Towards Scatterbox: a Context-Aware Message Forwarding Platform*

Stephen Knox, Adrian K. Clear, Ross Shannon, Lorcan Coyle,
Simon Dobson, Aaron J. Quigley, and Paddy Nixon

Systems Research Group, School of Computer Science and Informatics
UCD Dublin IE
stephen.knox@ucd.ie

Abstract. Context-aware systems that rely on mobile devices for user interaction must address the low bandwidth of both communications and more importantly the user's limited attention, which will typically be split between several competing tasks. Content delivery in such systems must be adapted closely to users' evolving situations and shifting priorities, in a way that cannot be accomplished using static filtering determined *a priori*. We propose a more dynamic context-driven approach to content delivery, that integrates information from a wide range of sources. We demonstrate our approach on a system for adaptive message prioritisation and forwarding.

1 Introduction

Rich sources of context data are an integral part of building intelligent pervasive computing applications. Central to pervasive computing is the notion that "technology recedes into the background of our lives" [1]. The classical view of human-computer interaction needs to be extended to include both a cloud of interoperating heterogeneous electronic devices and the ability to interact with one or many when and wherever desired. This extended view requires that disparate devices are able to interoperate easily. This supports enhanced interactions with the user which break away from traditional distribution channels, reaching the user through whichever device they currently have available. As small mobile devices with built-in wireless capabilities such as mobile phones and PDAs become more widespread, they present an ideal opportunity to afford ubiquitous communication services to the user.

Advances in technology in recent years have meant that computing devices have become cheaper, smaller, and more powerful. Mobile devices including mobile phones and PDAs are constantly increasing in utility through the addition of extra sensory equipment like gyroscopes, accelerometers and GPS receivers. In addition, communication devices are beginning to be embedded into everyday objects such as toys, refrigerators and coffee cups [2, 3]. Each of these advances expands both the opportunities

* This work is partially supported by Science Foundation Ireland under grant number 05/RFP/CMS0062, "Towards a Semantics of Pervasive Computing.", grant number 04/RPI/1544, "Secure and Predictable Pervasive Computing.", grant number 03/CE2/I303-1, "LERO: the Irish Software Engineering Research Centre.", and by an EMBARK Scholarship from the Irish Research Council in Science, Engineering and Technology.

for the delivery of ubiquitous communication services but also increases the system complexity in coordinating this cloud of devices.

With the tremendous increase in available contextual data, a context-aware system can determine many facts from the environment that can inform its behaviour e.g., who is present, or what task they are performing. An example of this is a smart meeting room that manages shared projectors [4], or a smart room for elderly people which detects incidents such as falls and reacts to them quickly [5]. These decisions can only be made if there are enough data, of sufficient fidelity, to support them. Therefore there must be many sensory devices, each providing differing inputs, that each contribute to the knowledge of the system as a whole. These devices will often go unnoticed by the user, but they provide valuable information about the user's surroundings and the context of their activities.

We define "context" as any aspect of the environment of a system understood symbolically – or, more concretely, as a measurable component of a given situation. By "situation" we mean a certain composition of various simple and derived contexts that gives rise to pervasive services. These contexts may be simple metrics which can be investigated with instruments, such as a time context (e.g. 18:48 GMT), a location context (e.g. Coffee Area) or a user context (e.g. Bruce), but also range to more complex computations such as a user's current social context, meaning the other users that they are sharing a space with.

A central challenge for pervasive computing is to integrate contextual information in order to best recognise and service the changing situation. Since individual sources of context are inherently error-prone, this cannot be accomplished by focusing on any one source but instead requires a fusion-based approach that can integrate all available information, giving due weight to the fidelity of each source.

This paper describes an approach to situation awareness being prototyped in a system we call Scatterbox, a "moving letterbox" that delivers relevant messages to a user's mobile device based on the context derived from sensors within a pervasive computing environment. The goal of this system is to take multiple, heterogeneous sources of contextual data, and extrapolate the situations that they map to. The characteristics of these situations define whether an appropriate action should be taken — and so whether a certain message should be delivered to a user's phone or PDA, limiting distraction by only sending messages that are timely and relevant. In accordance with the study done by Oulasdvirta *et al.* [6] on the "drastically short term" limited attention span of mobile users, Scatterbox provides short, concise messages requiring minimal attention.

