Architectural Implications for Context Adaptive Smart Spaces

Paddy Nixon[1,2] Simon Dobson[2] Sotirios Terzis[1] Feng Wang[1]

[1]*Global and Pervasive Computing Group*
Department of Computer and Information Sciences
TheUniversity of Strathclyde
GLASGOW, Scotland.
{Paddy.Nixon, Feng.Wang}@cis.strath.ac.uk

[2]*Aurium*
Clifton House,
Lower Fitzwilliam Street, Dublin 2
Ireland
Simon.Dobson@aurium.net

Abstract: *Buildings and spaces are complex entities containing complex social structures and interactions. A smart space is a composite of the users that inhabit it, the IT infrastructure that supports it, and the sensors and appliances that service it. Rather than separating the IT from the buildings and from the appliances that inhabit them and treating them as separate systems, pervasive computing combines them and allows them to interact. We outline a reactive context architecture that supports this vision of integrated smart spaces and explore some implications for building large-scale pervasive systems*

1. Introduction

Buildings and spaces are complex entities containing complex social structures and interactions. Organisations increasingly look to their buildings to fulfil a function more complex than simple accommodation: there is an increased awareness that the "logic of space" can be used to aid organisational goals such as increasing communication. Architects, of course, have long appreciated these issues and applied them to create structures that can be simultaneously beautiful and highly functional[27].

Buildings also contain increasing amounts of computing infrastructure ranging from cabling, through sensors such as burglar and fire alarms, up to environmental and other controls. The decreasing cost of devices and the increased desire for monitoring and control has led to a number of companies to develop building control systems and networks, allowing a central operator to observe the building and affect aspects of it.

It is clear, however, that a largely "passive" building will not always be suitable for its changing uses and users. Someone visiting a building for the first time often becomes lost or disoriented. Disabled users can have difficulty navigating around buildings in which not all routes may be accessible to them. Despite the availability of IT services *within* the building, users cannot interact *with* the building itself.

By contrast a *smart building* or *smart space* - a user environment which is reactive to its surroundings and occupants - can be viewed as a space that has been designed and constructed with user interaction in mind.

Contextual use of space

Rather than separate IT from buildings and from the appliances that inhabit them, and treating them as separate systems, why not combine them and allow them to interact? Why not provide a building that is sensitive to its users, their locations, interactions and tasks - the *context* within which they use the building's spaces and service - and can provide seamless, autonomous support for their activities? We believe in computer applications that react to what users are doing in the real world, and provide the *only* source of tools and expertise to making this happen.

In this paper we outline a reactive *context* engine that supports this vision of integrated smart spaces.

What is Context?

Historically, the use of "context" grew from roots in linguistics [6]. The term was first extended from implying inference from surrounding text to mean a framework for communication based on shared experience [7,8]. The importance of a symbolic structure for understanding was embraced in other fields such as [9,10,11,12,13,14,15,16,17] and subsequently developed from a purely syntactic or symbolic basis to incorporate elements of action, interaction and perception.

More recently, in the setting of mobile computing, "context aware" was at first defined by example, with an emphasis on location, identity and spatial relationships [18,19,20,21,22]. This has since been elaborated to incorporate more general elements of the environment or situation. Such definitions are, however, difficult to apply operationally and modern definitions [23,24] generalize the term to cover "any information that can be used to characterize situation".

Current work in the field addresses issues including:

0-7803-7686-2/02/$17.00 © 2002 IEEE

?? developing new technologies and infrastructure elements, such as sensors, middleware, communication infrastructures to support the capture, storage, management and use of context.
?? increasing our understanding of form, structure and representation of context;
?? increasing our understanding of the societal impact of these new technologies and approaches and directing their application;

A more detailed retrospective of the academic history of context can be found in [25,26].

