q^i, v, t)$. In order to obtain the evidence, we apply the following procedure [7]. Firstly, the parameters $\lambda = (a_{ij}, b_j(O), \pi_i)$ is retrieved from the learned HMM. That is, the transition probability $[a_{ij}] \equiv P(q_{t+1}=S_j | q_t=S_i)$, where q_t is the state of the model at time step t; an emission probability $[b_j(O)] \equiv P(O_t = v_m | q_t=S_j)$, where $\{ v_1 v_2 \ldots v_m \}$ is a finite set of values O_t can take on; and a set of initial state probability $[\pi_i] \equiv P(q_1=S_i)$. Secondly, given λ and a sequence of observation O, a state sequence $Q^i \in Q$ is inferred when it maximizes $P(O,Q^i | \lambda)$, which is equivalent to maximizing $P(Q^i|O, \lambda)$. Hence, the classifier can be expressed as:

$$P(O,Q^i|\lambda) \propto P(q_1^i) \prod_{t=2}^{T} P(q_t^i|q_{t-1}^i) \prod_{t=1}^{T} P(O_t|q_t^i)$$

Thirdly, when Q^i is classified over other state sequences $X = \{Q^k\}_{k=1}^{K}$, we can represent it as an equation $P(O,Q^i|\lambda) > P(O,Q^k|\lambda)$. Then take a logarithmic of the equation so as to express the classifier as a discriminant:

$$\text{HMM classification of } Q = \begin{Bmatrix} Q^i & log\left(\frac{P(O,Q^i|\lambda)}{P(O,Q^k|\lambda)}\right) > 0 \\ Q^k & otherwise \end{Bmatrix}$$

Then, we multiply the formula with N^T permutations, where N = number of possible states at time t, and T = number of time steps in a state sequence. This allows us to yield the discriminant of the classifier's formulation:

$$d = log\left(\frac{\prod_{k=1}^{N^T} P(O,Q^i|\lambda)}{\prod_{k=1}^{N^T} P(O,Q^k|\lambda)}\right) > 0$$

With some working (proof elaborated in [15]), we can derive the ensuing expression from the discriminant of the classifier:

$d(Q^i, v, t) = s + (\sum_{t=2}^{T} u) + (\sum_{t=1}^{T} \sum_{v=1}^{n} f)$ *where:*

$$s(Q^i) = N^T \, log \, P(q_1^i) - \left[\sum_{k=1}^{N} log \, P(q_1^k)\right]^T,$$

$$u(Q^i, t) = N^T \, log \, P(q_t^i|q_{t-1}^i) - \left[\sum_{k=1}^{N^T} log \, P(q_t^k|q_{t-1}^k)\right],$$

$$f(Q^i, v, t) = N^T \, log \, P(O_{t_v}|q_t^i) - \left[\sum_{k=1}^{N^T} log \, P(O_t|q_t^k)\right].$$

Finally, this gives us three weights of evidence:

$s(Q^i)$ – evidence due to initial probabilities of a selected state.

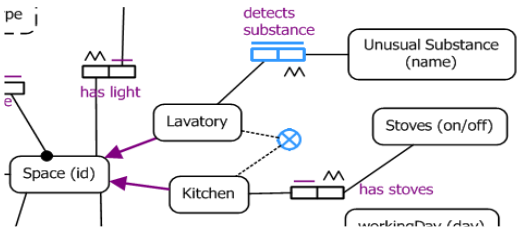
$u(Q^i, t)$- evidence due to transitions between states at time step t.

$f(Q^i, v, t)$– evidence due to feature vectors of an observation at time step t.

The evidence $f(Q^i, v, t)$ due to features vectors are employed to create the explanation interface shown in Figure 8. Utilising these evidences the framework can produce some other explanation types, e.g., what, why, why not and certainty, etc. as recommended by [10].

VI. VERBALSIATION OF CML CONTEXT MODELS

Context models capture low-level context information gathered from sensors and abstract it into context facts that are required by applications for deriving high level situations. This section presents the verbalisation algorithm from [8], which the framework utilises to explain CML context models to provide intelligibility. It is an extension to the Natural-language Information Analysis Method (NIAM), capable of verbalising the model's fact types, constraints, and derivation rules into plain English that is easily understood by ordinary users. The aim is to enable users to modify the models and introduce facts for new situations via a context modelling tool [4]. Figure 9 shows a portion of a CML model automatically verbalised to assist users in understanding and interpreting the model.



Verbalisation of the *Binary Uniqueness* constraint:
Lavatory detects Unusual substance is a sensed fact type.
It is possible that more than one Lavatory detects the same Unusual Substance and that same Lavatory detects more than one Unusual Substance.
Verbalisation of the *Exclusion* constraint:
No Space is both Kitchen and Lavatory. For each Space, at most one of the following holds: that Space is a Kitchen ; that Space is a Lavatory.

