Composition of Reusable Higher-level Contexts

Waltenegus Dargie, Tino Löffler, Olaf Droegehorn, Klaus David

Abstract— **Capturing contextual information, especially higher-level contexts enables systems to understand and predict the behaviour of a mobile user. This kind of information is mostly implicit contexts, which abstract a complex state of a situation and can only partly be captured by employing sensors. Higher-level context has the potential to make user-application interaction richer, simpler, and more intuitive. However, composing higher-level contexts from explicit, atomic contexts requires complex and painstaking reasoning procedures to resolve uncertainty due to inconsistent sensor readings and incomplete information. In this paper, we introduce a new approach that assists application developers to take higher-level contexts into account without the need to know the details of atomic contexts. To demonstrate our approach, we will introduce the Context-Aware E-Pad (CAEP) we have designed and implemented. CAEP "observes" the user making a decision and associates with it a set of atomic contexts. Likewise, correlated decisions are mapped to similar associations. By repeatedly "observing" the user making decisions, CAEP can predict user's wishes and behaviours.**

Index of terms—association, atomic contexts, decision, higher-level contexts, implicit contexts, lower-lever contexts

1. INTRODUCTION

Typical applications centre the user's attention to the computing task, which needs to be performed to fulfil the user's request [1]. These applications oblige the user to provide all input information pertaining to a task an application should accomplish. As far as the computing task is the only task, to which the user pays attention, this might not be a problem; but typically user has to divide his attention between other activities such as driving, talking to other people, holding a presentation, etc., while a computing task is in progress. In this case, these kinds of applications introduce enormous distraction instead of assisting a user to solve a specific problem.

Furthermore, development in integrated circuit technology has enabled the production and deployment of computing devices at an ever falling price. Whereas once it has been considered as a revolution to provide individuals with personal computers, it is now an everyday practice to carry with us mobile computers and communicate with them on the spur of the moment. The implication is twofold: on one hand, it has become commonplace for a user to own multiple devices; on the other hand, these devices as well as the users are no longer stationary. In consequence of this, resources have become pervasive and a mobile user can access and process information virtually from anywhere at anytime.

Therefore, mobile applications and services must evolve to make the needed computing task less obtrusive. In other words, the amount of explicit input a user has to supply for a computing task must be reduced.

It is widely acknowledged by researchers in the mobile field that context-aware computing can meet this desire to reduce explicit input. By enriching applications with implicit context information, we could increase their awareness of people, places, and computing devices. This awareness further leads to the execution of useful services with little or no involvement from the user.

Consider a business man driving inside a foreign city searching for a parking lot, expecting at the same time an important business related call. But his mobile phone, which is inside his jacket, is at the backseat, still on vibration mode because of a meeting he has been attending.

A context-aware application can assist such a desperate driver in a number of ways. To start with, it could automatically switch the phone to a ring mode, as its master could not detect the arrival of a call while driving; furthermore, the phone can autonomously discover resources inside the car, such as a loudspeaker and a microphone, to make conversation on the phone less obtrusive.

From this scenario, we can list four important aspects of context-aware computing:

A. Implicit understanding of user's situation

Context-aware applications, assisted by numerous sensors and other context data sources, can build an understanding of a user's situation. This greatly enhances the inherent input deficiency traditional mobile applications suffer [1] from. In our brief scenario above, this happens, if the mobile phone understands that the businessman is *driving.*

B. Reduction of user side input

Success in capturing the context of a user brings about the invocation of services suitable to the context. As a direct consequence of *A,* a context-aware application reduces the businessman's involvement by switching the mobile phone from *vibration mode* to *ring mode*.

