Context-awareness: some compelling applications

Peter Brown; University of Exeter; *pjbrown@ex.ac.uk*
Winslow Burleston; MIT Media Lab; *win@media.mit.edu*
Mik Lamming; Xerox Research Centre Europe; *mik.lamming@xrce.xerox.com*
Odd-Wiking Rahlff; SINTEF; *Odd-Wiking.Rahlff@informatics.sintef.no*
Guy Romano; Motorola Labs, Illinois; romano@rsch.comm.mot.com
Jean Scholtz; DARPA; *jscholtz@darpa.mil*
Dave Snowdon; Xerox Research Centre Europe; *snowdon@xrce.xerox.com*

ABSTRACT

It is claimed that, to become credible in the marketplace, context-awareness needs a killer application. Killer applications are rare in most fields, but we believe that there are context-aware applications that can at least merit the adjective 'compelling'. We present a list of six types of context-aware application that we think are either compelling now or soon will be. In order to present our list, we first discuss a key component of the necessary infrastructure: how context is captured and maintained.

KEYWORDS: context-aware, mobile, killer application, context memory.

Introduction

Like many emerging technologies, context-awareness has attracted a wide spectrum of claims as to its likely future impact. As an example on the positive side, the annual report of Hewlett-Packard to shareholders [1] holds context-awareness to be a key future technology for the company, thus showing that business executives as well as researchers are now taking notice. As a further example, some staff at Reuters [2] see the use of context as the most promising way of tackling the increasing problems of information overload. As additional positive evidence, there are plenty of successful applications just over the border of what are considered to be true context-aware applications: these include GPS applications in farming, in construction and in the military. On the negative side, pessimists say that, to bridge the gap between laboratory and marketplace, context-awareness needs a killer application, and they see no sign of one.

The purpose of this paper is to look at potential context-aware applications. The applications may not be universally accepted as killer ones, but, even if not, they hopefully offer some credibility. We try to describe applications in generic terms, though we give specific examples for the purpose of explanation. We also try to take a catholic view of the purpose of an application: it may range from an economic one of improving communications in an office to a personal one of enriching family life. The paper presents six types of context-aware application. We emphasize that it is *not* our aim to produce a taxonomy: there are plenty of applications that do not fit any of the generic applications we discuss.

There will be no pure context-aware applications, since context-awareness on its own is not something a user needs. Instead many see context-awareness as an enabling technology to help other applications perform better. Thus in the applications that we describe context-awareness may be a relatively small part of the whole application, though it will be a essential part.

Past work

Numerous context-aware projects have been undertaken, albeit mostly with the label ‘prototype’ or ‘pilot’. We would like to highlight one of these, because it has a generic quality.

This is the work at Xerox PARC in the early nineties, described in a survey article by Schilit *et al* [3]. The article has since become a seminal paper for researchers in the field. The article presented the earliest taxonomy of context-aware applications, and the Xerox researchers produced working prototypes of applications in each class of their taxonomy; these prototypes used the PARCTab, a device that combines the properties of active badge and PDA. The taxonomy consists of four classes. The first is *proximate selection*, which is concerned with automatically changing interfaces so that the natural defaults reflect the user’s current context. The second is automatic reconfiguration according to context. The third and fourth are concerned with context information/commands and with context-triggered actions, respectively; a majority of present-day applications come within these two classes. The four classes are centred on the form of the user interface of the application, rather the way context is used.

Arguably this work was ahead of its time, especially as the hardware then available was bulky, expensive and far from ubiquitous.

Capturing and maintaining contexts

Our six types of application, like others, require a basic infrastructure for maintaining and manipulating contexts. There are two elements to this: (a) the capture of the current context and (b) context memory.

The current context is typically captured partly from sensors, partly from existing information (diaries, todo lists, weather forecasts, share price feeds), partly perhaps from user and task models, partly from the state of the user’s computing equipment and the user’s interaction with that equipment, and partly from explicit settings by the user. As an example of the last of these the user may set their current interests/needs/state (‘I want a restaurant’, ‘I am on holiday’, ‘I want to know about historical buildings’). The current context may combine the physical and the virtual, and may switch contextual fields between the two: for example the user might want to pretend sometimes that they are at a different time and place from their physical one.

