

# Designing architectures to support mobile learning

School of Computer Science  
University of Birmingham  
Edgbaston  
Birmingham  
B15 2TT  
UK  
R.Beale@cs.bham.ac.uk

**Abstract:** Modern technologies support us learning in an ever-increasing number of situations, and mobile learning is moving from the feasible to the accessible. We identify some key characteristics for systems that support the user, and then go on to describe a software architecture that supports system intelligence and has all the necessary features to provide mobile learners with timely, simple, and relevant information.

**Keywords:** Software architectures, intelligence, user support, learning, mobility

## Introduction

We learn things all the time - whether we are reminding ourselves of something we once knew but have forgotten, are extending our knowledge of a familiar topic, or tackling something new, we undertake learning in a variety of places and settings. Learning is not restricted to classrooms and formal education - it can occur on an ad hoc basis, often 'just-in-time', and covers the spectrum of human experience, from science and language through culture and art to sport, gardening and DIY. We are now in a phase where modern technology allows us to provide electronic support for learning; technological advances have moved us on from the mobile computing paradigms of "anywhere, anytime" access to information and resources towards ubiquitous computing, which we can characterise as "everywhere, everytime".

The list of technologies that power our progression is large, and still-growing (802.11a-g, bluetooth, 3G, GPRS, etc.), and is supported by a raft of software implementations (J2ME, J2SE, .net, ...) and theories (OO programming, extreme programming, agent-based systems). These can be coupled with ever-decreasing sizes of portable device with increasing power, from the mobile phone to the handheld computer to the notebook to portable heads-up displays: all of these allow us to envisage, design and deliver systems to meet the needs of these technologically-aware users. However, we need to understand what these needs actually are, and identify the characteristics of systems that will effectively support such approaches.

## Key Characteristics for Supportive Systems

Workflow studies to e-learning assessments, from sociology to management, provide guidance on the characteristics of systems that can support people effectively. We can identify some of the key characteristics as follows:

- timely - providing the correct information just when the person needs it. In order to achieve this, it is necessary for the system to know what it is the user is doing, the wider context in which they are working, and their desires and preferences.
- simple - the technology that supports these activities must be easy to use. Mobile telephony is hugely popular, because not only does it allow people to organise their lives in a much more fluid, personally convenient way, but it does so through a simple interface. Interacting with the

technologies themselves should be hard. That is, they should be invisible, behind the scenes, acting much like magic to make things happen the way we want them to.

- relevant - understanding what is important at a particular time requires a strong notion of context. Whereas timeliness looks at the temporal nature of information presentation, relevance works on a larger scale and brings in the multiplicity of tasks that may occupy someone, and can guide them through expanding threads of enquiry by helping them focus their activities.
- information - there is a host of data available to the connected user, but information is a much more valuable resource. Information is a useful, structured, understandable presentation of data. We can class knowledge as a special form of information, though we have to recognise that users generate data, information and knowledge too - and that this happens both formally (e.g. creating an analysis and summary of field test data) or informally (e.g. in making notations that modify a recipe to enhance its flavour).

Such a system we term a LIFE support system - Learning, Information, Facilitation, Entertainment. It supports ongoing learning about tasks, both of the formal kind (how to do differential equations, the life of a famous painter, etc.) and informal learning (a faster route to work, etc.); it provides information as and when appropriate; it will facilitate communication with other people, other systems and the world at large; and it can offer us access to a variety of entertainment media from film to radio to music to television to interactive chat.

The identified features suggest that any system that is to meet the needs of these users has to have a number of characteristics. We can categorise these under three headings: learning, mobility and intelligence.

#### *Learning*

Systems to support education and learning must be appropriate themselves. Simply providing a medium to access information is not enough - learning is more than having a library available, it is a process, a dialogue, a guided path or a voyage of discovery. There are multitudes of learning styles, teaching styles and student styles that have to be understood and supported in order to effectively transfer knowledge and increase understanding [1]. There is a corpus of knowledge in this field that is being ever extended, and current research efforts are developing these learning theories and practices into the mobile arena. Whilst we will not discuss them in this paper, they should not be forgotten as a major component of a successful system.

#### *Mobility*

People are everywhere, doing tasks everywhere, and so may well want to learn wherever they happen to be. It is therefore imperative that LIFE support systems have ubiquitous availability. Systems need to be permanently accessible and always able to offer useful support and assistance. In practical terms, this means that they must be tough enough to survive everyday life, have a decent battery life, be reliable and not crash, not take ages to start up, have sufficient memory and decent display technologies to present information, and be small and light.

