Scatterbox: Context-Aware Message Management

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

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

ABSTRACT. *Applications that rely on mobile devices for user interaction must be mindful of 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. We demonstrate our approach using Scatterbox, a pervasive computing application we have developed which performs sensor fusion to derive a user's current situation. Based on the user's level of interruptibility, Scatterbox prioritises and forwards relevant messages to their mobile phone. We draw conclusions from a preliminary evaluation of the system.*

RÉSUMÉ. *Les applications embarquées dans les appareils portables tels que téléphones ou PDAs et qui interagissent avec leur utilisateur, doivent prendre en compte la disponibilité de celui-ci. L'utilisateur exécute en effet en général des tâches n'ayant pas de lien avec celles pour lesquelles il reçoit une notification de son appareil. Pour de tels systèmes, les priorités de l'utilisateur doivent être prises en compte pour lui livrer des messages pertinents á un instant donné. En général, ceci ne peut pas être effectué á partir de filtres calculés de manière statique. Une approche de livraison de messages, dynamique et dépendante du contexte de l'utilisateur, est ici proposée. Cette approche est illustrée en utilisant Scatterbox, une application pervasive qui permet la fusion d'information afin de déterminer le contexte dans lequel se trouve l'utilisateur. En fonction du niveau de disponibilité de celui-ci, Scatterbox rend prioritaires certains messages et les envoie sur son téléphone portable. Nous tirons ces conclusions suite á une évaluation préliminaire du systéme*

KEYWORDS: *situation-awareness, context, pervasive computing, sensor fusion*

MOTS-CLÉS : *situation consciente, contexte, application pervasive, fusion d'information*

RSTI - RIA — 22/2008. Modelling and Reasoning on Context, pages 549 to 568

1. Introduction

Rich sources of context data are an integral part of building intelligent pervasive computing applications. A central concept in pervasive computing is that "technology recedes into the background of our lives" (Weiser, 1991). To accomplish this, the classical view of human-computer interaction needs to be extended to include a cloud of interoperating heterogeneous electronic devices that offer services to the user. This scenario requires that disparate devices are able to interoperate easily, breaking away from traditional distribution channels and 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.

Despite all of the recent advances in the mobile device space, the majority of office workers are still tied to their desktop email clients throughout the day. Email has been overloaded as a means to communicate information to the user, and is now used to pass along personal messages, work notices, mailing list communications, calendar reminders, notifications of messages on social networking websites, calls for papers, news alerts, and more. The constant influx of information from these disparate sources, with different (and sometimes shifting) priorities, have left many users feeling overburdened and out of control (Whittaker *et al.*, 2006). Many find themselves constantly tending their email rather than doing meaningful work. Deciding which emails are the most important ones to read first can be a challenge in itself.

Mobile devices are now capable of performing as fully-featured email clients, but encounter problems due to their limited screen size, and the "drastically short term" limited attention span of mobile users (Oulasvirta, 2005). By notifying mobile users of every new email, people are being increasingly distracted in situations where they are even less likely to be able to attend to every incoming communication. Because of this, there is obvious utility to a method of filtering the messages soliciting the user's attention. Static email filtering is limiting, as it does not change over time (Zhou *et al.*, 2005), and thus cannot keep up with the user's constantly changing situations and tasks. A more dynamic approach is required.

Context-aware systems can determine many facts from the environment that can inform their behaviour, for example, who is present, or what task they are performing. An example of this is a smart meeting room that manages shared projectors (Chen *et al.*, 2004), or a smart room for elderly people which detects incidents such as falls and reacts to them quickly (Rialle *et al.*, 1998). Further related research in context-aware systems is discussed in Section 2.

Decisions in these systems can only be made if there are enough data, of sufficient fidelity, to support them. Therefore there must be a network of sensory devices, each providing differing inputs, that 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 view "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 UTC), 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, which could be a list of the people that are in their immediate vicinity. Section 3 contains a formal description of our approach to the modelling and composition of context into situational awareness.

A major challenge in pervasive computing is to manage the continual integration of additional contextual information in order to best recognise and service the changing situation. Since individual sources of context are inherently error-prone (Hightower *et al.*, 2002), 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 weighting to the fidelity of each source. Section 4 describes our context acquisition and reasoning methods using a context framework.