The system we propose monitors a user's e-mail inbox and dynamically forwards messages to him, depending on his situation. This means that he will be notified of the receipt of only important e-mails or messages relevant to his current task when he is in a certain situation, while all other messages will be stored as normal in his e-mail inbox. This provides a tangible demonstrator of a real-time pervasive system which is constantly adapting to changes in the user's situation.

The remainder of the paper is organised as follows. Section 2 describes related research in the area of contextual message forwarding. Section 3 contains a formal description of our approach to the composition of context into situational awareness. Section 4 describes our context acquisition and reasoning methods using a contextual

framework, while Section 5 describes how our Scatterbox system has been implemented to work in a physical location. Section 6 outlines our proposed evaluation that will be performed to test the efficacy of Scatterbox using real messages, real users and real context. In the final section, we offer some conclusions and plans for future work.

2 Related Work

Standard message delivery systems such as email and SMS do not take the user's context into account, and can make it difficult for a user to prioritise inputs, as well as lead to irritation due to unnecessary disruption. Context-awareness can be used as an effective augmentation of existing message delivery systems. Filtering messages using context can be used as a means to decrease the disruption that message delivery can cause to a user. This approach to message delivery has the potential to become the cutting-edge in messaging, self-organisation, and content filtering.

Multiple approaches to context-aware message delivery have previously been explored. The Stick-e note architecture [7] is one of the earliest of these. Stick-e notes can be messages or other objects which have contexts attached to them. This architecture incorporates the user's physical environment into service delivery; the principle contexts used are location and time. The notes may be stored on a user's PDA or a static device like a desktop PC. The message or object gets delivered to a user only when they enter the context that is attached to the object. For example, users may be reminded that they have to return a book to somebody when they are at their bookshelf and in the presence of the person from whom they borrowed the book.

Nakanishi *et al.* [8] proposed the Context Aware Messaging Service that uses schedule information, location information and available media to send an incoming message (or call) to users using an appropriate protocol and device. This system uses context to determine whether the message should be sent using e-mail, SMS, and so on, and what device it should be sent to. The authors performed an evaluation of a two month experiment in Tokyo. The results suggest that the use of context to determine what device a message should be sent to made the retrieval of messages more convenient. It also made users feel comfortable to know that another user would not be disturbed by a message if the timing was inappropriate.

Context-aware adaptation has also been introduced to applications with the goal of reducing a user's distractions. Miller *et al.* [9] leverage the Aura Contextual Information Service in order to build distraction-free context-aware applications. In their example, urgent messages should be sent using SMS or IM depending on the situation.

Ho *et al.* [10] also propose the use of context in order to prevent the user from being interrupted by messages, calls, etc. at inappropriate times. They list 11 factors which influence a person's interruptibility at a given moment and complete a user trial which shows interesting results. People are less likely to be disrupted when they are between activities (e.g., between a meeting and going to lunch).

3 Foundations of Context-Awareness and Adaptive Behaviour

The notion of context is key to context-aware computing, and a universally accepted definition has been difficult to realise. Yau [11] defines context as "any instantaneous, detectable and relevant property of the environment, the system or users", which is similar to, but more general than, Dey's [12] definition that context is "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Henricksen [13] proposes to make a distinction between the concepts of *context* and *context modelling* in order to achieve consensus and precision:

- "The context of a task is the set of circumstances surrounding it that are potentially of relevance to its completion."
- "A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time."

We aim to take a step further and formalise the notion of context based on the above definitions. The motivation for such a formalism is the need to derive an appropriate means to model context. From Yau's mention of the properties of the environment; Dey's mention of information about an entity; and Henricksen's mention of the more general term "circumstances", we propose to capture the structure of context in a tuple containing a subject, predicate and object. We can then view a predicate as a means to relate properties, information, or circumstances to an entity. Thus, we can express any detectable or realistically attainable facts relevant to the completion of a task.