We require the systems to have high levels of connectivity. Note that permanent connectivity is not required, if the system is able to anticipate both the user's needs and can be aware of the resources available to it in the future - reduced network access can be compensated for by a download in advance of the necessary information, for example. However, to support new activities and to allow for flexibility, the greater connectivity that can be achieved, the better. Again, mobile phones provide an exemplar system here. When a user is in a good coverage area and not in direct engagement with other people, they are easily able to make a call to whoever they like. However, in situations where the user has limited access to their phone, as in a meeting, they are still able to send text messages: much lower bandwidth and supporting a different activity, but still perceived as a useful activity.

#### *Intelligence*

In order to be effective, LIFE support systems need to have a degree of intelligence in order to provide the features required. For example, modelling the users preferences needs to be done in order to refine the available information to determine its relevance. Whether this is done directly using question and answer sessions to build up profiles or indirectly by monitoring behaviours and developing implicit user models is an implementation and usability question, rather than a requirements issue. Intelligence goes far beyond modelling the user, however. A useful system needs to develop task models so that it can anticipate what will it will be asked to support next, and needs to continuously monitor the network in order to balance bandwidth availability with local memory storage capabilities. Intelligence in a system allows it to choose the most appropriate sources of information, taking into account content, media and bandwidth, and can guide the user through a vast array of resources and competing demands on their time. Intelligence within systems allows the user to engage in dialogue with them, refining their needs and adapting to ever-changing circumstances.

One specific instantiation of intelligence within systems is that of context awareness. Being context-aware allows you to understand what someone is doing, where they are doing it and what other tools and people they are interacting with in order to do it. This provides a host of information that systems can then utilise to work more efficiently and effectively. Context is both awareness of environment and of activity, and it can be seen as a way of filtering and reducing the number of options that a system has to cover. One of the attractive features of context awareness is the relative ease with which it can be implemented at a technical level. Location is the prime factor in context, and GPS modules or IR tags can provide positional or semantic location information relatively easily.

Many of the characteristics we have identified are present in e-learning and mobile learning systems, but whilst these are developing all the time, they are offering support for a reduced set of the things that we do in our everyday lives. Recent research has shown that we have at least sixteen ongoing learning tasks at any one time, most of which are informal ones, and which go mostly unsupported by technology. Most systems focus primarily on the characteristics of learning and mobility; it seems clear that truly effective systems will have to incorporate some form of intelligence into their design in order to more appropriately support people in the wider context of their lives.

### **Architectures to support intelligence**

We need to address the question of what is the best architecture to support such systems which have variable connectivity, multiple channels, ad-hoc conversations, structured and informal interactions, uncertain environments, and the need for embedded intelligence. We have developed the activeSpace architecture, which is a component-based architecture focussed on supporting intelligent agents. The basic architecture has been successfully used to deploy agent-based systems for user modelling, shared calendaring and scheduling, and informal task support on the Windows and Unix desktops[2].

The architecture has a number of key distinguishing features.

- **independence of data flow and initiative**

The connection points of components are typed data sources and sinks. In addition each connection point can be designated either as a push (data-driven) or pull (demand-driven) kind. This enables mixed-initiative interfaces driven partly by the user's actions and partly by external contextual events. It also facilitates systems that can operate over different kinds of networks with delays, periods of disconnection etc., where different modes of push/pull information flow may need to dynamically change.

- **external linkage**

The links between connection points of components are established externally to the components themselves. This is unlike a typical object framework where one or other object has to 'know'

about others, and similar to brokerage systems such as ANSA or CORBA, but more lightweight. This allows 'emergent' properties of the system to develop, producing more complex and powerful interactions. The distributed nature of the architecture provides an impetus for such a lightweight mechanism as well.

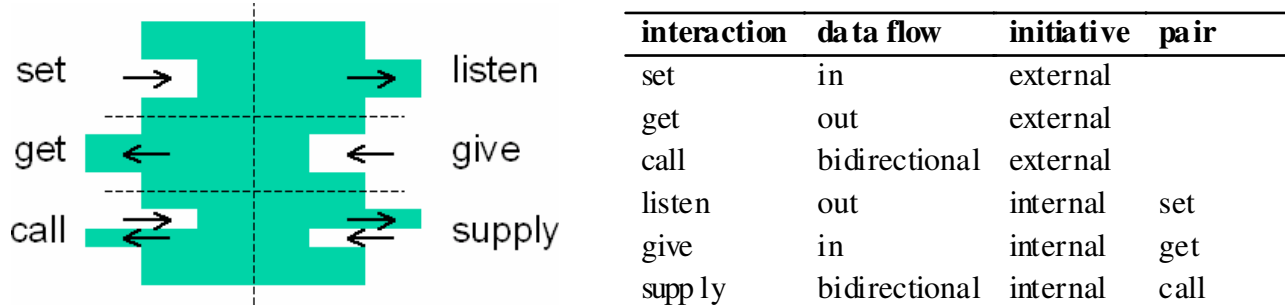


Figure 1 – Typical component connection points

The framework has at least three types of specialised components:

- **sources**  
Components that monitor some kind of external events or data
- **transcoders/recognisers**  
Components that take data sources of one type and attempt to match them and deliver more precise or derived data. For example, a recogniser might take a GPS location and translate it into a post code.
- **services**  
Components that take data of a certain type and do something with it. For example, a service might offer a map based on a postcode.

