The Journal of Systems and Software 83 (2010) 972–989

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An architecture-driven software mobility framework

Sam Malek [a,*], George Edwards [b], Yuriy Brun [c], Hossein Tajalli [b], Joshua Garcia [b], Ivo Krka [b], Nenad Medvidovic [b], Marija Mikic-Rakic [d], Gaurav S. Sukhatme [b]

[a] *Department of Computer Science, George Mason University, Fairfax, VA, USA*
[b] *Computer Science Department, University of Southern California, Los Angeles, CA, USA*
[c] *Computer Science & Engineering, University of Washington, Seattle, WA, USA*
[d] *Google Inc., 1333 2nd Street, Santa Monica, CA, USA*

ARTICLE INFO

Article history:
Received 1 December 2008
Received in revised form 25 August 2009
Accepted 1 November 2009
Available online 4 November 2009

Keywords:
Software architecture
Mobility
Quality of service analysis
Robotics

ABSTRACT

Software architecture has been shown to provide an appropriate level of granularity for assessing a software system's quality attributes (e.g., performance and dependability). Similarly, previous research has adopted an architecture-centric approach to reasoning about and managing the run-time adaptation of software systems. For mobile and pervasive software systems, which are known to be innately dynamic and unpredictable, the ability to assess a system's quality attributes and manage its dynamic run-time behavior is especially important. In the past, researchers have argued that a software architecture-based approach can be instrumental in facilitating mobile computing. In this paper, we present an integrated architecture-driven framework for modeling, analysis, implementation, deployment, and run-time migration of software systems executing on distributed, mobile, heterogeneous computing platforms. In particular, we describe the framework's support for dealing with the challenges posed by both logical and physical mobility. We also provide an overview of our experience with applying the framework to a family of distributed mobile robotics systems. This experience has verified our envisioned benefits of the approach, and has helped us to identify several avenues of future work.

1. Introduction

As the global computing infrastructure transitions from an emphasis on personal computers to mobile and embedded devices, ensuring the quality of complex distributed software systems remains an essential focus of research in software engineering and, particularly, software architecture. Software quality is measured in terms of *quality attributes*, such as performance and dependability, that are identified and prioritized by system stakeholders. In a mobile environment, system parameters such as network reliability and throughput are far less predictable than in static environments. Moreover, new quality attributes such as energy consumption would also need to be taken into account in the design and construction of these systems. Thus, for systems distributed on mobile hardware devices, such as smart phones and wearable computers, evaluating software quality is even more challenging than for traditional systems.

It has long been acknowledged that software architecture provides an effective foundation for the quality assurance of large, complex systems (e.g., Abowd et al., 1995; Medvidovic and Taylor, 2000; Clements et al., 2002; Malek et al., 2007). The key underpinning of our work is the observation that an explicit architectural focus can also be instrumental in facilitating mobile computing (Chan and Chuang, 2003; Ciancarini and Mascolo, 1998; Medvidovic et al., 2003; Malek et al., 2005b; Malek et al., 2006). Architecture-driven approaches to quality assurance use *architectural abstractions* – software components, connectors, communication ports, events, etc. – to manage complexity and leverage *architectural styles* to enforce constraints and promote desired system characteristics. Analogously, architecture-driven approaches to mobility enable system migration and adaptation during run-time in a controlled fashion by employing architectural constructs as the units of mobility.

While existing research (Chan and Chuang, 2003; Ciancarini and Mascolo, 1998; Sousa and Garlan, 2002), including our own (Medvidovic et al., 2003; Malek et al., 2005b; Malek et al., 2006), has verified the advantages of an architecture-centric approach in the development of mobile software systems, in practice, the adoption of such approaches has been limited. We argue that this is due to the lack of a comprehensive support for architecture-based development of mobile software systems. In other words, the majority

* Corresponding author.
E-mail addresses: smalek@gmu.edu (S. Malek), gedwards@usc.edu (G. Edwards), ybrun@usc.edu (Y. Brun), tajalli@usc.edu (H. Tajalli), joshuaga@usc.edu (J. Garcia), krka@usc.edu (I. Krka), neno@usc.edu (N. Medvidovic), marija@google.com (M. Mikic-Rakic), gaurav@usc.edu (G.S. Sukhatme).

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2009.11.003

of existing architectural research approaches and industrial tools have dealt with providing point solutions that address particular mobility concerns. As a result, the developers in the mobility setting have faced some difficulties with fully embracing architectural abstractions as the foundation for modeling, analyzing, implementing, monitoring, and adapting the system. Moreover, the discrepancies between the existing tools and techniques diminish some of the key advantages associated with taking an architecture-centric approach.

To better illustrate the current shortcomings and motivate the problem, let us consider a scenario in which a software developer uses an architectural modeling tool (Childs et al., 2006; Dashofy et al., 2005; Edwards et al., 2007) to design a system and analyze its quality attributes. Since the majority of mobile middleware platforms do not provide adequate support for the implementation of architectural abstractions (Malek et al., 2005b; Malek et al., 2007), the developer is forced either to implement them through a combination of low-level programming language constructs (e.g., variables, collections, classes), or to misuse other middleware constructs (e.g., implement a connector as a middleware component). Performing such a complex mapping between constructs with different semantics and levels of granularity promotes *architectural erosion* (Perry and Wolf, 1992). In turn, the analysis performed on the architectural models becomes useless, as one cannot be certain of the fidelity of the implemented system with respect to the models.