Definition 1. *A context is a tuple containing a subject, predicate and object (s,p,o) that states a fact about the subject, where*

1. *The subject is an entity in the environment.*
2. *The object is a value or another entity.*
3. *The predicate is a relationship between the subject and object that defines the domain of the object.*

A context may be either a constant or a variable. A constant context must take on a single value from a domain. A variable context may take on values from a domain. The value of a variable context may change over time, while the domain stays the same.

An environment of a context-aware system can be viewed as a finite number of constant and variable contexts. The values that a variable context may attain come from a well understood domain of variable types including nominal or categorical e.g., times of year {Summer, August,...}, ordinal, quantitative, interval e.g., temperature range {-32, +32}, and ratios. The domain of an object captures the "type" of information, property or circumstance that we may legally relate to the subject using a particular predicate.

Context is very fine-grained for use in developing complex adaptive applications, as it is typically a low-level interpretation of raw sensor data. When defining adaptive behaviour, we consider a holistic view, using situations, rather than the combination of individual bits of context data. A situation is a natural abstraction of context data, providing more realistic points to associate adaptive behaviour with. For example, imagine that a user, Bruce, in a lecture theatre, has a slide set open and there are 35 other people in the room. Bruce is at the front of the room and is talking. This presents a lot of contextual data for us to comprehend in parts, but a context-aware system should be able to condense it and infer that the current situation is a "presentation". Often individual component contexts may change but the situation will be maintained. Other situations will be labeled "meeting", "lunch", etc. This view of contexts and situations proves to be quite versatile.

In order to define situation, we firstly define *situation space*. Given that each predicate restricts an object to a particular domain, we can define the set of variable contexts A with common subject s_a and predicate p_a as $A : \langle s_a, p_a, o_i \rangle$ for all $o_i \in D_i$, where D_i is the domain of o_i. Given two or more sets of contexts, we can define their situation space as the Cartesian product of the sets as follows.

Definition 2. *Given the sets of contexts* $A : \langle s_a, p_a, o_1 \rangle$, $B : \langle s_b, p_b, o_2 \rangle \ldots Z : \langle s_z, p_z, o_n \rangle$ *for all* $o_i \in D_i$, *where* D_i *is the domain of* o_i, *we can define the* situation space $AB \ldots Z$ *as the set* $\{(a_1, b_1 \ldots, z_1), (a_1, b_1 \ldots, z_2) \ldots (a_m, b_n \ldots, z_p)\}$ *where* $a_1 \ldots a_m \in A$, $b_1 \ldots b_n \in B \ldots$, *and* $z_1 \ldots z_p \in Z$.

A situation space thus captures all possible combinations of two or more contexts. We can thus define a concrete situation as a subset of a situation space.

Definition 3. *A situation is a subset of a situation space.*

Given a situation space AB defined over the variable context sets A and B, we can define the situation AB_1 as $\{(a_1, b_1), (a_3, b_2)\}$. Situations themselves may be used in the construction of more complex situations. For completeness, a base behaviour should be provided in order to cover all possible circumstances. For a more concrete example, imagine we have the context sets over the domains $Months$ and $Weather$. We can define the situation space for $Weather \times Month$ and define situations like $SummerSun$ or $SpringWind$.

Adaptive behaviour is achieved by creating points in the system called *adaptation points*. When one of these is reached, the system exhibits a corresponding behaviour. An adaptation point could be as simple as a context variable taking on a certain value or as complex as an elaborate situation composition. In general, we are selecting states of the system where something useful should happen. We may wish to set a user's phone profile to silent when she is in a meeting, for example, or set her Instant Message (IM) status to away when she is out on lunch.

4 Context Acquisition and Modelling

A context-aware system collects relevant context data from the environment and then acts appropriately based on this information. Our context-aware applications are built

on top of Construct [14], a distributed fully decentralised open-source platform supporting the building of context-aware, adaptive, pervasive and autonomous systems[1]. Such systems interact through manipulation of a common data model rather than through the piecing together of services. Construct provides applications with a uniform view of context information regardless of how it was derived. Construct takes care of collating and distributing context data around a network. Applications can then query the nearest Construct node for data that their service relies on.