These are linked using blackboard style architecture to allow flexible context-driven applications, using the external linkage to allow 'plug-and-play' of various intelligent components.

The component structure is based on theoretical work of status–event analysis, which allows the specification of continuous phenomena typical of many context sources as well as event phenomena in digital systems[3,4]. The architecture is derived from some of the earliest work in context-aware interfaces[5,6,7].

#### **Implementation**

The architectural design outlined above has been implemented, in both C++ and in Java. The details of the implementation are language-dependent, but the overall system behaves similarly. Something about run time plasticity, adaptation to network services or not, support for intelligence in components.

For LIFE support systems, sources provide information into the system - they capture environmental and user-based events or triggers, and make the system aware of them. The transcoders or recognisers provide one of the levels of intelligence within the system, and can be as simple or as complex as is required. For example, we have a 'name' recogniser that looks for two consecutive words that are each capitalised, and if it finds them it provides a semantic tag to identify them as a name. This name

component may then be picked up by another recogniser, which uses names to search in telephone directory, returning a phone number. If we find that this name recogniser is much too error-prone, the component nature of the system means that it is easy to replace it with a much more complex algorithm that may use an built-in database to much more successfully identify names, whether capitalised or not. The interlinking of the results of one recogniser with another means that very soon quite complex patterns of information analysis, refinement and flow can be built up, not all directly programmed but inferred from the environment and the series of interpretations that the recognisers put on the data. We have produced all manner of recognisers, from the simple ones described above to task modelling ones that identify particular sequences of user actions and statistical ones that analyse numeric information and summarise it. It is interesting to note, however, that some of the more interesting results that the system comes up with are as a result of unforeseen sequences of interactions between recognisers, showing that the emergent intelligence properties of the system are at least as important as the intelligence explicitly programmed into the individual recognisers. Within LIFE support systems, one of the main tasks of the recognisers is to provide context awareness.

The final part of the architecture, the services, provide access to the actual materials and teaching/learning styles that influence their delivery. Given information on the user, their task and its context from the transcoders, they are then in a position to deliver and interact in an appropriate manner to provide an effective outcome. Such a system is dynamic, in that transcoders are constantly monitoring the learning interaction and providing hints and advice to the service system, as well as maintaining an awareness of the user's wider goals - for example, are they still actively engaged in this task or have them moved on to another?

### **Conclusion**

We have argued that intelligence is as necessary a part of a learning support system as is mobility and the appropriateness of material and its delivery. In the real world, the variety and number of tasks that need support require that the information around them be filtered, focussed and understood in order to be appropriately supported. We have developed an architecture within which we can implement such systems. Work is progressing on both identifying and understanding the major factors that contribute towards context-awareness, with specific efforts going into building in resources that provide location and task awareness. These can be coupled with user models of varying degrees of complexity, and simple services that perform relatively simple functions. Allowing these to interact with each other produces a system of holistic complexity in which complex behaviours can emerge from the interactions of even the simplest components. We are now entering a fascinating stage in the development of mobile learning systems, which have the potential to provide effective, appropriate support in a wide variety of circumstances.

### **Acknowledgements**

Andy Wood and Alan Dix made substantial contributions to the development of this work. Thanks also to the aQtive team.

### **References**

1. R. Beale and M. Sharples (2002). **Design Guide for Developers of Educational Software**, British Educational Communications and Technology Agency (Becta).
2. R. Beale, A. Dix and A. Wood (2000), onCue, [www.aQtive.net](http://www.aQtive.net)
3. A. Dix and G. Abowd (1996). **Modelling status and event behaviour of interactive systems**. *Software Engineering Journal*, **11**(6) pp. 334-346.
4. A. Dix (1998). **Finding Out - event discovery using status-event analysis** *Formal Aspects of Human Computer Interaction* FAHCI98, Sheffield, 5th&6th September 1998.
5. A. K. Dey, G. D. Abowd, A. Wood (1998). **CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services**. In Proceedings of Intelligent User Interfaces '98 (IUI '98), pp. 47-54, Jan, 1998.

6. A. Dix, R. Beale and A. Wood (2000). **Architectures to make Simple Visualisations using Simple Systems.** *Proceedings of Advanced Visual Interfaces - AVI2000*, ACM Press, pp. 51-60
7. R. Beale (1998). **Intelligent components for interactive multimedia**, IEE, *Neural networks and multimedia*, Oct 1998.