Sensors tend to be at a low level; for example sensors in an office may detect whether doors are open or chairs are in use. Humans are interested in higher-level contextual elements, such as whether people are busy or whether a meeting is taking place. Past work on synthesising high-level events from low-level ones, such as Pepys [4], may well find increasing application in the future. As this synthesis becomes cleverer at detecting the user’s context, the amount the user will need to tell the system will decrease. For example the synthesis might be able to deduce from a number of low-level details that the user is on holiday. Past experience, however, shows that such deductions can often be too clever by half, and we will need to find out where the practical boundaries lie.

Synthesis is also important in making probabilistic judgements based on sensor readings. This could apply when several alternative sensors relate to the same physical quantity: location may potentially come from GPS, from the user’s cellphone and from image recognition of camera pictures. Any of these can give unreliable readings and any can be non-operational, but the application needs to be provided with the most probable location based on the data derived from the three types of sensor.

As a final comment on synthesis, the synthesized context may itself be valuable information that can be presented to the user: this can apply especially when context is rapidly changing as in emergencies, battlefield situations and financial upheavals.

We need aggregation mechanisms for contexts. Context may be recorded at several hierarchical levels, e.g. personal, project, group and organization. Clearly all but the first of these will be shared, and equally clearly there is a need for standards for representing and exchanging contexts. In addition to hierarchical aggregates we may require other aggregates, such as personal trails.

The second fundamental requirement, in addition to capturing and representing the current context, is what we call *context memory*. (This may also be called ‘history’ or ‘logging’, but we prefer the term ‘memory’ as it implies a degree of organization and intelligent recall.) At the lowest level context memory is important in applications where the context values are not *per se* important, but change is important, e.g. someone moving. To detect change, the application obviously has to memorise past values. At a higher level, context memory goes hand in hand with synthesis: analyzing past contexts and in particular analyzing what is changing and what is not, can lead to intuitions (‘The user has been viewing web pages about old buildings for the past then minutes, therefore ... ’, ‘The user has been composing messages to work colleagues, therefore ... ’).

For many applications context memory needs to straddle the past, the present and the future. Some contextual events, at the time they are captured, relate to intended future events (e.g. diary entries, todo lists, forecasts). Normally such events are at the human task level rather than that of low-level sensors, and are thus more valuable. As time proceeds, future events become past events in the context memory, especially if other contextual readings at the time of the event indicate that the event really did happen.

We have discussed the topic of recording context at some length because, as has been widely observed, large amounts of infrastructure covering context will be needed before the potential applications can realise their full potential. For more discussion of some of the issues see Salber *et al* [5].

Privacy

Privacy issues underlie most context-aware applications: the more the application knows, the better it can perform, but the more it knows, the more invasive it is of privacy. Privacy concerns have killed many potential applications already, and will continue to do so. Unfortunately debates on privacy tend to generate heat rather than light. This is not a paper about privacy, but we will give a few principles here that might help make an application acceptable:

- each user must have control over the information about themselves, and must be free at any time to change the accessibility of the information or even the information itself. In the latter case they can tell lies: thus if there is an automatic record of how much time they spend at their desk, they are free to modify this data so that to the world outside it appears that they work 16 hours a day; nevertheless for their own personal applications, the true reading may be left alone -- perhaps it might be used by their personal health monitor context-aware application.
- the ways in which the application uses information must be clear to the users. (This is a challenge since most real applications will use information in subtle ways.)

- the user can only retrieve information that they were aware of at the time. Thus a user should be able to retrieve information about who was present at a meeting while they were present, but not to find out who came and went after they had left.

For discussions of the ethical issues underlying privacy see [6, 7].

Rich capture, rich variety of behaviours

As humans we have senses of sight, hearing, touch, smell, etc. Current contextual sensors can only capture a tiny part of these. They are, however, improving -- particularly small cameras -- and we will see a much richer capture of context in the future, covering multiple senses and multiple media.

If the captured material is rich it becomes useful in a wide variety of ways. This leads into an issue that most workers on context-aware systems have found: there is a blurred and often continually changing boundary between what an application treats as context and the other objects manipulated by the application. Thus a picture of what the user is currently viewing may be used as a contextual element (for example if there is later a similar picture, this may lead to automatic retrieval of information that related to the time when the user was previously viewing what is apparently the same scene). Alternatively the picture may be an object for retrieval, and this retrieval may be determined by other contextual parameters, for example GPS sensors detecting the user being at the same place some while later.