Tiered architectures

Modern enterprise systems architecture is built around a "tiered" model. There is much discussion as to exactly how many tiers should be used in an application, the details of what each tier provides, etc. However, the common theme is that each tier represents a *level of abstraction* in the design of the system, allowing developers to focus their attention at one level (for example business logic) without having to worry about the details of other levels (such as storage management). For this approach to function, each tier needs to export an agreed set of interfaces exposing its functions to its neighbouring tiers. The implementation of these interface functions may be changed freely, and these changes will not affect other tiers accessing them through the interface. A typical model is the five tier architecture shown below:

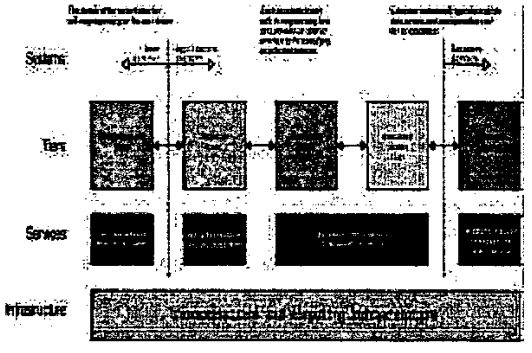

Figure1: A five-tier architecture

This industry accepted architectural style has evolved, most recently in areas such as webservices, because it supports a stable and appropriately partitioned design approach. However, consider this approach from the perspective of providing dynamically adaptive systems which are:

?? User centred – configured on the fly for the user;
?? Supporting dynamic aspects;
?? Respnsive to the changing environmental characteristics; and
?? Responsive to the different networked appliances in the proximity.

Such systems must take account of changing and varying contextual information. In the rest of this paper we consider the industrial needs for architectures that support context as a core concept.

Adding context

As organisations deliver context-enhanced applications to their users, context services will be integrated into the tiered, web-service-based enterprise architectures that are becoming recognised as best practice. Imagine that you can monitor your users physical activity and location and then tailor the response of the local appliances to that users needs. This is what context can do for a system. Context enhancement brings together a collection of information streams - location, diaries, preferences, time, appliances characteristics, access policies and situation - the context - and reacts to combinations of this information based on a simple scripting interface. It is true to say that most context solutions can be achieved by developing point solutions: however these become complex very quickly. In this paper we outline a single platform based development solution that can be used to transform software components and applications into context aware solutions.

The impact of context on architecture

The first question to ask is: how does the use of context affect the *purpose* the architecture is intended to serve?
The most obvious impacts of context are in the front tiers - presentation and session. An application may wish to change its presentation based on a user's context, for example by moving automatically from an instant message to an SMS when a user leaves their office.
However, we may also see contextual changes on other tiers. For example, some processes may be intrinsically "easier" in some contexts than others, and could be streamlined. Alternatively a system might adapt the processes available in a session to avoid those that would be inappropriate for whatever reason, or provide monitor service delivery and react to potentially costly breaches in service-level agreements.
This cross-tier impact is what differentiates full-on contextual enhancement from simple location-based services, device transcoding or personalisation. A simple presentation tier add-on such as transcoding, for example, may allow applications to target the user's device but

cannot capture and express the business implications of this choice - such as a reduction in security or attention, or the reduction in detail in the services being provided. A context-enhanced application, by contrast, can draw application-level implications from the low-level information about devices, allowing adaptation across the whole application. This removes context enhancement from the "gimmick" category and moves it to the status of an infrastructural service with enterprise impact.

Context: conditioning the layers

In general, one may view context as conditioning the behaviour of the tier in an architecture. That is to say, the way the tier provides its functions will be affected by the context in which those functions will be used.

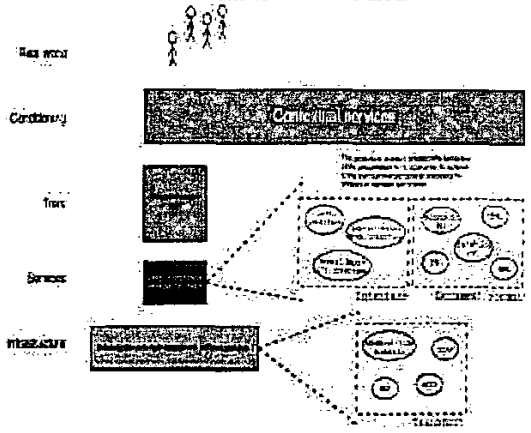

Figure 2: Context Presentation

The effect of context is to inform the way in which tiers dispatch their exported functions to the objects and services used to provide them. For example, contextualising the presentation tier above may cause a single user interface function (such as "show the user this alert") to be implemented differently whether the user is using a workstation or a small device, or in one location rather than another.