Figure 9. Verbalisation of selected constraints

In order to generate verbalisations for CML context models, the framework utilised the following approach [8]. Firstly, verbalisation is generated based on constraints enforced on a selected context fact. Each constraint is represented by an interface, which is an XML/XSLT-based document that offers a formal and unambiguous definition of how exactly the constraint is to be verbalised. The interface is implemented automatically on-the-fly with snippets (i.e., dynamic data relating to the constraint) and variables defined on a CML model, precisely capturing complex patterns of a verbalization specification for an individual constraint. Given the patterns and snippets represented in the XML interface, they are translated into codes for generating verbalisation phrases using a field replacement approach, then all phrase delimiters are applied such as capitalisation and punctuation of sentences, add lines between selected snippet, and indent phrases on aggregated snippets.

VII. Framework Architecture

There are certain requirements that a framework needs to satisfy in order to provide intelligibility and control of application behaviours. This section presents key functional requirements for the framework to support application intelligibility followed by an architecture of the framework.

Req 1: Externalise situations and preferences evaluations.

One of the most fundamental requirements of any pervasive infrastructure is to allow decoupling of context-aware functionalities (in the form of the middleware models) from the remainder of the application logic. The aim is not only to ease software engineering of applications by reducing their complexity, but it is also critical for supporting intelligibility of applications. By externalising the models into the middleware, the models can be made transparent and modifiable by users.

Req 2: Facilitate generation of appropriate explanations.

Users may not be interested in all the explanations an application can produce as some maybe more useful than the others [1]. Hence, the framework is required to support generation of explanation types that are of users' interest, such as those listed in [10] and allow users to selectively choose which of the types they prefer to receive, and when.

Req 3: Account for various levels of users' expertise.

Users can have varying levels of expertise in technology and experiences to a system. This can affect their understanding of adaptation decisions. Thus, the framework should allow applications to vary the amount/nature of explanations and control provided to users with respect to users' requirements and capabilities.

Req 4: Maintain transparency in generating explanation.

The framework should not require developers to understand the algorithms for generating explanations for each of the middleware models. Algorithms are encapsulated and can be accessed via a set of programming interfaces.

Figure 10 provides an overview of the framework, where the coloured components show an extension of the PACE middleware [18] to support intelligibility and control of application behaviours. The decision process of an adaptation involves reasoning of user preferences and situations (FOL and/or HMM), which is handled by the Adaptation Layer.

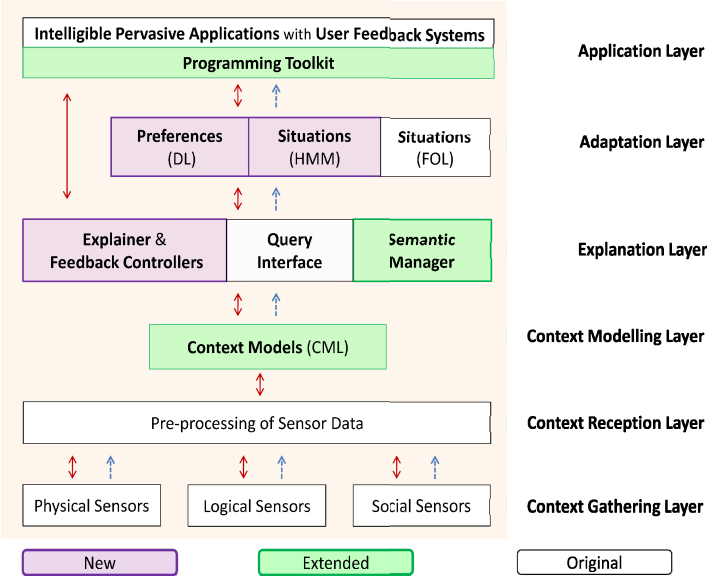


Figure 10. Simplified view of the framework and interactions among components.

The evaluations can either be initiated by a trigger [2] as the result of monitoring changes of context information for particular situations, or initiated directly by application requests. During each evaluation, the reasoning traces are

recorded and stored in a trace database at the Explanation Layer. These traces contain information required by the Explanation Generators (EG) to generate explanations. Such information includes unique ID, the outcomes of the evaluations, timestamps, proof schemas and evidence calculated, etc. In order to generate explanations, the EG also requires additional information obtained from a Semantic Manager, which holds meta-data for the middleware models and various adaptations. The Semantic Manager also stores a User Model that is designed to be adapted by developers to create feedback systems that account for various levels of users' expertise. The model defines three skill levels (novice, advance and expert) in which a user can interact with applications. When a novice user inquires about a particular adaptation, explanations regarding evaluations of the corresponding preferences are generated, since they are the easiest to understand and modify. Advance users are permitted to drill down on situations associated with the preferences and are provided with explanations about their inferences together with write access to their specifications. Expert users are allowed to view and edit the context model that captures context facts required by the situations. They can learn about the current state of these facts and their verbalisations. The user model is described in more details in [17]. Its aim is to avoid overexposing application information and overwhelming users with unfamiliar system features, which can cause confusions to users. Had users made any modification to the middleware models via the feedback systems, Feedback Controllers (FC) are responsible for mapping the changes back into the corresponding models. There is a dedicated EG and FC for each middleware model. From the perspective of application developers, by employing the API of the EG and FC, the developers can create user feedback systems without having detail knowledge of how explanations are generated, or how mapping algorithms work for mapping users' modifications back to the corresponding models.