This paper describes an approach to situational 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 mobile device, limiting distraction by only sending messages that are timely and relevant. Scatterbox allows the user to stay mobile while respecting both the limitations of the communication mechanism and the user's attention by providing short, concise messages requiring minimal attention.

The system we have designed monitors a user's email inbox and dynamically forwards messages to him, depending on his situation. This means that he will be notified of the receipt of only important emails 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 email inbox. Section 5 describes the design of the Scatterbox application, which is based on our previous work on this system which was presented in Knox *et al.* (2007), while Section 6 describes the current implementation in detail. This system provides a tangible demonstrator of a real-time pervasive system which is constantly adapting to changes in the user's situation. Section 7 outlines our ongoing evaluation 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

Including context in the decision-making process of pervasive systems provides clear research challenges in modelling, reasoning about and managing all of this data. There have been many approaches proposed to deal with these challenges.

Wang *et al.* (2004) propose CONON, an ontology for modelling and reasoning in pervasive systems. They used first order logic to reason over data, as well as ontological reasoning. They identified issues in reasoning over large data sets in real time, a result of the use of large ontologies. To control the scale of the data sets used, they suggest the use of domain-specific ontologies. In Scatterbox, we do this by combining all ontologies for location sensors into one model, so as more location sensors are added, the complexity remains constant.

Padovitz *et al.* (2005) developed an approach to context fusion, in which a set of "context attributes", or raw sensor data, define "context states", which in turn define a "situation space". Algorithms for fusing uncertain context attributes to accurately derive situation spaces are also described. We leverage some of the ideas in this work to provide a more extensible framework for dealing with context fusion.

Context-awareness can be added 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 a standard feature of messaging, self-organisation, and content filtering.

Multiple approaches to context-aware message delivery have previously been explored. The Stick-e note architecture (Pascoe, 1997) 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 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.* (2000) proposed the Context Aware Messaging Service that uses schedule information, location information and available media to send all incoming messages (or calls) to users using an appropriate protocol and device. This system uses context to determine whether the messages should be sent using email or SMS and what devices they 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 people around them would not be disturbed by messages at inappropriate times.

Context-aware adaptation has also been introduced to applications with the goal of reducing a user's distractions. Miller *et al.* (2004) 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, IM, or email depending on the user's current state or situation (e.g., only the most urgent messages would be sent if the user was in a lecture).

Ho *et al.* (2005) propose the use of context in order to prevent the user from being interrupted by messages, calls, etc. at inappropriate times. They list eleven factors which influence a person's interruptibility at a given moment and present a user trial which shows, for example that people are more likely to be tolerant of interruption when they are between activities (e.g., between a meeting and going to lunch). Their results support the argument that message forwarding will be more useful as the accuracy of situation determination increases.

3. Foundations of Context-Awareness and Adaptive Behaviour

Although the concept of context is key to context-aware computing, a universally accepted definition has been difficult to realise. Yau *et al.* (2006) define context as "any instantaneous, detectable and relevant property of the environment, the system or users", which is similar to, but more general than, the definition offered by Dey (2001) 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 (2003) 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 refine the above definition and formalise the notion of context. 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, most typically a user. 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.

The environment of a context-aware system can be viewed as a finite number of constant and variable contexts. Context values have well understood types; they may be nominal or categorical, for example, times of year {Summer, August, ...}, ordinal, quantitative, interval, for example, temperature range {-32, +32}, or 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. For example:

$\langle RoomB1.08 \rangle \ \langle hasTemperature \rangle \ 21$

$\langle hasTemperature \rangle \ \langle hasDomain \rangle \ \langle degreesCelcius \rangle$

The view of context we have adopted can be represented using well-understood standards in computer science, specifically the Resource Description Framework (RDF) (W3C, 2004). This framework provides the basis for generating more complex information structures. We refer to these structures as situations.

Situations are derived from fusing 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.

To generate situations within a computing system, we must formally define what a situation is with respect to context. In order to define situation, we first 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 the object 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 therefore captures all possible combinations of two or more contexts. Contexts can be either raw sensor data or derived from a number of sensors. The generation of a situation space is how we perform sensor fusion.

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.*

Imagine we have the context sets over the domains $Location$, $Schedule$ and $Time$. We can define the situation space for $Location \times Schedule \times Time$ and define situations like $OnTheWay$ or $RunningLate$.