C. Adaptation

Adaptation occurs in two ways: firstly, the mobile phone's ring mode is adjusted to the driving context; secondly, it adapts to available resources inside the car. The latter occurs

Waltenegus Dargie, Tino Loeffler, Olaf Droegehorn, Klaus David
Chair of Communication Technology (ComTec), University of Kassel,
Wilhelmshoeher Allee 73,
34121, Kassel, Germany
Phone: +49 561 6304; Fax: +49 561 6360.
Email: {Droegehorn , David}@ uni-kassel.de

when the mobile phone splits incoming voice to an external speaker, while a microphone is used to stream in voice from the user.

In general, in pervasive computing environments, due to the mobility of both users and devices in time and space, the states of resources available may potentially change. In all these circumstances, context-aware applications attempt to adapt their behaviour to the ever changing context of a user, the devices he uses, the co-located people around, and the physical environment.

D. *Provision of useful Services*

Depending on the context of the businessman, the mobile phone may prioritise incoming phone calls. This implies that context-aware applications are sensitive to change in preference of the mobile user and know which type of services to provide and which not to provide [2].

Having said all this, designing and implementing context-aware applications, which exploit implicit context information and dynamically available resources, is not an easy task. In this paper, we will discuss some outstanding issues context-aware application developers face and show how we have addressed a few of them.

The rest of this paper is organised as follows: in section 2, we will discuss some of the challenges in designing context-aware applications; in section 3, we will discuss related works; in section 4, we will introduce the Context-Aware E-Pad (CAEP) and discuss its typical features; and finally, in section 5, we will give a brief conclusion.

2. Challenges in Designing Context-Aware Applications

Human-human interaction, besides the expressiveness and the richness of the language used, involves implicit understanding of the context that encompasses the subject of interaction. This is due to the fact that humans are fit with powerful sensing organs which enable them to perceive the physical world appropriately. Here we are not merely referring to the common catalogue of five senses – light, hearing, touch, taste, and smell. These hardly cover all that is taking place. There are other vital organs which sense muscle tension and pressure on joints and tendons; the human brain knows intuitively the tilt of a head, the bend of an elbow, the position of a foot; below the conscious level, automatic systems adjust the chemical components of blood, control air pressure in the lung and blood pressure in the arteries, and monitor organ stretch receptors [3].

Computers are relatively good to measure and report their internal states as well as the internal states of other computers. They are, however, greatly limited to measure and capture the context of its users as well as the physical environment in which they operate in. Employing sensors may augment their measuring capacity and may enhance their awareness of the external world. But awareness is more than just measuring. Awareness means making sense of the data measured.

Our brain presents the world to us not as a collection of raw data, but wholly, conceptually, and meaningfully [3]. Likewise, computers must translate the huge amount of raw data that is collected from sensors and other context data sources in a way that is meaningful to a human user. Unfortunately, the richness and complexity of the world in which a mobile user and his devices operate make this process extremely overwhelming. Creating software or i.e. a system to capture a context, translate it to a desirable format, perform various manipulations and comparisons on it, and present it to the user in a meaningful way, is a difficult and time consuming software development task [4]. In addition, the challenge to potentially utilize a variety of hardware must be addressed. Besides, since context information can be gathered in a variety of ways, undoubtedly semantic conflicts may occur due to heterogeneity of context data sources. There are two types of semantic conflicts: representational and ontological. Representational conflict occurs when two context data sources use different terms to refer to the same context; ontological conflict occurs when two context data sources use the same term to refer to two different contexts. Therefore, resolving such plethora of semantic conflicts in order to enable interoperability is a significant challenge.

Furthermore, we cannot capture all types of contexts by employing sensors. There are higher-level contexts, which abstract complex states of a situation. They are implicit by nature and focus on the essential aspects of an entity and ignore or conceal less-important or non-essential aspects. Higher-level contexts are composed of numerous atomic contexts[1] and involve tedious reasoning procedures to resolve uncertainty due to inconsistent sensor readings and missing information. Apart from the reasoning task, identifying every relevant atomic context that has the ability to describe some potential properties of the higher-level contexts has to be addressed by the application developer.

In the next section we will discuss related work before we introduce our contribution.

