DiaSim: A Parameterized Simulator for Pervasive Computing Applications

Julien Bruneau*, Wilfried Jouve, Charles Consel
INRIA / ENSEIRB, Talence, France
{julien.bruneau, wilfried.jouve, charles.consel}@inria.fr

***Abstract*—Pervasive computing applications involve both software concerns, like any software system, and integration concerns, for the constituent networked devices of the pervasive computing environment. This situation is problematic for testing because it requires acquiring, testing and interfacing a variety of software and hardware entities. This process can rapidly become costly and time-consuming when the target environment involves many entities.**

This paper introduces DiaSim, a simulator for pervasive computing applications. To cope with widely heterogeneous entities, DiaSim is parameterized with respect to a description of a target pervasive computing environment. This description is used to generate both a programming framework to develop the simulation logic and an emulation layer to execute applications. Furthermore, a simulation renderer is coupled to DiaSim to allow a simulated pervasive system to be visually monitored and debugged.

DiaSim has been implemented and used to simulate various pervasive computing systems in different application areas, demonstrating the generality of our parameterized approach.

I. Introduction

Numerous pervasive computing applications coordinate a variety of networked entities collecting context data from sensors and reacting by triggering actuators. To collect context data, sensors process stimuli that are observable changes of the environment (*e.g.,* fire and motion). Trigerring actuators is assumed to change the state of the environment. Developing a pervasive computing application requires to address a number of issues such as entity heterogeneity, physical constraints, and types of stimuli present in the target environment. Also, such an application needs to implement strategies to manage a variety of scenarios *e.g.,* fire situations, intrusions, and crowd emergency-escape plans. Consequently, in addition to the challenges of developing any software system, a pervasive computing system needs to validate the environment entities both individually and globally, to identify potential conflicts. For example, a fire manager and an entrance manager could issue contradicting commands to a building's door to respectively enable evacuation and ensure security. In practice, the many parameters to take into account for the development of a pervasive computing application can considerably lengthen this process. Not only does this situation has an impact on the application code, but it also involves changes to the physical layout of the target environment, making each iteration time-consuming and error-prone.

*Current affiliation: Thales Airborne Systems.

Various middlewares and programming frameworks have been proposed to ease the development of pervasive computing applications [1], [2], [3]. However, they require a fully-equipped pervasive computing environment for an application to be run and tested. As a result, an iteration process is still needed, involving the physical setting of the target environment and the application code.

In fact, the development of a pervasive computing system is very similar to the development of an embedded system. Like a pervasive computing system, an embedded system coordinates a number of heterogeneous hardware components that can be viewed as sensors (*e.g.,* microphones and buttons) and actuators (*e.g.,* displays and speakers). Some embedded systems are capable of discovering components dynamically, such as a smartphone detecting bluetooth components. As in the pervasive computing domain, embedded systems developers need to anticipate as wide a range of usage scenarios as possible to program their support. Despite similarities, the embedded systems domain differs from the pervasive computing domain in that it provides approaches and tools to facilitate software development for a system under design. Indeed, embedded systems applications can be tested and debugged using simulators [4]. Hardware components are *simulated* via software components that faithfully duplicate their observable behavior. And, the embedded systems application is *emulated*, executing as if it relied on hardware components, without requiring any code change. The study of embedded systems simulators gives us a practical basis for identifying the requirements for pervasive computing systems. Let us now examine these requirements.

a) Area-specific simulator: Like embedded systems, pervasive computing systems target a variety of application areas, including home automation, building surveillance and assisted living. Each area corresponds to specific pervasive computing environments, consisting of classes of entities dedicated to a given activity (*e.g.,* a light sensor, a motion detector or a wireless heart rate monitor). Correspondingly, the related stimuli drastically vary with respect to the target area. As a consequence, a simulation tool for the pervasive computing domain is required to deal with different application areas, enabling new classes of entities and stimuli to be introduced easily.

b) Transparent simulation: A key feature of most embedded systems simulators is that they emulate the execution of an application without requiring any change in the application

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6907
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6907

code. As a result, when the testing phase is completed, the application code can be uploaded as is and its logic does not require further debugging. The same functionality should be provided by a simulator for pervasive computing applications.