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. For example, a message arriving from a colleague may not be initially forwarded to the user's mobile phone, due to the fact that they were not interruptible. However, a change in the user's location, schedule, or even a change in time might change their situation and give this message a new relevance, resulting in Scatterbox deciding to now forward the mail.

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 (Stevenson *et al.*, 2005; Coyle *et al.*, 2006b), a distributed, fully decentralised, open-source platform supporting the building of context-aware, adaptive, pervasive and autonomous systems[1].

Construct-enabled applications interact with sensors and each other 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, by continually collating and distributing context data around the local network. Applications running within this network can then query Construct for data that their service requires.

4.1. *Sensors*

All context used by Scatterbox (and by Construct in general) is gathered by sensors. Sensors are categorised into two types: physical and virtual. Physical sensors

1. More information is available from the Construct home page:
`http://construct-infrastructure.org/`

directly detect characteristics of the environment for example, sensors for location data obtained from Ubisense[2] and Bluetooth proximity sensors. Virtual sensors obtain information from the Internet, local network or local computer, for example, sensors for obtaining calendar information, monitoring computer activity and collating syndicated data feeds.

Scatterbox utilises 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 tags carried by users of the system. The sensor calculates the x, y, and z coordinates of an individual with a peak granularity of 30cm in 3D space.

– **Bluetooth Proximity 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 at any time allow us to infer a range of possible positions for the device.

– **Calendar Sensors** monitor a list of web-published Google 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 passed to Construct. Sensor data in Construct is modelled using the best principles in pervasive ontology modelling (Ye *et al.*, 2007). Furthermore, the Construct middleware is being complemented by the development of an uncertainty framework (Dobson *et al.*, 2008). This performs aggregation of context and deals with inconsistencies that may arise when conflicting 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 Situations and Behaviour*

In order to make it easier for applications to be able to interpret context and situations, we use Semantic Web technologies to work with them. We model situations as ontologies in the Web Ontology Language (OWL)[3] [4]. 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 *email*, and variables such as

2. More information about the Ubisense positioning system is available from their website `http://www.ubisense.net/`
3. The OWL specification is located at `http://www.w3.org/TR/owl-features/`.
4. We define our ontologies using the OWL DL dialect

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 are timestamped by sensors.

We create situation spaces as OWL classes in a similar way. Their attributes are the context variables or constants that 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. These 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 for example, if we wish to route a message to a user, we must first choose the best means to send it, using location information.

5. Scatterbox Design

Scatterbox has been designed to serve as a test bed for context-aware computing in a pervasive computing environment. It provides a content-filtering service to its users by forwarding relevant messages to their mobile phone. The user's context is derived by tracking his location and monitoring his daily schedule. These context data are accessed through Construct, and situations are identified that indicate the user's interruptibility. As messages arrive, Scatterbox forwards them to subscribed users provided the user's available context suggests they are in a situation where they may be interrupted.

5.1. *Use case*

We demonstrate how Scatterbox works in practice using an example interaction that shows how a reminder email from a calendar application is processed and sent to a Scatterbox user, Bruce. Bruce 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. Bruce's calendar application sends him an email to remind him of this appointment. In this case an email reminder will be of no use to Bruce, as he is not currently at his computer. Scatterbox should forward messages that normally arrive in a user's email inbox to their mobile device, if they are away from their computer and the message is deemed to be contextually relevant by the system. Scatterbox'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, as well as who else is invited).

– Bruce's current position in the building.

– The details of other participating users that are currently in the room with Bruce (in this case, his colleague's details).

Scatterbox should decide if this message is contextually relevant and whether or not the message will be pushed to his mobile phone. 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, and we infer that they have decided to change the venue of their meeting.

5.2. *Message Delivery*

In January 2008, 79 new mobile phone models were introduced worldwide, 57 of which were Bluetooth-enabled (Robinson, 2008) and all 79 of which supported the sending of SMS. Transmission of messages from Scatterbox to a user's mobile phone is accomplished through Bluetooth's Push protocol, which allows a file to be transferred between devices. Each device can be uniquely identified within the system. The Bluetooth capabilities of most mobile phones and similarly-powered devices are typically limited to a transmission range of approximately 10 metres. If a user's Bluetooth device is in range of a Bluetooth-enabled node, a message can be routed to that node and pushed to the mobile device.