II. Related Work

Cohen et al [5] defined and implemented a nonprocedural programming language called *iQL* for higher-level context composition. The language assists application developers to identify and bind heterogeneous data sources. An *iQL* programmer expresses requirements for data sources instead of identifying specific sources; a runtime system discovers appropriate data sources, binds to them, and rebinds when properties of data sources change.

Dey [1] proposes aggregators in his architecture for context-aware computing. Aggregators collect all relevant low-level contexts from the surrounding environment to characterise the situation of an entity as wholly as possible, where an entity can be a person, a device, or a place. Dey

[1] We use atomic contexts and lower-level contexts interchangeably; in both cases we mean context types that can directly be captured by employing sensors.

identifies three essential low-level contexts: identity, location, and activity. So an aggregator searches context data sources that provide these low-level contexts. However, aggregators perform no actual higher-level context composition. It is up to the application to make sense of the set of lower-level contexts.

Korpipää et al [6] propose a *context recognition service* that is capable of composing higher-level contexts. It uses a naïve Bayesian classifier which reasons about a higher-level context by aggregating numerous simple context atoms. The context atoms are selected based on their ability to describe some potential properties of the higher-level context. Other criteria include feasibility of measuring or recognising the context chosen as accurately and unambiguously as possible. Among its most important tasks, the framework manages uncertainty of sensor readings through the use of probability based inference and fuzzy membership.

Using the framework, a mobile device recognises whether it is outdoors or indoors; whether a sound is from a car, a water tap, rock music, classical music, an elevator, a speech, or from some other source. For the first two higher-level contexts, namely, indoors and outdoors, 14 atomic contexts describing the environment's light, humidity, and temperature were used, while 47 audio-based atomic context atoms were used to describe the remaining seven higher-level contexts.

The approaches above identify and describe at design time the type of atomic contexts required for composition. In *iQL* they were described as requirements; *aggregators* receive from the application a description of the type of context sources it binds; the naïve Bayesian classifier of *context recognition service* requires an ontological description of the atomic contexts together with a vector of context atom confidence values.

To decide atomic contexts at design time limits a composition assignment to those context atoms. Besides, application developers are forced to deal with lower-level context details, and to determine the way these atomic contexts should be organised and reasoned about.

We strive to free application developers from setting requirements for higher-level context compositions. Towards this end, the behaviour of the user plays a significant role. We associate the decisions of a user with a set of atomic contexts that have indirect influence on the decisions. Furthermore, related decisions are organised in such a way that an implicit, higher-level context can be inferred from them.

3. CONTEXT-AWARE E-PAD (CAEP)

CAEP is a word processor and exhibits similar characteristics to that of other word processors: a new pad can be created; existing pads can be loaded, edited, deleted and exchanged with different participation levels. Participation levels determine, with which access rights a pad is dispatched to recipients and includes editing and saving rights.

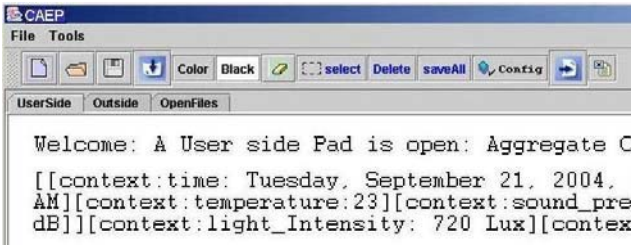

Figure 1: A snapshot of CAEP

Unlike many word processors, however, CAEP is context-aware in that it reacts to the context of a user to undertake a word processing task less obtrusively.