However, context can affect more than simply presentation, and it is here that the differences between context and simple location services or user preferences becomes apparent. An interface might change because of device or location - but also because of task, or the presence of other people, or some other high-level trigger. By maintaining a contextual model of users, applications can leverage contextual triggers across their entire operation rather than simply as add-on personalisation or location adaptation in the front tiers. The model is uniform and can be used, and make use of, elements deriving from anywhere in the architecture, not simply from user interface cues.

Integrating contextual services into a tiered architecture

The philosophy behind tiered architectures is to separate concerns in a system into different levels of abstraction and then encapsulate each level behind its own interface within its own distributed service. There is a degree of "linearity" implicit in the approach, in that tiers interact with their neighbours and do not "jump" to use tiers arbitrarily. This preserves the abstraction boundaries.

Contextual modeling has an end-to-end impact, however, and so is not a "neighbour" of *any* tier. For maximum effect a contextual model should accept information from *anywhere* and be used *everywhere*.

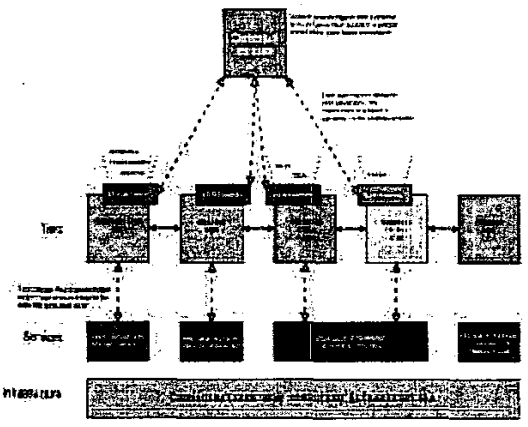

Figure 3: Integrating Context

One way to resolve this is to situate context modeling outside the normal tier structure. This does not violate the attractive properties of the tiered approach, as the context model can present a uniform, well-encapsulated interface to every tier.

The model takes input from the *environment* of each tier affected. For the presentation tier, this might include the user's device, preferences and location; for the business logic tier their tasks and roles within the application. It unifies these environmental factors into a model of the systems users' that may then be used to inform the selection of services within each tier.

Implications for tier design

This approach - providing a uniform model accessible across the architecture - does have some implications for tier interface design and function selection.

The most important factor is that context forces abstraction upon tiers. This is most obvious in the presentation tier: if the details of the interface can be changed by context, then the tier interface can only expose abstract functions rather than detailed access to the individual component objects that might be used. The reason for this is simple: if the details are exposed and then change, all tiers accessing those detailed functions are impacted.

More subtly, the tier cannot allow any assumptions about interfaces to propagate across its interface. This is sometimes more difficult. Consider, for example, the case where the application wants to "alert" the user to some event. Not all interfaces have obvious alert capabilities: HTML pages, for example, are not typically "pushed" at the client. Providing the alert function may have implications for the design of the exposed functions in the interface.

Conclusion

This paper sets out to provide a guide to adding context-enhanced mobile services to existing IT architectures, highlighting the importance of such an architecture for smart spaces.

We identify that context affects architecture by providing a uniform and well-founded framework within which to control and adapt the behaviour of a system to changing user circumstances. There are implications in the design of tiers and their interfaces needed for these benefits to be fully realised, but these changes reflect good software engineering practices and have other benefits anyway.

Many development tools provide the basics of context, mainly focused on the presentation tier of an architecture. However, the major benefits accrue from the end-to-end use of context throughout an enterprise architecture. The architecture described in this paper directly addresses this end-to-end problem.