When it arrives on the user's handset, they have the opportunity to accept or reject it based on whether they want to receive messages in their current situation, and this decision is passed back to the Scatterbox system. In Section 7 we show how we can use this acceptance or rejection to drive our evaluations.

We also use SMS, or Short Message Service text messages as a fallback. SMS is a standard feature of all new mobile phones, and so ensures compatibility with all users. SMS messages are limited to 160 characters and do not afford the user the ability to actively reject messages that are sent at times they would prefer not to be disturbed, so we use this method of transmission only when Bluetooth is not available.

5.3. *Situations in Scatterbox*

To demonstrate how contextual information can be collected about Bruce, we will use the sensors mentioned in Section 4.1. Scatterbox has been designed to accept input from various sensor types. The heterogeneity of these sensors requires Scat-

terbox to fuse incoming contextual data. All data from functionally similar sensors conforms to the same specification, giving uniformity to the data within the system. For example, Ubisense and Bluetooth scanners both track a user's location to various degrees of granularity, but the basic information they provide is very similar. This makes querying for a user's location easier, as no device-specific queries are needed.

Scatterbox obtains 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 by the Situation Reasoner 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 and relevant to his situation. When Bruce changes location or as time passes, it is possible that his situation will change, and if so the system's behaviour will change accordingly.

Context data can be effectively represented as a graph as mentioned in Section 3. In RDF terminology, the nodes of the graph are subjects and objects, and the edges between them are predicates. This structure allows complex relationships to be represented by a traversal of the edges. An example of this can be seen in Figure 1. A Bluetooth device is seen by a proximity sensor, and so is given an associated location. By traversing the context graph we deduce that the device is a phone, which belongs to a certain person. As a result, we know the location of that person. [5].

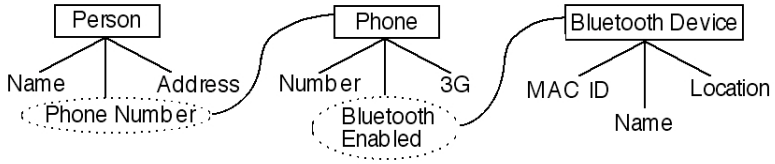

Figure 1. *By modelling context data using ontologies, we can infer complex relationships given single sensor readings*

In order for Scatterbox to best identify which messages should be sent in each situation, it must first learn the user's preferences. When starting to use Scatterbox, there

5. The ontologies currently in use can be found at *http://kind.ucd.ie/ stephenknox/ontologies/*

are no initial user preferences, so we ask the user to provide some initial information to bootstrap the process. Before starting to use Scatterbox, Bruce selects a number of emails and flags the senders as being the type of correspondents that typically send important messages (e.g., close friends or colleagues are more likely to be important than others). He does this by assigning the senders' email addresses to a number of ranked sets, which denote their importance.

Initially Scatterbox uses this information to predict whether an email is important, but as Bruce uses the system, providing corrective feedback, this bootstrapping method is used less and less. This corrective feedback comes in two forms. The first is when he rejects messages on his phone that should not have been sent (false positives). The second is when at the end of the day Bruce flags emails in his email client that should have been sent to him but were not (false negatives). This reinforcement learning process will eventually generate an accurate mapping between emails and situations.

Scatterbox will improve its performance in two ways; by learning better email classification, and by improving its situation determination. It learns the email classification task by re-weighting the relative importance of a user's correspondents. If a mail from someone is constantly being flagged as a false positive, that email address's weight will be reduced, and vice versa.

6. Implementation

Our implementation of Scatterbox acts as a testbed for our context and situation reasoning techniques. Scatterbox is currently deployed with a limited number of users. The automatic learning and feedback algorithms are not yet fully developed but provide adequate support for offline evaluation.

Scatterbox consists of the following components, as shown in Figure 2:

– A context gatherer that queries Construct for all relevant context about the user;

– A Bluetooth and calendar sensor which provide Scatterbox with contextual data;

– An email handler that interacts with an email server to access a user's email and determine each email's importance;

– A situation reasoner that takes these context data and determines whether or not the user is interruptible, and whether a particular message is relevant enough to be forwarded to the user;

– The message delivery component, which sends relevant messages to the user via either Bluetooth or SMS.