4.1 Sensors

Sensors are used to gather context data from the environment. We have categorised our sensors into two types: physical and virtual. Physical sensors directly detect characteristics of the environment e.g., sensors for location data obtained from Ubisense[2] and Bluetooth spotters. Virtual sensors obtain information from the Internet, local network or local computer, e.g., sensors for obtaining calendar information, monitoring computer activity and collating syndicated data feeds.

In this project we utilise the following physical and virtual sensors:

- **Ubisense Location Sensors** poll a Ubisense location tracking system which has been set up throughout our research lab and tracks special tags that users can carry around with them. The sensor calculates the x, y, and z coordinates of an individual with a peak granularity of 30cm in 3D space.
- **Bluetooth Location Sensors** poll for a user's designated Bluetooth-enabled device. A number of statically positioned "base stations" have been positioned throughout the lab. Due to Bluetooth's limited range, the set of static devices within the system which can connect to a mobile Bluetooth device allow us to infer a range of possible positions for the device.
- **Calendar Sensors** monitor a list of published iCal and vCal calendars for data about a user's appointments and location (e.g. room number) and availability for a period of time (e.g. during the next week).
- **Computer Activity Sensors** determine whether an individual is located at a computer by checking if they are logged-in and active at that terminal.

Data from each of these sensors is fed into a connected Construct node, from where it is disseminated to other nodes in the system. The Construct middleware is being complemented by the development of an uncertainty framework [15]. This performs aggregation of context and deals with inconsistencies that may arise when the same type of data is produced from different sensors (e.g., location sensors providing data that says a person is in two places at once).

4.2 Modelling Context, Situations and Behaviour

In order for the heterogeneous nodes running Construct to be able to interpret and communicate about context and situations, we need a means to model them.

[1] The Construct home page is located at http://construct-infrastructure.org/

[2] Learn more about Ubisense at http://www.ubisense.net/

We model context and situations as ontologies in the Web Ontology Language (OWL)[3]. Our reason for choosing OWL is that it fits our mathematical description of context and situations from Section 3 most appropriately. Entities and context categories are modelled as OWL classes. Their attributes correspond to context variables and constants from the previous section. For example, the Person class has constants such as name and e-mail and variables such as location. In order to create a person, an instance (or individual in OWL terminology) is made of this class and values are assigned to its attributes. Variable contexts may be changed frequently and timestamped by sensors.

We create situation spaces as OWL classes in a similar way. Their attributes are the context variables or constants which the space encapsulates. In order to create concrete situations, we create instances of the situation spaces by assigning values or intervals to the contexts. This is similar to composing context variables and creating a subset of the resulting situation space as described in the previous section.

Adaptation points can be viewed as augmented virtual sensors which consist of a context or situation and a corresponding behaviour. The virtual sensors asynchronously poll Construct to monitor whether their situations have been realised. They then execute their behaviours and possibly introduce more contextual information into the network. The prescribed behaviours may require context information to function e.g., if we wish to route a message to a user, we must first choose a device, using location information, to send it to. As mentioned in Section 3, for completeness adaptation points should be specified for all subsets of a situation space. One means to approach this requirement is to introduce a default behaviour on a subset of a situation space and specialise the behaviour for other, more significant subsets.

The expressiveness of our mathematical model for situations leads to some problems regarding conflict, however. By allowing situations to be composed of other situations, and adaptation points to be defined on elements as fine-grained as individual contexts, we must decide what behaviour the system should exhibit in occurrences where situation definitions overlap. Some possible approaches to this problem are given in Ye *et al.* [16].

5 Scatterbox

In order to demonstrate our approach to situation determination and use of context in pervasive systems, we are developing Scatterbox, a context-aware message forwarding platform. Scatterbox forwards certain incoming e-mails to users in a pervasive environment based on their context. The user's context is found by tracking his location and monitoring his daily schedule. This context data is accessed through Construct, and situations are identified based on this data. As messages arrive, Scatterbox forwards them to subscribed users should their situation warrant it.

We now describe in detail how Scatterbox is implemented. We illustrate the situations and behaviours that the application uses, describe our mode of message delivery, and show the use of the application with an example case study.

