Émigré: Metalevel Architecture and Migratory Work

Paul Dourish and André van der Hoek

Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425 USA
{jpd,andre}@ics.uci.edu

Abstract. Migratory work extends traditional mobile work with an innate awareness of, and adaptability to, both technical and social surroundings. We are designing a technical framework, Émigré, that is based on the use of architectural meta-level representations to support rapid development and semi-automated run-time adaptation of migratory work applications.

1 Introduction

Not too long ago, for computer-based workers, moving outside the office meant leaving work behind—regardless of whether such a move involved getting coffee down the hall, attending a company meeting, visiting a customer in town, or traveling outside the country. Rapid and major technological advances, however, lead to the use of laptops, cell phones, pagers, Personal Digital Assistants, and many other electronic devices that allow users to bring work and stay connected with the office at all times and all places. Mobile computing has arrived; now the struggle is how to use it effectively.

Conventional approaches to mobile computing focus on *eliminating* boundaries by delivering seamless access to information and computation wherever a user goes. The motto of "any time, anywhere" is testament to this vision. Unfortunately, the vision of complete seamlessness is an illusion. In part, this is a technical matter, caused by variations in processor performance, storage architecture, and network throughput; in part, it is a social matter, reflecting the different forms of information needs and acceptable practice in environments as diverse as offices, meeting rooms, cars, homes, and restaurants. These technical and social variations create clearly visible boundaries that current mobile technology blithely ignores.

Rather than attempting to eliminate boundaries, our work focuses on *accommodating* boundaries. In contrast to mobile work, we focus on what we call *migratory work* – activities that move from place to place, device to device, and environment to environment, adjusting to suit changing circumstances. Migratory work applications, then, distinguish themselves in their ability to adapt to their surroundings—not only in terms of location, but also in terms of devices on which they operate, available infrastructure, additional devices and applications usable in the vicinity, and social and organizational settings. For example, consider the migration of an application from a person's office to a meeting room. The application must shift from an environment in which it executes on a powerful desktop with a range of input devices and multiple high-resolution dis-

plays, to an environment in which it executes on a simple PDA with a stylus as the single input device and a small text display as the only output device. Perhaps even more importantly, the application must be sensitive to its context of use: in the office it has the sole attention of the person and can behave rather intrusively, but in the meeting room it has limited attention and must be as unobtrusive as possible.

These are the challenges that our work aims to address. The broad research question we are tackling, then, is *how a migratory work application can relate its current and future behavior to the setting in which it is currently operating*—thereby allowing an application to effectively migrate from setting to setting and support its users in a manner appropriate to each of those settings. Our hypothesis is that *meta-level architectures provide a convenient and effective approach* for addressing this question. One common problem of meta-level architectures, however, is the complexity of developing both a base level and a meta-level representation, and maintaining the synchronization between them. We will address these problems by harnessing ongoing research into run-time architectural description languages. Specifically, software architecture promotes the use of explicit architectural models of components and connectors as part of the development process. We intend to exploit these capabilities in investigating the use of run-time architectural models as the basis for dynamic meta-level application control.

2 Background

2.1 Metalevel Representations

In traditional approaches to system design, each module offers a single interface to its client modules, creating an abstraction barrier between clients and the modules' implementations. This barrier confers many important benefits on the design, including portability, compositionality, and reuse. However, it is also a source of problems. Abstraction barriers hide implementation decisions that may, in fact, be important for specific clients (consider a virtual memory paging algorithm that is optimized for certain patterns of memory use but can lead to disastrous behavior with other patterns of use). To circumvent this problem, some systems now offer both a traditional interface through which they can be used and a meta-level interface through which they can be examined and controlled (Kiczales, 1992). In effect, through the meta-level interface, these systems offer a representation of their own behavior - a representation that can be manipulated in order to adapt the system to different needs and different circumstances.

These kinds of meta-level representations, and the principle of computational reflection on which they are based, were originally explored in the area of programming languages, although the same problems appear in many other areas of system design (e.g., Dourish, 1996). However, specific meta-level representations have typically been developed in an ad hoc fashion, in response to particular problems, but without a consistent frame of reference or grounding in software design practice.