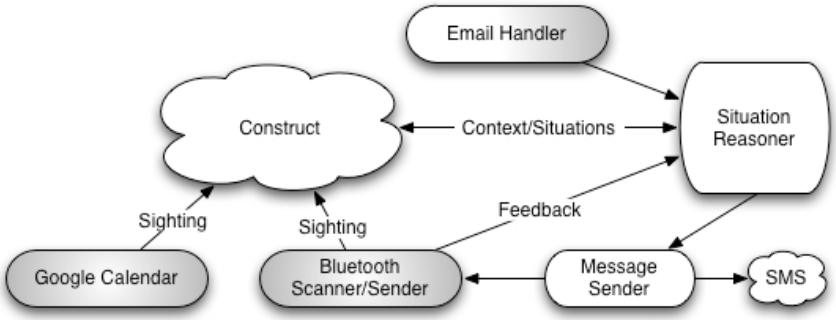

Figure 2. *Scatterbox architecture diagram*

6.1. *Context Gatherer*

To determine the context of a user, Scatterbox makes a number of queries to Construct. Provided the appropriate virtual sensors are connected to Construct, Scatterbox will have access to each user's calendar and location data.

6.1.1. *Calendar Data*

Google offer a service called Google Calendar[6] that allows its users to populate and publish an online calendar. Users can schedule entries with a title, location information, invited participants, time, duration and description.

We have implemented a calendar sensor which polls a published Google Calendar for all information about events scheduled in the next twenty-four hours. The sensor also detects supplementary information from keywords we have defined in the title of an event. These keywords allow users to specify situations directly, for example, if the calendar entry's title contains the keywords "Busy", "Seminar" or "Meeting", the interruptibility of the user is reduced for that period. Alternatively, if the location of the event was in the coffee area and the title read "Lunch", then their interruptibility is increased. Again, these manual cues give Scatterbox useful training data with which to learn how to determine the characteristics of a user's situation.

6.1.2. *Bluetooth Component*

The Bluetooth component running on desktop machines in our lab has two functions. It constantly scans for mobile Bluetooth devices (scanner) and pushes messages

6. Google Calendar's website is here: `http://www.google.com/calendar`. Our Google Calendar sensor uses the Google Data API to access Google calendar information. More information about Google Data is available here: `http://code.google.com/apis/gdata/`

addressed to users in range (sender). If it sees a user's phone, it records this event by entering a sighting into Construct. A sighting contains the details of the device seen, the IP address of the Bluetooth scanner and a port number on which it is listening.

Once Scatterbox has a message to be sent to a user it connects to the component where the user's phone was last sighted. The component verifies that this device is still in range and if so, pushes the message to the phone.

6.2. *Email Handler*

Scatterbox's email handler connects to the user's email server as an IMAP client. Throughout the day, it downloads all unread messages and extracts information from them that will be used by Scatterbox's reasoning component to determine their importance. This information includes the sender, the recipients, and the content of the subject and message body. If Scatterbox decides to forward a message to the user's phone, the email handler will flag the message as *read* on the server and will not process this message again. As a result, when the user checks their mail at the end of the day, the messages which have been forwarded are marked as read.

6.3. *Situational Reasoner*

The Situational Reasoner takes information about email in the user's inbox and context information from the Context Gatherer as input. It processes this information in order to decide whether or not to forward that email to the user at this time. This task has two parts, email classification and situation determination.

6.3.1. *Email Classification*

After the email handler has retrieved a new email and extracted its metadata, it sends this mail to the Email Classifier. Email classification is currently done in a relatively naïve manner, using only the sender and recipient fields of the message. As messages arrive, the sender and recipient email addresses are checked against a list of addresses that have already been encountered.

Sender email addresses will be in one of a number of sets, each of which has a "closeness" value relating that person to the user themselves. The first set contains the email addresses of people that are closest to the user (this set includes the user's own email addresses). The implication is that emails involving these addresses will be favoured above all others. A number of other sets follow, corresponding to lowering levels of subjective importance. The final set is for addresses of messages from strangers — people who have never sent messages through Scatterbox to the user before. Messages from these addresses will tend to be treated as unimportant by Scatterbox.