Overall, applications may need to show a rich variety of behaviours in how they deliver the fruits of their context-aware behaviour: they may present information to the user and/or change interfaces and/or perform actions. Often the form of delivery will itself depend on the context: whether the user is in a meeting, what languages the user understands, etc.

We now move on to discuss the six types of application.

Type 1: proactive triggering

Proactive triggering applications are those that record the user's current context and trigger information to be presented or actions to be performed that appear to be relevant. The information might be specially designated by its author to trigger in certain circumstances. Thus in the archetypical context-aware application, the tourist guide, the author may specify that information about a garden in spring should be triggered when the user is near the garden in spring during the garden's opening hours. Alternatively the application might detect what past contexts appear to match the present one, and trigger information that was automatically captured then. Examples are:

- 'Your grandfather was here in 1999, and he saw this view ... '.
- 'When you last used this photocopier to produce slides you hit the following buttons'. Better, the application could pop up a control panel on the user's hand-held device, and this could be used to perform the appropriate photocopier actions -- with the previous sequence of actions as the default.
- 'X is present at this conference: you should talk to her about your interest in ... '.
- 'Y [who is an expert or mentor] suggests that in this situation you ... '. More generally, this example leads into persuasive and assistive technologies [8].

In applications involving presentation of information to the user, the pieces of information might be ranked according to how well they relate to the user's current context -- an example of Schilit's proximate selection. Thus a list of nearby restaurants might be ranked inter alia by their closeness to the user.

Proactive triggering can be extended to cover filtering out information: a negative proactive trigger can *stop* information being delivered.

Of the six types of application we consider, this is the one that is best covered by existing applications. These applications may not be compelling at the moment; however we believe that if, taking advantage of the improving infrastructure, some of the above suggestions can be effectively delivered, then the applications will become compelling.

Type 2: streamlining interaction

Abowd and Dey [9] observe: *When humans talk with humans, they are able to use implicit situational information, or context, to increase the conversation bandwidth. Unfortunately, this ability to convey ideas does not transfer well to humans interacting with computers.*

The applications we wish to discuss are a hybrid between human-to-human communication and human-to-computer interaction: humans are in face to face contact, but are using computers (perhaps PDAs or the like) to aid their communication. In particular, when humans talk with each other, they also want to exchange documents as well. In order to do this we need to have the underlying hardware for the computing devices to communicate with each other and with devices such as printers. Fortunately such hardware, e.g. that which supports beaming, is becoming commonplace. The bigger challenge is to create suitable interfaces, applicable to a tiny screen, which allow humans to streamline the interchange of documents and other materials. The Satchel project [10] is an example of an application that is designed to achieve this. The key is hugely simplifying the computer interface by taking advantage of the context: i.e. knowing what the user is likely to want to do, and what facilities are available nearby. For example, Satchel has one menu-button called 'services', which produces of menu of what services are currently available.

The ideas behind Satchel, which is aimed at document exchange, can be generalised to streamline human interfaces by allowing more general exchange and capture.

Type 3: memory for past events

Probably the most ambitious context-aware application to date is the *memory prosthesis* work at Xerox European Research Centre in Cambridge [11], and its realisation in the Forget-me-not prototype [12, 13]. This work aimed to capture events in the user's working life -- the prototype was based on office workers, working in an environment rich in sensors. The context of each event is captured automatically at the time it happens: the room, the other people present, the slides being displayed at a meeting, the current weather, the notes that the user made on their PDA, etc. Context is then used for retrieval. Thus a user might say: 'Find me the notes I made at a meeting about a year ago in the Conference Room. I remember Bill and John were there, and it was raining outside'.

We all have fallible memories, a fallibility that further increases after middle age. Thus the economic potential, both to individuals and to organizations, of an application that helps resolve these fallibilities is immense -- even if the application only succeeds with a fraction of them. Moreover there is even greater potential for applications designed for the memory-impaired.