[3] The OWL specification is located at http://www.w3.org/TR/owl-features/

5.1 Situations and Behaviour

The situation spaces that we use for Scatterbox are *EmailRelevance*, *Meeting*, and *PresenceAtInbox*. *EmailRelevance* is dependent on factors such as the situation of the user, the sender of the e-mail, the e-mail content and the user's calendar information. In order for Scatterbox to know when a situation has occurred, it must poll Construct regularly for newly acquired context, and match the new context to the situation spaces.

For the Scatterbox application, we need only define one adaptation point, *relevantEmail*, which reacts by sending a summary message to the user's Bluetooth device. This behaviour can be defined as *sendMessage(user, msg)*. This command can be entered into Construct, distributed, and picked up by a node within range of the user's Bluetooth device. An application running on the Construct node then sends the message to the device.

5.2 Message Delivery

Bluetooth wireless technology is built into most devices that we carry around every day, like mobile phones, PDAs and other portable devices, which means Scatterbox is transparent, reliable, and scalable.

Transmission of messages is accomplished through Bluetooth's Push protocol. The Bluetooth capabilities of most mobile phones and similarly-powered devices are typically limited to a transmission range of approximately 10 metres. Each device can be uniquely identified within the system. If a user's Bluetooth device is in range of a Bluetooth-enabled Construct node, a message can be routed to the node and pushed to the mobile device. The message can then be accepted or rejected by the user. In Section 6 we show how we will use this acceptance or rejection to drive our evaluations.

5.3 Use case

Consider the case of Bruce, a user who has a meeting scheduled after lunch with one of his students. The meeting is to take place in his office, and Bruce is currently upstairs in the coffee area having lunch with one of his colleagues. The system's contextual inputs at this point are:

- The current time of day.
- Schedule information from Bruce's calendar application (which includes both the time it takes place and the room it takes place in).
- Bruce's current position in the building.

Many calendaring applications will offer the ability to e-mail the user a reminder of an appointment such as this meeting before the meeting is to take place. However, in this case an e-mail reminder will be of no use, as Bruce is not near his computer. Scatterbox routes messages that would normally arrive in a user's e-mail inbox to their mobile device, if they are away from their computer and the message is deemed to be contextually relevant by the system. In the case of calendar appointments, it is not a requirement that an e-mail with an appointment reminder is received; the user's calendar is sufficient, as contextual data can be extracted from it directly.

Shortly before a meeting is scheduled to occur, the system will query Bruce's location. When it notices he is not in the prescribed location, a message will be pushed to his phone to remind him of the meeting.

Variants of this situation that would not lead to a message being sent would be:

- had Bruce been in his office at the time the meeting was to start.
- had the student that Bruce was to meet also been up in the coffee area at the time the meeting was to start. In this case a message will not be sent, as Bruce's social context overrides the location information.

5.4 Situations in Scatterbox

To demonstrate how contextual information can be collected about Bruce, we will use the sensors mentioned in Section 4.1. Each sensor is modelled using an ontology, which describes how that sensor's data should be interpreted. This gives uniformity to the data within the system, so in the event of a query like: "What room is Bruce in?", it is a simple matter of making a single query based on the location ontology, rather than multiple queries for each separate type of sensor data.

Scatterbox continuously seeks context data using queries such as the following:

- Select last location of Bruce.
- Select upcoming entries from Bruce's calendar.
- Select sensor readings from Bruce's computer activity sensor.

Bruce's situation can be determined from the responses to these queries. For example, a Construct node could return:

- "Boardroom, 3.02pm"
- "Meeting with head of school, Boardroom, 3.00pm"
- "Inactive"

From these results, it is inferred that Bruce is in a meeting and not in his office. He therefore should only be sent a message if it is classified as being important, relative to his situation.

Every situation is defined by the following criteria: the values context data must hold for the situation to be realised, and the resultant behaviour. We allow the user to define their own message filters, which take the form of standard e-mail filters, such as filtering by sender, recipient, or keywords. For message classification, Scatterbox takes an approach based on common spam filtering techniques, such as white-lists, black-lists, and simple keyword classification. These filters are entered into Construct in the form of RDF. From that point on, they are viewed by Construct as being additional context data.