The above example highlights only one set of problems that could arise due to the lack of complete life-cycle support for architecture-based development of mobile software systems. In this paper, we present and evaluate an integrated framework that aims to alleviate the shortcomings of the existing point solutions.[1] Specifically, our framework comprises:

- a tailorable *model* of mobile software architectural abstractions (Medvidovic et al., 2003; Edwards et al., 2007), mobile hardware platforms on which the software executes (Mikic-Rakic et al., 2004), and system quality requirements that are of particular importance to mobile systems (Mikic-Rakic et al., 2008);
- an extensible suite of architectural *analysis* techniques for mobile systems, including scenario-driven system simulations (Edwards and Medvidovic, 2008) and determination of effective deployments based on quality requirements (Mikic-Rakic et al., 2004, 2005; Malek et al., 2005a);
- a middleware platform (Malek et al., 2005b) targeted at architecture-centric *implementation* of mobile software, and an accompanying facility for stateful and stateless run-time *migration* of software components (Carzaniga et al., 1997);
- a continuous *monitoring* and architectural *awareness* methodology for detecting execution-condition changes in mobile software systems (Tisato et al., 2000); and
- a facility for *(re)deployment* and run-time *adaptation* of a software system distributed among a set of mobile hardware hosts (Malek et al., 2007; Mikic-Rakic et al., 2008).

With the exception of mobility support, about which we have only hypothesized in the context of system deployment in Mikic-Rakic and Medvidovic (2002) and Mikic-Rakic et al. (2008), the individual elements of the above framework have been published previously. This paper describes and illustrates those aspects of our framework that are pertinent to mobility. Moreover, the main contribution of our work is the manner in which they are combined to provide complete architecture-driven mobility support.

The framework is broadly concerned with the challenges mobility presents. We model the impact of physical mobility on the system's resources, such as network connectivity and battery power. We use simulation and analytical models to assess the degradation of quality attributes due to movement of devices and employ run-time adaptation to mitigate such problems. Note that since the framework has no explicit control over the actual movement of devices, we do not model the movement, but rather its impact on the system. However, if necessary, we believe the framework could be extended to model these aspects of mobility as well. We model logical mobility in terms of changes to the system's *deployment architecture* (Malek, 2007) – a representation of the system's software architecture superimposed on its hardware configuration and network topology. By adopting an architecture-based approach to development and adaptation, we avoid architectural erosion due to logical mobility. At run-time, we optimize the software system's quality attributes by finding a new deployment architecture and effecting it through logical mobility. Finally, the framework addresses other concerns in the mobile setting, such as heterogeneity of platforms and efficiency of implementation.

Our experiences with applying the framework on several mobile software systems have been very positive. For evaluation, we elaborate in detail on one such experience dealing with a family of mobile robotics systems, provide quantitative data that summarizes the results obtained in other real-world and synthesized examples, and qualitatively compare the framework with existing architectural frameworks.

The remainder of the paper is organized as follows. Section 2 details the challenges of building mobile systems and the framework's objectives in mitigating them. Section 3 provides a high-level overview of the framework, its accompanying tool suite, and how they are integrated with one another. Sections 4–8 describe the framework's support for mobility modeling, analysis, implementation, monitoring, and adaptation, respectively. Section 9 presents an overview of our experience to date with the framework, with data drawn primarily from the domain of mobile robotics. Section 10 relates this approach to existing work. We conclude the paper with the discussion of challenges that are guiding our ongoing work.

2. Challenges and objectives

As already alluded to in the previous section, mobile setting presents a number of unique software development challenges that permeate the entire software-engineering life-cycle:

Fluctuating execution context. Mobile software systems are characterized by their unknown operational profiles and fluctuating execution contexts. Since the properties of such systems (e.g., network connectivity, bandwidth, and energy consumption) constantly change at run-time and unanticipated events occur, an accurate analysis of the system's quality attributes is often not feasible at design-time.

Constrained resources. Mobile devices often have limited power, network bandwidth, processor speed, and memory. Constraints such as these demand highly efficient software systems in terms of computation, communication, and memory. They also demand unorthodox solutions, such as off-loading or migrating parts of a system to other devices.

Heterogeneity. Traditional computing increasingly relies on standard methods of representing data, computation, and communication, the best example of which is the SOA technology standards (i.e., XML, SOAP, WSDL) (Weerawarana et al., 2005). In contrast, mobile technologies remain largely proprietary. Engineer of such systems must reconcile proprietary operating systems such

[1] Note that our notion of *framework* is consistent with the term *architectural framework* as defined in IEEE 1471 and ISO/IEC 42010 standards (Maier et al., 2001; ANSI/IEEE, 2007). Our framework consists of several view points (e.g., deployment, dynamic, static), modeling languages (e.g., xADL, FSP), and is accompanied by a tool suite for specification and analysis.