An interesting facet of the memory prosthesis is that users can streamline some of their actions if they have confidence in the prosthesis. Thus with our example of retrieving the notes made at a meeting, there is no need for the user to bother to save the notes in an explicit file (whose name they might forget anyway). Instead the context memory acts as a filing system, with the context in which a file was created, rather than a filename, as the key for retrieval.

More recent, separate, research has proposed a new device called the Factoid [14], which can be used to capture continuously facts about the user's life. The data captured from this could be the bedrock of a memory prosthesis.

A project related to the memory prosthesis is the Remembrance Agent of Rhodes [15]. This is 'a continuously running proactive memory aid that uses the physical context of a wearable computer to provide notes that might be relevant in that context'. For example if the user is detected as taking notes in a meeting, the agent can automatically provide notes of previous meetings of the same group, relevant papers and e-mails, etc. Each suggestion is initially presented as a single line on the display, but can be expanded. The Remembrance Agent in fact straddles our Type 3 and our Type 1.

Another potential application of this type is dealing with the interruptions that we all suffer throughout our working day: after an interruption has been dealt with, a user wants to retrieve what their context was at the time of the interruption, so that they can more easily resume their previous activity.

Overall we believe that the ideas for Type 3 applications have already been explored to the extent that, when the required infrastructure is there, highly successful applications can be built.

Type 4: reminders for future contexts

Consider applications where the user (or a third party) sets some information with a future context attached to it, so that when this future context arises the information is triggered. Examples are:

- 'When I next meet X, tell him ... '.
- 'When next at a grocery store, buy ... '.
- The features found in the WatchMinder [16], a watch designed to give reminders, especially to people with impairments. The context used is just time.
- 'There will be a meeting of the project group at 4 pm in Room 219 on Friday'.

The last of these is rather different from the others: it is saying that the user should be in a certain future context. Thus it is stating a required context. Indeed the first two could be rephrased to introduce an element of requirement, e.g. 'When next in a grocery store buy ... . *This must be done before the reception on Saturday*'.

Without the required context, these examples are logically of Type 1: proactive triggering. When there is a required context, however, the application should behave in a different way, and this is why we have made it a separate type. Clearly one simple action a Type 4 application might do is to remind the user (by beeping, or whatever) when the required time is approaching. The timing and nature of this reminder can take account of the user's current context: for example if the meeting is at 4 pm and if he is in a different building, ten minutes' walk away, the reminder might be given at 3.45 pm.

Type 5: optimizing patterns of behaviour

There are many applications where capture of context is an end in itself: this applies, for example, to fieldwork [17], which might range from recording animal behaviour in a game reserve to helping archaeologists record the results of a dig. In these applications the context can be captured partly via sensors (position, weather, time, etc.) and partly by the user (e.g. via a form on a PDA screen). These applications are already successful, and have made collection of data more reliable and less labour intensive.

The type of application we have in mind here goes beyond this by analysing patterns of behaviour. Examples of the resultant suggestions to a user that may come out of the analysis are:

- 'You normally catch tram 12, and then tram 7; did you know that tram 14 did the journey directly?'.
- 'It is a good time to eat as restaurants are not crowded yet'.
- 'I suggest you go to exhibit 23 now, as it is close and uncrowded.' (The analysis behind this suggestion may just be based on current state, e.g. number of visitors at each exhibit, or it may also be based on historical patterns, and on the intentions of the other people currently in the exhibition.) See [18] for an existing application of this type.
- 'X appears to be working on the same task as you. She can be contacted at ... '.
- Similar applications involving tracking of vehicle fleets or asset management in general.

Type 6: sharing experiences

Recently there has been a growth in collaborative applications where users -- normally remote from one another -- share an experience, ranging from shopping by following a set of shared web links to simulating the experience of being co-present with a friend. These applications are based on sharing a common context: the context of one user (or perhaps a combination of many users) is captured and other users are presented with a simulation of this context.

The basic element needed to implement shared experience is exchange of context among the sharers. Sometimes the identity of these sharers will be fixed in advance, and sometimes it will be discovered on-the-fly by comparing contexts and trying to identify suitable sets of sharers. This type of application above all others therefore needs a standard way to represent contextual data, and a context discovery protocol.