The data going into Construct is checked against a set of ontologies. This piece of data can then be associated with something in the environment. In the case of a Bluetooth reading, the datum is seen to be an attribute of a *Bluetooth device*. This Bluetooth device ontology is associated with the *Person* ontology with the $hasCellPhone$ relation. These relations allow Scatterbox to see whether a person is in the environment,

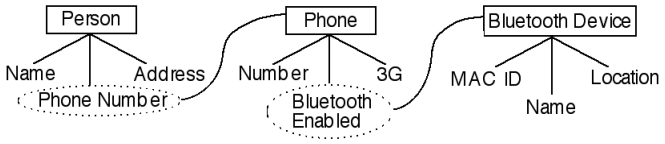

Fig. 1. An example of how ontologies are traversed.

where they are, and consequently determine their situation. This traversal of ontologies is illustrated in Figure 1.

Following on from the example above, Bruce must define how Scatterbox is to decide which messages are appropriate for each situation. This involves naming a situation and stating who he would accept messages from in that situation. An empty list implies no messages should be delivered. Secondly, he has the option of defining a set of keywords which have to appear in incoming messages for them to be deemed important enough for delivery. Finally, Bruce defines which criteria indicate a situation. He does this by creating instances of situation spaces in an ontology editor, for example, as described in Section 5.1.

From this point on, Scatterbox scans incoming e-mails and, using keyword classification, assigns e-mails to particular situations. It is assumed that the user will already have adequate spam protection preventing unwanted messages from reaching their e-mail inbox.

6 Evaluation

In order to evaluate Scatterbox's effectiveness in using context to determine which messages to forward, and the accuracy and usefulness of these messages, we have designed a user test. This is a more complex problem than standard spam filtering as we assume that context determines the level of tolerance a user has at any time for reading a message. We believe that given the situation, the willingness of a user to read a message changes in a predictable manner [10]. Therefore an evaluation of this system must account for the context that was used to support the decision to pass the user a message.

We will perform the evaluation with a number of real users and their real e-mail. The transactions take the following form: messages are sent to the user via Bluetooth's Push protocol. The user's phone then gives them the option of accepting the message or rejecting it. This user feedback is logged by the Scatterbox application running on a Construct node. The feedback is then used to determine the appropriateness of Scatterbox's decision to send the message to the user.

Our evaluation will estimate the utility of context-aware message filtering by quantifying both the number of unwanted messages that are sent to a user (false positives), and the number of important messages that are not sent to the user (false negatives). Feedback elicited from rejection of unwanted messages indicates false positives, but an alternative approach is needed to account for false negatives (messages which should have been sent but were not). For the purposes of evaluation we capture this form of

feedback by occasionally sending messages to the user that Scatterbox does not believe the user will want to see. Our evaluation will count rejections of these messages as true negatives (and thus appropriate behaviour) and acceptances of these messages as false negatives (i.e., inappropriate). We believe that it is more important for Scatterbox to avoid false positives than false negatives — since the user will always check their inbox eventually — and that our evaluation will bear this out. (Interestingly this is the opposite of what one would expect from the perspective of spam filtering.)

Further to performing a live evaluation, we propose to amass a data set of real context and behaviours over the course of this user test and use this to perform a number of offline evaluations. By gathering all relevant context features with Scatterbox's decision to send messages and the user's feedback, we will be able to investigate which features were most important in predicting correct behaviour using simple feature selection [17]. By examining how changes in context affect situation and behaviour, we will be able to examine the ability of Scatterbox to consistently and smoothly respond to its environment.

7 Conclusions and Future Work

We developed a system, Scatterbox, that determines a user's situation by composing numerous sources of contextual data. This system can then intelligently forward useful messages to a user's mobile device based on their current situation. These messages comprise both important e-mails that the user has received while they were away from their computer, and also contextual notifications of important events, such as meeting reminders derived from the user's calendar. The system notices changes in a user's situation, and reacts accordingly.

Scatterbox was created using a context infrastructure (Construct), which has been used for context reasoning in many other application areas, such as in location awareness for health care [18], smart homes [19], and recommender systems [20].

We have also designed an evaluation of this system, which uses the rate of rejection of messages to determine whether the system classified the situation correctly and if it acted appropriately.

