Guidelines and Open Issues in Systems Support for Ubicomp
Reflections on UbiSys 2003 and 2004

Manuel Roman, Adrian Friday, Christian Becker and Jalal Al-Muhtadi

One of the most enjoyable features of the Systems Support for Ubiquitous Computing (UbiSys) workshops in 2003 and 2004 has been the freeform small group and plenary discussion sessions. Last year, we focused on identifying 'hot topics' from the paper sessions to form break out groups for further discussion. This year the discussion session was motivated by a thought provoking talk from Wim Stut from Philips Research, where he highlighted two aspects they had identified during talks with researchers in this domain, that he believed were often overlooked in existing scenarios:

1. Self-configuration

2. Graceful degradation

He argued that existing supporting infrastructures often make unreasonable assumptions about the degree to which the location, context and users of devices involved in typical Ubicomp application scenarios are known by the system. This knowledge is typically 'just there', i.e. is established a priori by system developers or "power" users, or is even so implicit to the scenario that the authors do not realize that they are making these assumptions in the first place! Such information makes systems too complex to use in real-life scenarios. These systems are complex and dynamic and therefore require functionality to configure themselves automatically, or through trivial user interaction.

The second issue is graceful degradation and affects the response of the system to changes and particularly failures in the environment. According to Wim, most existing projects assume the availability of certain resources such as Internet connectivity and specific servers to be present permanently. However, in situations where these resources are not available, the entire system stops working. While this might be acceptable in experimental lab environments, real life demands systems that can cope with the lack of resources. Or better still, systems should be able to adapt gracefully to these changes, preserving as much functionality as possible using the resources that are available.

This resonated with an issue from last year's discussion on, as one of our attendees Deborah Zukowski so aptly put it, the systems' need to demonstrate 'tolerance for ignorance'. We often propose data-driven systems where low level sensing leads to derived context (e.g. based on sensor fusion, statistical or knowledge based techniques) in order to move closer to concepts understandable by the user. In reality, the system won't be able to perfectly sense the state of the world – there will inevitably be 'gaps' in this knowledge, e.g. things that can't be sensed or have been erroneously or imprecisely recorded. This may in turn lead to incorrect 2nd order data that is mistakenly inferred or derived from it. Our systems will need to gracefully adapt, again preserving functionality,

and crucially, should involve the user by exposing this ambiguity or explaining the motivation for any actions taken on their behalf.

These points were also reflected in the more detailed discussion on the following topics:

1. Mapping between physical objects, their digital representation, and their location

2. Distinguishing technical aspects from user intent when reasoning about self-configurable systems

3. Keeping the users always 'in the loop'

4. Defining assessment criteria to assist in the evaluation of ubiquitous computing systems

The first topic, mapping between physical and digital objects was triggered during Wim's presentation (and is clearly related to the issue of self-configuration). Many projects assume knowledge in the system, such as the physical locations of users and devices. A light bulb may know that it is 'in the lounge', the orientation of the television screen and position in the room are both accurately known, the presence, name, relationships or rights of users are represented and understood by the system. How this information came to be known, sensed or inferred is typically not explained. In real life scenarios the system must be configured with this information or must be able to obtain this information automatically. During the discussion, some participants suggested using 'physical space servers' that provide information about the physical layout of the environment, as well as properties such as location of physical objects. For example, when looking for a projector for a presentation, the system should provide a list of projectors located in the presentation's room and should discard projectors available in neighbouring rooms. Discovery services should be extended to take into account physical constraints. Keeping this information up to date is clearly a challenge, but it was pointed out that connectivity and interaction patterns over time can provide useful hints.

The second and third topics were the most extensively discussed. These topics arose during the first discussion. If the system requires information such as physical properties and relationships of surrounding entities, should the user provide this information? Most people agreed that the system should be able to configure itself. However, the fact that the system can configure itself does not mean that the system can predict what the user wants to do. There are a large number of research papers that describe scenarios where users perform different activities and ubiquitous computing systems automatically configure existing resources or take actions proactively to optimise the environment for such activities. However, during the discussion it was clear that we did not believe it possible to accurately infer user intention and therefore ubiquitous computing systems should focus on the technical (enabling) aspects of Ubiquitous Computing. Research on systems support should thus focus on providing functionality to simplify the management of resources (discovery, coordination, aggregation, use etc.) and should let the user

choose how to combine and use those resources at any time. This raised the suggestion that we apply two 'rules of thumb':

1. That the user in Ubicomp should be always part of the loop, and

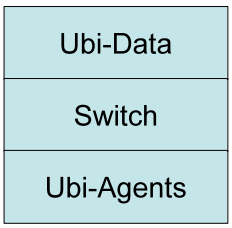
2. To 'sanity check' our systems and scenarios by applying the following test: "If you had a personal human servant that followed you around, could they correctly anticipate what you wanted under the same circumstances?"

During the session, we discussed the fact that providing self-configuration support for individual resources is possible, and in fact, most existing software already provides mechanisms to react to certain errors or configuration states. The complexity arises when combining heterogeneous resources dynamically and trying to automate self-configuration and operation of the system as a single entity. This is clearly an important topic for future research.

A significant emergent issue from last year's workshop, was the apparent need for evaluation frameworks for systems support for Ubicomp: i.e. whether it is possible to create a set of benchmarks or a test suite that allows us to compare and contrast our approaches, allowing for systematic evaluation of Ubicomp infrastructures. This framework would be analogous to the standard test suites selected to exercise audio or video compression algorithms, or the well known datasets used for Information Visualisation or Natural Language Processing. One insight in this regard was that each community picks and accepts its own preferred mechanisms or suites of benchmarks, it is just that Ubiquitous Computing seems to be composed of many communities!

This year we continued this discussion by considering the definition of assessment criteria to assist in the evaluation of Ubicomp systems. Although we could not quickly identify such a set of criteria, the participants proposed the creation of a workshop that would specify a particular Ubicomp scenario, and would request submissions from authors explaining how they would support such scenario using their particular technology. As part of the workshop guidelines, we would ask authors to explain how their system deals with self-configuration and graceful degradation problems. The result of the workshop would be a collection of solutions describing a common scenario that would allow us to generate a document with criteria to evaluate ubiquitous computing systems. These topics would form the basis of a new set of benchmarks against which researchers could evaluate their contributions.

The final thread of discussion was that of how we might design systems to support Ubiquitous Computing environments that scale while retaining the resilience to failures we've recognised. Last year this turned to a more fundamental discussion of the underlying composition of architectures for supporting Ubicomp applications. We regarded this architecture as a layered model consisting of a pool of sensed, inferred and application data (*Ubi-Data*), application agents (in the loosest sense) working with and on behalf of the user and '*the switch*' that binds these together.



We might draw parallels between ‘the switch’ and existing Ubicomp middleware platforms. However, while such platforms typically promote the sharing of information between applications (such as the context described earlier) and the rewiring and flexible composition of components to create new applications, we noted that we did not yet have mechanisms for avoiding ‘interference’ between these applications – potentially a major limitation to scale and resilience as these environments increase in complexity. Studying the architecture of ‘the Ubicomp switch’ seemed to be an important direction for understanding how to build scalable and robust Ubicomp environments in the future.