2.2 Architectural Description Languages

Our starting point for infrastructure design is our current work on architectural modeling languages (Dashofy et al., 2001). These languages allow system developers and de-

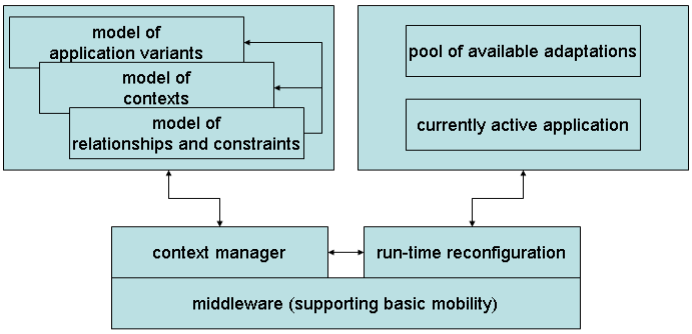

Figure 1 Architectural outline.

signers to construct models of system architecture - components, connectors, interactions, etc. xADL 2.0, the language with which we have been working thus far, provides us three important advantages over other ADLs. First, it models not just static architectures, but also aspects of dynamic run-time configurations, which makes it ideal as a basis for dynamic adaptability; second, it provides basic functionality for modeling architectural variants, making it ideal for modeling the alternative run-time incarnations of a migratory work application; and third, it is specified in XML, which makes it highly extensible and therefore suitable as a base modeling language to which we can add the necessary constructs that are unique to modeling migratory work applications.

One important component of the infrastructure is the means by which the system will sense aspects of its environment. A range of existing technologies provide support for local and remote service discovery (e.g., Sun's Jini), localized communication (e.g., BlueTooth), etc. Our intention is employ these existing solutions rather than to develop novel sensing techniques. However, the model-based approach allows us to decouple sensing from the use of this information, so that Émigré can incorporate novel mechanisms for sensing and discovery as they become available.

3 Approach

Figure 1 introduces the overall architecture of Émigré. The architecture separates the models, infrastructure, and application into three separate entities. The *models* are further subdivided in models of potential variants in which an application can reside at run-time, models of context, and models of the relationships and constraints that are present among application variants and contexts. Separating the models into these three categories not only promotes a strong separation of concerns, but also promotes reuse of models among different migratory work applications.

The *infrastructure* is further subdivided in a basic middleware that supports mobility, a context manager, and a run-time reconfiguration component. The context manager, which is aware of different application contexts, combined with the run-time reconfiguration component, which is able to drive the reconfiguration of an application from one variant into another by drawing from the pool of available adaptations, extends the basic mobile middleware into a middleware that is suitable for migratory work.

4 Challenges and Directions

The work described here is at a very early stage. At the time of writing, we are engaged in initial experiments integrating various infrastructure components to demonstrate the use of xADL 2.0 as a metalevel representation. Further work will address the core challenges, which are to determine the effective bounds of the design space imposed by particular architectural models, and the appropriate design parameters that balance flexibility against cost of development. This work is ongoing and we look forward to being able to present it in the future.

However, even at this stage, we believe that some of the central insights that have arisen and which motivate our design are potentially valuable for others working in this area. Three of these are significant here.

The first is the separation of migratory from mobile work. Rather than respecting the detail of the different settings and situations in which work must be conducted, much research into mobility attempts to obscure that detail by fostering an attitude of seamlessness and ubiquity. In contrast, our approach recognizes the inherently heterogeneous nature of different social and technical settings and attempts to incorporate that heterogeneity into the interactive experience.

The second is the identification of the role for explicit metalevel representations. The diversity of settings for migratory applications potentially leads to a combinatorial explosion which renders the development problems intractable, as we attempt to deal with the variety of applications, components, infrastructure demands and social factors. The particular advantage of the metalevel approach is that it yields a basis for flexibility which is at once principled and extensible. Using architectural description languages as the metalevel “glue” further allows us to integrate this new approach into conventional software development practice.

The third insight is that the problems of migratory work cannot be solved purely from a software perspective or purely from an interaction perspective, but require a new approach that brings these two concerns together. The relationship between software architecture and design on the one hand and interaction and user experience on the other is never stronger than in domains in which each is tested to the limit, as is the case in most mobile applications. A unified and coordinated approach is absolutely essential.

Our current work with Émigré is an example of the integration of these three principles. Our early experiences are positive, and we hope to encourage others to adopt the principles and explore other parts of the design space.

References

[1] Dashofy, E., van der Hoek, A., and Taylor, R. 2001. A Highly-Extensible, XML-Based Architectural Description Language. *Proc.Working IEEE/IFIP Conf. Software Architectures WICSA 2001*. Amsterdam, Netherlands.

[2] Dourish, P. 1998. Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications. *ACM Trans. Computer-Human Interaction*, 5(2), 109-155.

[3] Kiczales, G. 1992. Towards a New Model of Abstraction in Software Engineering. *Proc. IMSA Workshop on Reflection and Metalevel Architectures*. Tokyo, Japan.