The essential features of CAEP are summarised as follows:

- Whenever a user interacts with it, it associates the interaction (in the form of decisions) with the situation in which the interaction takes place. This situation is characterised by a set of atomic contexts which can directly be determined.
- From the association, it generates a user profile or a subpart thereof to reason about higher-level contexts in terms of correlated decisions.
- From repeated observation and an aggregate of associations, it learns the behaviour of its user in order to provide useful services in a proactive manner.
- Moreover, the user's profile is useful for a semantic-based interaction between a user and the application. For instance, for people who have to deal with many files per day, remembering a file's name is a nightmare. Using CAEP, accessing a file is fairly simple: all the user has to remember is a situation (context) that is related to a file. Internally, CAEP maps the context to a file name and helps the user narrow his searching space.

In the two subsections following we discuss the features above in more detail[2].

A. Reasoning about higher-level contexts in terms of user's decisions

One common feature shared by all context-aware applications is that they react to the context of the user. Here we use context loosely. It may apply to the state of devices, networks, places, as well as co-located people. In most context-aware applications, this reaction to contexts entails one or more event-condition-action (ECA) rules. An ECA rule consists, primarily, an event (context), a condition (context parameter), and an action.

We present a model of the introspected world by a set of well-formulated formulas of the first-order logic. Hence a typical ECA rule may look like:

[2] Due to space limitation, we do not discuss semantic-based interaction in this paper.

$$(\forall i)(\exists j)(\exists k)(\exists l)(\exists m)\big((temperature(i)\wedge(i<j))\supset((heater(k))\wedge(l\le k\le l))\big) \quad (1)$$

Rule (1) defines two predicates: *temperature* and *heater* to state a general rule that if the value of a temperature measurement descends below a certain threshold, a heater should be switched on.

Now let us consider another formula as given by rule (2):

$$(\exists y)(\exists x)\big(lecture(y)\supset(pad(x)\wedge loaded(x))\big) \quad (2)$$

In rule (2), we want to load a pad for the user whenever he attends lecture y. In this case, the predicate *lecture* is a higher-level context, which is an abstraction of a complex situation. Since by employing sensor information it cannot directly be concluded: lecture y, our approach to solve this, is by using a mechanism to reason about this implicit context using a set of atomic contexts.

A simple inference rule may comprise a formula such as:

$$(\forall r)(\forall h)(\forall i)(\exists j)(\exists k)(\exists l)(\exists m)(\exists n)\begin{pmatrix} room(r)\wedge temperature(r,h)\wedge \\ relative\_humidity(r,i)\wedge \\ ((j\le h\le k)\wedge(l\le i\le m) \\ \supset lecture(n) \end{pmatrix} \quad (3)$$

Defining our "world model" using the formulas and rule (3) at design time has two drawbacks. Firstly, the application is bound to react only to the context types: *temperature* and *humidity,* thereby missing the chance to describe *lecture* in terms of other contexts. We do not mean, of course, that these two atomic contexts alone can describe the context in a lecture room. We rather mean that other context types may as well describe a lecture session instead of these two context types. Besides, here the availability of context data sources (or sensors) that deliver these context types is already assumed. Secondly, all the existential quantifiers must be predetermined either by the application developer or by the user himself.

CAEP's design concept frees both the application developer and the user from describing a higher-level context in terms of lower-level contexts. No specific rules are required at the side of the application at design time. Instead, the application "observes" for a set time as the user makes decisions[3] to associate every decision with a set of contexts that are acquired at the time the decision is made. Note that a decision is not associated with a specific set of atomic contexts. Decisions are the basic elements with which useful services are executed. Each decision corresponds to an action routine whose execution causes CAEP to take certain actions, transforming one world model to some other world model.

[3] Decisions in this context are loading, editing, creating, sending, and receiving a pad. External decision histories such as switching a mobile phone from a ring mode to a vibration mode are stored as the user's profile and are accessible to CAEP.

The observation time can be either an absolute temporal event or a relative temporal event. An absolute observation time corresponds to a unique time span on the time line with a clearly defined reference time and an offset time. A relative observation time corresponds to a unique time span on the time line, but in this case the reference event can be other than a temporal event. A relative observation time may be specified using any one of the decisions we mentioned earlier.