The next step in Scatterbox's development is the addition of Machine Learning techniques to correct its distribution algorithms in a similar way to that done by intelligent spam filtering techniques. Research is also ongoing regarding the use of truth maintenance to reduce the need for continually resolving recurring inconsistencies within a data store.

References

1. Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1991.
2. Emmanuel Munguia Tapia, Stephen S. Intille, Louis Lopez, and Kent Larson. The design of a portable kit of wireless sensors for naturalistic data collection. In *4th International Conference on Pervasive Computing, Dublin, Ireland, May 7-10, 2006*, pages 117–134, 2006.

3. Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. *Lecture Notes in Computer Science*, 2201:116–122, 2001.
4. Tim Finin Harry Chen and Aravind K. Joshi. A context broker for building smart meeting rooms. In *The Knowledge Representation and Ontology for Autonomous Systems Symposium*, AAAI Spring Symposium, March 2004.
5. Vincent Rialle, Nancy Lauvernay, Alain Franco, Jean-Franois Piquard, and Pascal Couturier. A smart room for hospitalised elderly people: essay of modelling and first steps of an experiment. *Technology and Health Care*, 5(7):343 – 357, January 1998.
6. Antti Oulasvirta. The fragmentation of attention in mobile interaction, and what to do with it. *interactions*, 12:16–18, 2005.
7. Jason Pascoe. The stick-e note architecture: Extending the interface beyond the user. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 261–264, New York, NY, USA, 1997. ACM Press.
8. Yasuto Nakanishi, Takayuki Tsuji, Minoru Ohyama, and Katsuya Hakozaki. Context aware messaging service: A dynamical messaging delivery using location information and schedule information. *Personal Ubiquitous Comput.*, 4(4):221–224, 2000.
9. Nancy Miller, Glenn Judd, Urs Hengartner, Fabien Gandon, Peter Steenkiste, I-Heng Meng, Ming-Whei Feng, and Norman Sadeh. Context-aware computing using a shared contextual information service. In *Pervasive'04, "Hot Spots"*, Vienna, April 2004.
10. Joyce Ho and Stephen S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 909–918, New York, NY, USA, 2005. ACM Press.
11. Stephen S. Yau, Dazhi Huang, Haishan Gong, and Yisheng Yao. Support for situation awareness in trustworthy ubiquitous computing application software. *Software: Practice and Experience*, 36(9):893–921, July 2006.
12. Anind Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
13. Karen Henricksen. *A Framework for Context-Aware Pervasive Computing Applications*. PhD thesis, The School of Information Technology and Electrical Engineering, University of Queensland, September 2003.
14. Graeme Stevenson, Lorcan Coyle, Steve Neely, Simon Dobson, and Paddy Nixon. Construct — a decentralised context infrastructure for ubiquitous computing environments. In *IT&T Annual Conference, Cork Institute of Technology, Ireland*, 2005.
15. Simon Dobson, Lorcan Coyle, and Paddy Nixon. Hybridising events and knowledge as a basis for building autonomic systems. *IEEE TCAAS Letters*, 2007. To appear.
16. Juan Ye, Adrian K. Clear, and Simon Dobson. Towards a formal semantics for pervasive adaptive systems. *Computer Journal*, 2007. To Appear.
17. Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
18. Lorcan Coyle, Steve Neely, Paddy Nixon, and Aaron Quigley. Sensor aggregation and integration in healthcare location based services. In *First Workshop on Location Based Services for Health Care (Locare'06), Nov 28 2006, Innsbruck Austria*, 2006.
19. Lorcan Coyle, Steve Neely, Gaëtan Rey, Graeme Stevenson, Mark Sullivan, Simon Dobson, and Paddy Nixon. Sensor fusion-based middleware for assisted living. In *Proc. of 1st International Conference On Smart homes & heath Telematics (ICOST'2006) "Smart Homes and Beyond"*, pages 281–288. IOS Press, 2006.
20. Lorcan Coyle, Evelyn Balfe, Graeme Stevenson, Steve Neely, Simon Dobson, Paddy Nixon, and Barry Smyth. Supplementing case-based recommenders with context data. In *1st International Workshop on Case-based Reasoning and Context Awareness at ECCBR*, 2006.