At the end of every day, the user is prompted to move the addresses from the strangers set into sets with higher priorities. The listings in the strangers set are ordered by the number of emails received from each address to facilitate this. Future implementations of Scatterbox will attempt to perform this task automatically using the feedback from the user in response to messages from those users.

A similar process is used for the recipients field of each message. If the message was sent directly to the user alone, the weight of the mail remains the same. If the mail was sent to more than one person, the importance of the mail is reduced with respect to the email address in the "To" field with the highest "closeness" to the user (addresses in the "CC" field are also added to the strangers set if they are newly encountered).

Scatterbox combines these values to assign each email with an importance weight. This will be used in combination with the situation determination algorithms to decide whether to send that email to the user.

6.3.2. *Situation Determination*

We define a situation in Scatterbox as being the measure of a user's interruptibility, or how likely a user will be content to receive a message at the current time. Once an email has been given an importance weight, the interruptibility of the user must be determined to decide whether Scatterbox should forward it. Scatterbox uses the available context from the Context Gatherer component to do this.

The interruptibility of a user is found by examining whether there is a calendar entry for the current time (if there is no entry, the user is set to a high level of interruptibility by default). If there is an event currently occurring, keywords in the title and description of the event are examined. Words such as "Busy" or "Do not disturb" decrease interruptibility and words such as "Free" or "Available" increase it. At this stage we ask our users to be proactive in filling out their calendar for the day.

Once the importance weight of the incoming mail has been determined and the interruptibility of the user is known, these two values are simply multiplied together. If the result goes above a given threshold, the message is forwarded to the user.

One special case we have made allowances for is when Bruce has meetings scheduled for today. As his calendar will also contain the email addresses of any users who are attending these meetings with him, we can give messages from these users a temporary boost in relevancy so that they are more likely to be forwarded, as they may contain information important to the meeting, or may be saying that the other person is going to be running late.

6.4. *Delivery*

It is important to deliver mail in a timely, reliable manner. If the user is in range of a registered Bluetooth-enabled computer which is running Construct, the message can

be pushed to them over Bluetooth. If the user is not in range of a Bluetooth device, a summary of the mail is sent to them via SMS.

Bluetooth was chosen over SMS as the default sending mechanism for a number of reasons:

– There is no monetary cost when using Bluetooth to send the message.

– Bluetooth messages can contain significantly more information than SMS, which are typically limited to 160 characters.

– Rejection of messages can be done conveniently by the user, whereas rejection over SMS would involve the sending of another message in reply.

However, using SMS to forward messages to the user has a number advantages over Bluetooth:

– Predictability — The implementation of Bluetooth protocols on phones can vary. For example, some Samsung phones will not allow pushed messages to arrive without first pairing with the sender. This is not the case for Nokia or Sony Ericsson. Since there is no standard way of dealing with incoming Bluetooth files, there can be no guarantee the message will deliver. SMS messages are sent over a "best effort delivery" protocol, and so are in general more likely to be received successfully.

– An SMS can reach any mobile phone where that there adequate mobile network coverage, unlike Bluetooth, which only extends as far as the nearest registered device, which are almost always indoors.

7. Evaluation

To evaluate Scatterbox's effectiveness in using context to determine which messages to forward, and the accuracy and usefulness of these messages, a number of user tests have been defined. 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 (Ho *et al.*, 2005). Therefore an evaluation of this system must account for the context that was used to support the decision to pass the user a message.

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. 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 at that time. Negative feedback to these messages sets them as true negatives (and thus appropriate behaviour) and positive feedback sets them 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.

The evaluation will be done in two steps. The first step involves members of our research group, who are working closely with the development team, acting as alpha testers, giving feedback regarding the punctuality and accuracy of message delivery.

This first stage has been completed successfully, with Scatterbox running reliably for all users for over a week. No filtering was done on emails, meaning every email a user received was forwarded to them. This strategy allowed us to judge when users were happy to receive useful emails and vice versa. The participants were asked to keep note of the situation they were in when they received a message, and whether the mail was useful. Over three hundred mails were forwarded. Of those messages, 20%–30% were deemed useful. The initial impressions confirm many of our theories, specifically regarding interruptability and the dynamics of email usage. At this early stage, we can draw a number of useful conclusions.

– 160 characters was generally thought to provide enough information for a user to determine the importance of a message.