During an observation time, CAEP associates a set of context atoms with every decision the mobile user makes. Our goal is to express a decision in terms of the situation this set of context atoms characterises. Hence, the set of context atoms should represent the situation as accurately as possible. To minimise erroneous conclusions about a situation, CAEP searches at runtime and binds to heterogeneous aggregators, since heterogeneous aggregators gather and aggregate context data from sources which are spatially and temporally diversified.

Once the observation time is over, context types, which appear in the association repeatedly and which maintain predictable characteristics both with respect to their previous values as well as with respect to other context types, are taken as representative contexts to express a decision in terms of a complex situation.

We will elaborate this by an example. Suppose we want CAEP to reason about its whereabouts inside a university campus in the absence of a GPS receiver or an indoor localisation system. Likely places are: lecture rooms, conference halls, offices, corridors, a library, a cafeteria, a gym, an amphitheatre, an outdoor tennis court, and so on. CAEP should gather and processes environmental contexts such as temperature, light intensity, sound pressure, humidity, etc., and together with empirical and heuristic context knowledge, reason about particular places. Moreover, we want CAEP to learn the user's behaviour of loading a pad whenever he attends a particular lecture.

Since the user does not set a specific rule, there is no need to know a priori, which context types should best describe the loading situation. Therefore, during the observation time, CAEP listens to a loading decision. When a decision occurs, it binds to available heterogeneous aggregators to gather context data characterising the situation. Equation (4) represents the semantics by which a decision is associated with a set of context atoms

$$decision\big((pad(pad\_x)\wedge loaded(pad\_x))\big)\,when$$
$$context\begin{pmatrix} light\_Intensity(720\,Lux)\wedge sound\_pressure(13\,bB)\wedge \\ relative\_humidity(40\%)\wedge temperature(23\,^{o}C) \end{pmatrix} \quad (4)$$

One reason to remain flexible in the use of atomic context sources is that since we expect resources to be pervasive, their states may potentially change over time. Context data sources may come, move, or their performance may deteriorate due to aging of sensors, depletion of battery power, channel dynamics, etc. Consequently, two associations may

TABLE I
DECISION-CONTEXT ASSOCIATION FOR LOADING A SPECIFIC E-PAD

Date	A set of low-level contexts
1	[[context:time: Tuesday, September 21, 2004, 9:57:23 AM][context:temperature:23][context:sound_pressure:3 dB]][context:light_Intensity: 720 Lux][context:RH:50%]]
2	[[context:time: Tuesday, October 1, 2004, 10:10:23 AM] [context:Sound_pressure:6 dB][context:temperature:23] [context:Light_Intensity:9000 Lux][context:RH:20%]]
3	[[context:time: Tuesday, October 5, 2004, 10:10:23 AM][context:temperature:22][context: sound_pressure :3.86 dB]][context:light_Intensity: 700 Lux] context:RH:45%]
4	[[context:time: Tuesday, October 12, 2004, 10:00:07 AM][context:temperature:23][context: sound_pressure :3.98 dB]][context:light_Intensity: 716 Lux] context:RH:48%]

RH = relative humidity; temperature is measured in degree centigrade. 20 micropascal is used as a reference to measure sound pressure.

incorporate different sets of atomic contexts.

Once representative context atoms are identified and associated with a decision, the result is saved as a loading profile or a subpart thereof so that it can be accessible to other applications as well. This rule-based profile is useful to associate additional correlated decisions to reason about more complex situations.