References

[1] Software Process Modeling and Technology, edited by A. Finkelstein, J. Kramer and B. Nuseibeh, Research Studies Press, John Wiley and Sons Inc, 1994.

[2] J. Estublier, P.Y.Cunin, N. Belkhatir, "Architectures for Process Support Ineroperability", ICSP5,Chicago, 15-17 juin, 1997.

[3] J. L. Crowley, "Integration and Control of Reactive Visual Processes", Robotics and Autonomous Systems, Vol 15, No. 1, décembre 1995.

[4] J. Rasure et S. Kubica, "The Khoros application development environment ", in Experimental Environments for computer vision and image processing, H. Christensen et J. L. Crowley, Eds, World Scientific Press, pp 1-32, 1994.

[5] M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Disciplines, Prentice Hall, 1996.

[6] T. Winograd, "Architecture for Context", Human Computer Interaction, Vol. 16, pp401-419.

[7] R. C. Schank and R. P. Abelson, Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[8] M. Minsky, "A Framework for Representing Knowledge", in: The Psychology of Computer Vision, P. Winston, Ed., McGraw Hill, New York, 1975.

[9] M. R. Quillian, "Semantic Memory", in Semantic Information Processing, Ed: M. Minsky, MIT Press, Cambridge, May, 1968.

[10] D. Bobrow: "An Overview of KRL", Cognitive Science 1(1), 1977.

[11] R. Brooks, , "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, RA-2, no. 1, 1986.

[12] A. R. Hanson, and E. M. Riseman, , VISIONS: A Computer Vision System for Interpreting Scenes, in Computer Vision Systems, A.R. Hanson & E.M. Riseman, Academic Press, New York, N.Y., pp. 303-334, 1978.

[13] B. A.Draper, R. T. Collins, J. Brolio, A. R. Hansen, and E. M. Riseman, "The Schema System", International Journal of Computer Vision, Kluwer, 2(3), Jan 1989.

[14] M.A. Fischler & T.A. Strat. Recognising objects in a Natural Environment; A Contextual Vision System (CVS). DARPA Image Understanding Workshop, Morgan Kauffman, Los Angeles, CA. pp. 774-797, 1989.

[15] R. Bajcsy, Active perception, Proceedings of the IEEE , Vol. 76, No 8, pp. 996-1006, August 1988.

[16] J. Y. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active Vision", International Journal of Computer Vision, Vol. 1, No. 4, Jan. 1988.

[17] J. L. Crowley and H. I Christensen, Vision as Process, Springer Verlag, Heidelberg, 1993.
[18] B. Schilit, and M. Theimer, "Disseminating active map information to mobile hosts", IEEE Network, Vol 8 pp 22-32, 1994.
[19] P. J. Brown, "The Stick-e document: a framework for creating context aware applications", in Proceedings of Electronic Publishing, '96, pp 259-272.
[20] T. Rodden, K.Cheverest, K. Davies and A. Dix, "Exploiting context in HCI design for mobile systems", Workshop on Human Computer Interaction with Mobile Devices 1998.
[21] A. Ward, A. Jones and A. Hopper, "A new location technique for the active office", IEEE Personal Comunications 1997. Vol 4. pp 42-47.
[22] K. Cheverest, N. Davies and K. Mitchel, "Developing a context aware electronic tourist guide: Some issues and experiences", in Proceedings of ACM CHI '00, pp 17-24, ACM Press, New York, 2000.
[23] J. Pascoe "Adding generic contextual capabilities to wearable computers", in Proceedings of the 2nd International Symposium on Wearable Computers, pp 92-99, 1998.
[24] Dey, A. K. "Understanding and using context", Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.
[25] J. L. Crowley 1 , J. Coutaz , G. Rey and P. Reignier, "Perceptual Components for Context Aware Computing", Proceedings of Ubicomp 2002, Sweden, September 2002.
[26] P Nixon, S Dobson, and G Lacey (Eds), Managing Interaction in Smart Environments, Springer Verlag Press, pp. 220, 1999.
[27] W Hillier, Space is the machine, Cambridge University Press, 1996.