Conclusions

Many -- perhaps most -- technological ideas were developed before the time at which widespread usage became feasible. Babbage is credited with inventing computers 150 years ago. As a software example, hypertext was invented by Bush in 1945, and several good working systems were available in 1970, yet it was the mid-eighties before the acceptance of hypertext became significant, and a decade later, with the success of the web, before we had killer applications. Of course for many technological ideas the time of widespread usage never comes.

We believe the day of widespread usage of context-awareness will indeed arrive, and we trust that our generic examples here will help to make this claim more compelling. The problems of inadequate hardware infrastructure, which have dogged past applications, are fast disappearing. More serious now is the lack of a context infrastructure of the kind we have outlined, but we believe that this too will gradually evolve to provide the platform for the applications we have described.

Acknowledgements

This paper grew out of a working group within CHI 2000 workshop W11, organised by Steve Armstrong, Anind Dey and David Morse.

Peter Brown would like to thank the Leverhulme Trust for support.

References

1. Hewlett-Packard Company, *Annual report to shareholders*, 2000.
2. Reuters PLC, Personal communications from staff, 2000.
3. W. N. Schilit, N. I. Adams and R. Want, ‘Context-aware computing applications’, *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85-90, Santa Cruz, California, IEEE Computer Society Press, 1994,
4. William M. Newman, Margery A. Eldridge and Michael G. Lamming, ‘Pepys: generating autobiographies by automatic tracking’, *Proc. ECSCW ‘91*, Amsterdam, September 1991.
5. D. Salber, A.K. Dey and G.D. Abowd, ‘The context toolkit: aiding the development of context-enabled applications’, *Proc. Proceedings of CHI’99*, Pittsburgh, PA, ACM Press, 1999.
6. C. Larson, Chapter 2, ‘Perspectives on ethics in persuasion’, pp. 27-54 in *Persuasion: Reception and responsibility*, 7th ed. Belmont, CA: Wadsworth Publishing, 1995.
7. D. Berdichevsky and E. Neunschwander (1999). ‘Towards an ethics of persuasive technology ’, *Communications of the ACM* **42** (5), pp. 51-8, 1999.
8. W. Burleston, *Peak performance: a technological approach*, http://www.media.mit.edu/context/win/peakperformance.html, MIT Media Lab, 1999.
9. G.C. Abowd, A.K. Dey, P.J. Brown, N. Davies. M. Smith and P. Steggles, ‘Towards a better understanding of context and context-awareness’, (panel statements), *Handheld and Ubiquitous Computing*, (H.-W. Gellersen, Ed.), Springer, Berlin, pp. 304-307, 1999.
10. M.G. Lamming, M. Eldridge, M. Flynn, C. Jones and D. Pendlebury, ‘Satchel: providing access to any document, any time, anywhere’, to be published in ToCHI, 2000.
11. M. G. Lamming, P. J. Brown, K. Carter, M. Eldridge, M. Flynn, P. Robinson and A. Sellen, ‘The design of a human memory prosthesis’, *Computer Journal* **37** (3), pp. 153-163, 1994.
12. M. Flynn and M.G. Lamming, ‘Forget-me-not: intimate computing in support of human memory’, *Proceedings of FRIEND 94 International Symposium on Human Interface*, pp. 1-8, Tokyo, Japan, 1994.
13. M. Flynn and M.G. Lamming, *XRCE Cambridge Technical Report EPC-1994-103*, http://www.cambridge.xrce.xerox.com/publications/trs/EPC-1994-103.html, 1994.
14. Bob Mayo, *Factoid*, http://www.research.digital.com/wrl/projects/Factoid/index.html, Compaq Western Research Lab., 2000.
15. B.J. Rhodes ‘The wearable remembrance agent: a system for augmented memory’, *Personal Technologies*, **1**, pp. 218-224, 1997.

16. WatchMinder, http://www.watchminder.com, 2000.
17. J. Pascoe, D.R. Morse and N.S. Ryan, ‘Developing personal technology in the field’, to be published in ToCHI, 2000.
18. G.G. Abowd, C.G. Atkeson, J. Hong, S. Long, R. Kooper and M. Pinkerton, ‘Cyberguide: a mobile context-aware tour guide’, *ACM Wireless Networks*, **3**, pp. 421-433, 1997.