Multiple correlated decisions increase CAEP's capability to learn the behaviour of a mobile user. For instance, suppose the user switches his mobile phone from a ring mode to a vibration mode while he attends *lecture y,* at which time the decision to load *pad x* was also made. Assuming that the user habitually switches his mobile phone to vibration mode whenever he attends a lecture, this *decision* also entails similar associations during the observation time. Since both *decisions* are related decisions, internally they will be mapped to the same sets of contexts. So a central profile administrator merges the two profiles as shown in equation 5. Table 1 displays four associations for a *loading* decision during a one month observation time.

$$\begin{aligned} &decision\left(\left(pad\left(pad\_x\right)\wedge loaded\left(pad\_x\right)\right)\right)\ followedBy \\ &\left(phone\left(phone\_y\right)\wedge ring\_status\left(phone\_y, viberation\right)\right) when \\ &context\begin{pmatrix} light\_Intensity\left(720\,Lux\right)\wedge sound\_pressure\left(13\,bB\right)\wedge \\ relative\_humidity\left(40\%\right)\wedge temperature\left(23\,^{o}C\right) \end{pmatrix} \end{aligned} \quad (5)$$

Because of a *decision* profile, both the application developer and the user are now shielded from the concern of lower-level context details. This is summarised by equation (6). As can be seen, the existential quantifiers are illuminated from the equation.

$$(\forall r)(\forall h)(\forall i)(\forall j)(\exists k))\begin{pmatrix} room\left(r\right)\wedge temperature\left(r,h\right)\wedge relative\_humidity\left(r,i\right)\wedge \\ sound\_pressure\left(r,j\right)\wedge light\_Intensity\left(r,k\right)\wedge \\ ((22^{o}C\le h\le 23.6^{o}C)\wedge(30\le i\le 50\%)\wedge \\ (0\le j\le 15dB)\wedge(100\le k\le 1000\,Lux) \\ \supset lecture\left(lecture\_y\right) \end{pmatrix} \quad 6a)$$

$$(\exists v)(\exists x)(\exists y)(\exists z)\begin{pmatrix} pad\left(x\right)\wedge loaded\left(x\right)\wedge phone\left(y\right)\wedge ring\_status\left(y,z\right)\supset \\ lecture\left(v\right) \end{pmatrix} \quad (6b)$$

B. Learning user's behaviour

Contexts which appear often in associations and maintain relatively deterministic characteristics contribute significantly to learn the behaviour of the user; contexts which appear either infrequently or which maintain nondeterministic characteristics despite their frequent appearance, contribute little to learn the behaviour of the user. Even those contexts which exhibit deterministic characteristics are subject to uncertainty. We distinguish two types of uncertainties: uncertainty due to the inherent limitation of sensing elements and uncertainty due to the unpredictable nature of the user. We cannot control uncertainties due to sensing elements; but uncertainties due to the user's unpredictable characteristics are studied in view of the aggregate associations and with respect to the higher-level context we are interested. For example, a user may not arrive at a lecture exactly on a set time; he may be sometimes late and sometimes early. Nevertheless, he may not be earlier or later than the duration of a lecture.

Temporal contexts are very helpful to study habitual actions. If there is a continuously increasing time context associated with a user's decision, and if this continuity exhibits some deterministic pattern, CAEP attempts to infer a habit using equation (5).

To determine time pattern in the user's *loading* behaviour, CAEP decomposes the temporal context in to two parts: time and day. As can be seen from the table, the second loading decision occurred 10 days after the first decision; the third decision occurred four days after the second and fourteen days after the first; the fourth decision occurred 7 days after the third, 11 days after the second, and 21 days after the first. Therefore, CAEP calculates a mean interval by considering two decisions as related decisions.

Therefore:

$$l_i = \frac{\sum_{j=1}^{m}\alpha_j}{i-1} \quad (5a)$$

$$\alpha_x = \frac{\delta_x}{x} \quad (5b)$$

Where l_i is the mean loading interval; i is the total number of decisions; α_j is a decision interval in days per j adjacent decisions; m is the number of all possible decision intervals; x is the number of related decisions within a decision interval; δ_x is the breadth of a decision interval. Figure 4 graphically displays equation (5) for the scenario of table 1.