VIII. CONCLUSION

We have presented a framework for developing context-aware applications that support intelligibility and user control. The framework adopts a layered approach based on loosely coupling components and middleware models. It allows users to scrutinise the models (DL preferences, HMM and FOL situations, and CML context models), request explanations regarding their evaluations and apply appropriate feedback to the models to control application behaviours according to users' capabilities and expertise. A user study is under development to investigate in what capacity do intelligible applications (created using the framework) improve user experiences, and identify potential issues and opportunities for future advancement of the framework.

ACKNOWLEDGMENT

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.

REFERENCES

[1] B. Lim and K. Dey, "Investigating intelligibility for uncertain context-aware applications", *UbiComp '11*, pp. 415-424, Beijing, China.

[2] K. Henricksen, J. Indulska, A. Rakotonirainy. "Using context and preferences to implement self-adapting pervasive computing applications", *Journal of SPE,* Vol. 36, pp. 1307-1330, 2006.

[3] J. Fong, H.P. Lam, R. Robinson and J. Indulska, "Defeasible preferences for intelligible pervasive applications to enhance eldercare", *PerCom Workshops – CoMoRea'12*, Lugano, Switzerland, 2012.

[4] J. Fong, J. Indulska, R. Robinson, "Tool Support for Designing CML Based Context Models in Pervasive Computing", *ICPS Workshops '10*, Berlin, 2010

[5] G. Antoniou, A. Bikakis, G. Governatori , "Proof explanation for a nonmonotonic Semantic Web rules language", *DKE*, vol. 64, 662-687, 2008.

[6] K. Kaljurand, "Paraphrasing Controlled English Texts", *Controlled natural Language (CNL 2009),* Marettimo Island, Italy

[7] B. Lim, and A. Dey, "Toolkit to Support Intelligibility in Context-Aware Applications", *Ubicomp '10*, Copenhagen, 2010

[8] T. Halpin and M. Curland, "Automated Verbalization for ORM 2", *OTM 2006 Workshops*, Monpellier, France.

[9] D. Nute, "Defeasible logic: theory, implementation, and applications", *Proceedings of INAP 2001*, Japan

[10] B. Lim, A. Dey, "Assessing Demand for Intelligibility in Context-Aware Applications", *Ubicomp '09,* NY

[11] N. Fuchs, K. Kaljurand, T. Kuhn., "Attempto Controlled English for Knowledge Representation", *Reasoning Web*, Italy, LNCS 5224, 2008.

[12] N. Fuchs, K. Kaljurand, T. Kuhn. "Discourse Representation Structures for ACE 6.6", *Technical Report ifi-2010.0010*, UZH, 2010.

[13] A. Bulling, J. A. Ward, "Robust Recognition of Reading Activity in Transit Using Wearable Electrooculography". *Pervasive '08,* 19-37, Sydney

[14] EEGData, *www.meb.uni-bonn.de/epileptologie/science/physik/*, last accessed 1 Nov, 2012.

[15] J. Fong, Proof for Deriving Evidence, *www.paceware.org*, last accessed 1 Nov 2012

[16] Wong, "A stochastic framework for evaluating seizure prediction algorithms using hidden markov models", *Journal of Neurophysiology*, vol 97, 2007.

[17] J. Fong, J. Indulska, R. Robinson, "A Preference Modelling Approach to Support Intelligibility in Pervasive Applications", *PerCom Workshops CoMoRea'11*, Seattle, USA, 2011

[18] K. Henricksen and J. Indulska, "Developing Context-Aware Pervasive Computing Applications: Models and Approach", *PMC*, Volume 2, 2006.

[19] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. "Verbalising Formal Languages in Attempto Controlled English I". *Technical report Deliverable I2-D5.*, REWERSE, 21 pages, 2005.

[20] D. Zhu, *Time-frequency and Hidden Markov Model Methods for Epileptic Seizure Detection*, MSc Thesis, University of Cincinnati, 2009

[21] B. Poulin, R. Roman, D. Szafron , P. Lu , R. Greiner , D. S. Wishart , A. Fyshe , O. Pearcy , C. Macdonell "Visual Explanation of Evidence in Additive Classifiers",*Proceedings of the 18th conference on Innovative applications of artificial intelligence IAAI'06*, Vol 2, 2006.