– What would be considered a non-spam email may be considered a spam text message, and in general users were less tolerant of unimportant messages when received on their phone.

– Some of our participants began to deal with their normal email client differently. Since they had seen the message previews already, they only interacted with mails what they deemed important when they received them via SMS.

– Since the messages arrived in SMS format, participants could not reply to the email from their phone. This meant that they had to wait until they were using their email client before responding. The implication of this is that important emails could be forgotten about. This showed a clear need for some feedback mechanism to flag a message as important.

The full user trial will involve roll-out of Scatterbox to more than twenty people over a period of one week. Between the hours of 09:00 and 17:00, the users will be completely dependent on Scatterbox for receiving their email. During these times they will keep a diary, recording where they were, who they were with, and what situations they were in at the time when they received each message (i.e., when they were interrupted). This data will then be compared to the situations that Scatterbox had determined when deciding to send each mail to test its performance. At the end of each day, participants will be allowed to check their mail using their own traditional email client. At this time they will be asked to take note of false positives and false negatives (this can be done by filing the messages using tags or folders, depending on their client). The feedback provided will be used to improve the message forwarding process. Initially this will be completed by hand — changing priorities of correspondents and changing the keywords and locations which define situations — but future versions of Scatterbox will automate this process.

We will evaluate Scatterbox by analysing the feedback from each day. We will examine the number of mails received versus the number of false negatives and false positives per day. This analysis will enable us to determine how well situations were

determined and what caused incorrect determinations. We will also be able to determine how well the system learns over the course of the user trial by quantifying the reduction of false positives and false negatives.

Further to performing this live evaluation, we will collect and publish a data set of real context and behaviours over the course of our evaluation and use this to perform further, offline evaluations. By gathering all relevant context features, along 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 (Liu *et al.*, 1998). 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.

8. Conclusions and Future Work

We present an application, 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 emails 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 deals with the problem of determining a user's situation using numerous heterogeneous sensors in a context-aware environment. It does this by modelling context using ontologies so that heterogeneous sensors and applications can communicate with one another. This extensible way of modelling contextual data can the be used by any number of context-aware systems. Scatterbox accesses this context information using the Construct pervasive computing framework, which has been used in the past for context reasoning in many other application areas, including assisted living (Coyle *et al.*, 2007) and recommender systems (Coyle *et al.*, 2006a).

An evaluation is currently underway which will allow us to test our techniques for situation determinism and email classification. This evaluation allows us to examine the utility of our reasoning algorithms in real-time context-aware systems and gives an indication of the best approach to improve its utility. The evaluation will also allow us to find the most important features to determine "interruptibility" and email classification and as a result, implement more sophisticated algorithms and improve the performance of Scatterbox.

Currently, the user is required to provide Scatterbox with feedback manually at the end of each day. Automatic feedback and learning techniques will be used to allow all feedback to occur with minimal user intervention. This will be achieved by automatically recalculating the weights of correspondents and situations based on the acceptance/rejection of pushed Bluetooth messages. Accuracy of message clasifica-

tion can also be improved by analysing email usage patterns, specifically how quickly a user responds to a mail or whether a mail is deleted without being read.

Acknowledgements

We are grateful for the detailed and helpful feedback received from external reviewers on earlier drafts of this paper. This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062, 04/RPI/1544, 03/CE2/I303-1, and by an EMBARK Scholarship from the Irish Research Council in Science, Engineering and Technology.

9. References

Chen H., Finin T., Joshi A., " A context broker for building smart meeting rooms", *The Knowledge Representation and Ontology for Autonomous Systems Symposium*, AAAI Spring Symposium, p. 53-60, March, 2004.

Coyle L., Balfe E., Stevenson G., Neely S., Dobson S., Nixon P., Smyth B., " Supplementing Case-based Recommenders with Context Data", *Procs of the 1st Workshop on Case-Based Reasoning and Context Awareness at ECCBR 2006, CEUR Workshop Proceedings*, 2006a.

Coyle L., Neely S., Rey G., Stevenson G., Sullivan M., Dobson S., Nixon P., " Sensor fusion-based middleware for assisted living", *Proc. of 1st International Conference On Smart homes & health Telematics (ICOST'2006)*, IOS Press, p. 281-288, 2006b.