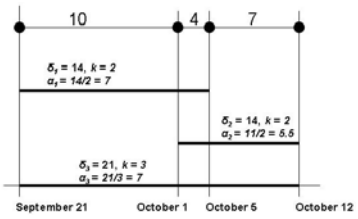

Figure 4: Graphical representation of interval calculation for time variant contexts.

Therefore equation (5) yields to:

$$l_i = \frac{\left(\frac{10+4}{2}\right)+\left(\frac{4+7}{2}\right)+\left(\frac{10+4+7}{3}\right)}{3} = 6.5 \cong 7\ days \qquad (6)$$

CAEP used the environmental contexts to decide whether the decision was made inside a library, on a corridor, inside a cafeteria, in his own room, or in a lecture room. Table 2 summarises the aggregate decision-context association. Note that *lecture room* is a higher-level context that was a result of the aggregation of decision-context associations of table 1.

TABLE 2: A SUMMARY OF LOADING DECISION AND CORRESPONDING ASSOCIATION.

Decision	**Context**		
	Place	Time	Day
Load	Lecture Room	10:02 AM{±7}	Tuesday

III. CONCLUSION

Reduction of obtrusiveness is a major goal in context-aware computing. Whereas enriching applications with context information reduces the explicit input a user has to provide for a computing task, there are, however, contexts, which cannot directly be captured by employing sensors or other context data sources. These are higher-level abstractions of a complex situation, and must be reasoned about in terms of numerous atomic contexts. Existing context-aware applications incorporate rules that dictate a composition procedure and identify at design time the type of atomic contexts, which potentially describe some properties of the higher-level context. In this paper we introduced a way to exploit runtime contexts instead of defining atomic contexts at design time. As a demonstration, we introduced the Context-Aware E-Pad (CAEP). CAEP associates a user's interaction with a set of atomic contexts that are collected at runtime from a computing environment; where a computing environment encompasses persons, devices, places, and the application itself. By observing repeated decisions a user makes and by aggregating correlated decisions and runtime atomic contexts, CAEP (1) shields the user from the concern of lower-level context details; (2) reasons about higher-level contexts using the set of atomic contexts and previous context-decision associations; and (3) helps the user define additional higher-level contexts in terms of decisions.

IV. ACKNOWLEDGEMENT

We would like to acknowledge partial funding of the German "Bundesministerium für Bildung und Forschung (BMBF) in the framework of the Wireless Internet and mik21 projects.

V. REFERENCE

[1] A. Dey, "Providing Architectural Support for Context Aware Applications," PhD Dissertation, pp. 41, 81, 2002.

[2] W. Dargie, O. Droegehorn, K David, "Sharing of Context Information in Pervasive Computing," In Proc. of the 13th Mobile and Wireless Communication Summit, pp. 839 – 843. IST, 2004.

[3] P. Yancey and P. Brand, "In His Image," Sondervan Publisher, pp. 131-132, 1987.

[4] J. Pascoe, "Adding generic contextual capabilities to wearable computers," In the Proceedings of the 2nd International Symposium on Wearable Computers (ISWC'98), pp. 92-99. IEEE, 1998.

[5] N.H. Cohen, A. Purakayastha, J. Turek, L. Wong, D. Yeh, "iQueue: A Pervasive Data Composition Framework," In the Proceedings of The 3rd International Conference on Mobile Data Management. IEEE, 2002.

[6] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Kernen, and E-J. Malm, "Managing Context Information in Mobile Devices," Pervasive Computing, IEEE, 2003.

[7] http://www.ashrae.org

[8] T. Malmstrom, J. Andersson, F.R. Carrié, P. Wouters, and Ch. Delmotte, "Source Book for Energy Efficient Air Duct System in Europe," Airways Partners, 2002.

[9] C.W. Bayer, R.J. Hendry, S.A. Crow, and J. Fisher, "The Relationship between Humidity and Indoor Air Quality in Schools," In Proc. Indoor Air, 2002.

[10] D.R. Wulfinghoff, "Energy Reference Manual," Energy Institute press, 1999.