Coyle L., Neely S., Stevenson G., Sullivan M., Dobson S., Nixon P., " Middleware Sensor Fusion for Assisted Living", *International Journal of Assistive Robotics and Mechatronics (IJARM)*, vol. 8, p. 53-60, June, 2007.

Dey A., " Understanding and using context", *Personal and Ubiquitous Computing*, vol. 5, p. 4-7, 2001.

Dobson S., Coyle L., Nixon P., " Hybridising events and knowledge as a basis for building autonomic systems", *IEEE TCAAS Letters*, 2008. To appear.

Henricksen K., A Framework for Context-Aware Pervasive Computing Applications, PhD thesis, The School of Information Technology and Electrical Engineering, University of Queensland, September, 2003.

Hightower J., Brumitt B., Borriello G., " The location stack: a layered model for location in ubiquitous computing", *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications,*p. 22-28, 2002.

Ho J., Intille S. S., " Using context-aware computing to reduce the perceived burden of interruptions from mobile devices", *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, New York, NY, USA, p. 909-918, 2005.

Knox S., Clear A. K., Shannon R., Coyle L., Dobson S., Quigley A., Nixon P., " Towards Scatterbox: a Context-Aware Message Forwarding Platform", *Fourth International Workshop on Modeling and Reasoning in Context (MRC 2007)*, p. 13-24, August, 2007.

Liu H., Motoda H., *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Miller N., Judd G., Hengartner U., Gandon F., Steenkiste P., Meng I.-H., Feng M.-W., Sadeh N., " Context-Aware computing using a shared contextual information service", *Pervasive '04, "Hot Spots"*, Vienna, April, 2004.

Nakanishi Y., Tsuji T., Ohyama M., Hakozaki K., " Context Aware Messaging Service: A Dynamical Messaging Delivery using Location Information and Schedule Information", *Personal Ubiquitous Comput.*, vol. 4, p. 221-224, 2000.

Oulasvirta A., " The fragmentation of attention in mobile interaction, and what to do with it", *interactions*, vol. 12, p. 16-18, 2005.

Padovitz A., Loke S. W., Zaslavsky A., Bartolini C., Burg B., " An Approach to Data Fusion for Context Awareness", *Fifth International Conference on Modelling and Using Context (CONTEXT)*, Springer-Verlag, Paris, France, p. 353-367, 2005.

Pascoe J., " The Stick-e Note Architecture: Extending the Interface Beyond the User", *IUI '97: Proceedings of the 2nd international conference on intelligent user interfaces*, ACM Press, New York, NY, USA, p. 261-264, 1997.

Rialle V., Lauvernay N., Franco A., Piquard J.-F., Couturier P., " A smart room for hospitalised elderly people essay of modelling and first steps of an experiment", *Technology and Health Care*, vol. 5, p. 343-357, January, 1998.

Robinson S., " Global Handset Specification Database: Feb 2008", Feb, 2008, http://www.strategyanalytics.net.

Stevenson G., Coyle L., Neely S., Dobson S., Nixon P., " ConStruct — A Decentralised Context Infrastructure for Ubiquitous Computing Environments", *IT&T Annual Conference, Cork Institute of Technology, Ireland*, 2005.

W3C, " Resource Description Framework (RDF)", , http://www.w3.org/RDF/, 01, 2004.

Wang X. H., Zhang D. Q., Gu T., Pung H. K., " Ontology Based Context Modeling and Reasoning using OWL", *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, Washington, DC, USA, p. 18, 2004.

Weiser M., " The Computer for the 21st Century", *Scientific American*, vol. 265, p. 94-104, September, 1991.

Whittaker S., Bellotti V., Gwizdka J., " Email in personal information management", *Commun. ACM*, vol. 49, p. 68-73, 2006.

Yau S. S., Huang D., Gong H., Yao Y., " Support for situation awareness in trustworthy ubiquitous computing application software", *Software: Practice and Experience*, vol. 36, p. 893-921, July, 2006.

Ye J., Coyle L., Dobson S., Nixon P., " Ontology-based Models in Pervasive Computing Systems", *The Knowledge Engineering Review*, vol. 22, p. 315-347, 2007.

Zhou Y., Mulekar M. S., Nerellapalli P., " Adaptive Spam Filtering Using Dynamic Feature Space", *ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, Washington, DC, USA, p. 302-309, 2005.