c) Testing a wide range of scenarios: Some pervasive computing applications address scenarios that cannot be tested because of the nature of stimuli involved (*e.g.,* fire and smoke). In other situations, the scenarios to be tested are large scale in terms of stimuli, entities and physical space they involve. These situations would benefit from a simulation phase to refine the requirements on the constituent entities of the environment, before acquiring them. Regardless of the nature of the target pervasive computing system, its application logic is best tested on a wide range of scenarios, while the system is under design. This strategy allows improvements to be made as early as possible in both its architecture and logic.

d) Simulation renderer: Like an embedded systems simulator, one for pervasive computing systems needs to visualize the simulation of scenarios. This simulation renderer needs to take into account various features of the pervasive computing domain. Specifically, it should support visual representations for an open-ended set of entities and stimuli, visual support for scenario monitoring, and debugging facilities to navigate in scenarios in terms of time and space.

Some existing approaches propose to visualize the simulation of pervasive computing applications [5], [6]. However, these approaches are limited because they require significant programming effort to address new pervasive computing areas. Furthermore, they do not provide a setting to test applications deterministically. The Lancaster simulator addresses this issue but does not support scenario definition [7]. The PiCSE simulator provides a comprehensive simulation model and generic libraries to create new scenarios. However, users have to manually specialize the simulator for every new application area [8].

II. Our approach

Our approach has been implemented in DiaSim, a simulator for pervasive computing applications based on sensors and actuators. This simulator is parameterized with respect to a high-level description of the target pervasive computing environment. Such a description defines the classes of entities, whether hardware or software, relevant to a target pervasive computing area. Both simulated and actual environments must conform to the same environment description, ensuring a functional correspondence between the two. Furthermore, the environment description is used to generate an emulation layer to run pervasive computing applications and a simulation programming framework for developing the simulation logic. Our approach makes it possible for the same application code to be simulated or executed in the actual environment. The resulting simulated pervasive computing environment enables to test the application logic against the full range of scenarios corresponding to the environment description. This simulation phase allows the pervasive computing system to be refined in terms of application logic and environment entities. DiaSim includes a simulation renderer enabling the developer to visually monitor and debug a pervasive computing system, navigating in terms of time and space in a simulation.

The contributions of our approach are as follows.

- *Parameterized simulator.* We present a simulator that is parameterized with respect to a high-level description of a pervasive computing environment.
- *Transparent simulation.* Our approach makes it possible for the same code to be simulated or executed in the actual environment. We ensure a functional correspondence between a simulated environment and an actual one by requiring both implementations to be in conformance with the pervasive computing environment description.
- *Hybrid environments.* An application can be executed in a hybrid environment, combining simulated and actual services. Hybrid simulation is a key feature to successfully transition to an actual environment: it allows actual services to be added incrementally in the simulation, as the implementation and deployment progress.
- *Generated simulation support.* A pervasive computing environment description is used to generate both an emulation layer, to execute applications, and a simulation programming framework, to develop simulated entities.
- *Simulation renderer.* We present a simulation renderer that enables the developer to visually monitor and debug a pervasive computing system.
- *Validation.* Our approach has been implemented in a tool called DiaSim. The generality of our parameterized approach has been demonstrated by simulating applications in a variety of pervasive computing areas. The practicality of DiaSim has been shown on a large-scale simulation of an engineering school.

References

[1] A. Ranganathan, S. Chetan, J. Al-Muhtadi, R. H. Campbell, and M. D. Mickunas. Olympus: A high-level programming model for pervasive computing environments. In *PERCOM'05*, pages 7–16, 2005.

[2] R. Grimm. One.world: Experiences with a pervasive computing architecture. *IEEE Pervasive Computing*, 3(3):22–30, 2004.

[3] W. Jouve, J. Lancia, N. Palix, C. Consel, and J. Lawall. High-level programming support for robust pervasive computing applications. In *Proceedings of the 6th IEEE Conference on Pervasive Computing and Communications (PERCOM'08)*, pages 252–255, mar 2008.

[4] iPhone SDK, http://developer.apple.com/iphone/program/download.html.

[5] J. J. Barton and V. Vijayaraghavan. Ubiwise, a ubiquitous wireless infrastructure simulation environment. Technical report, Hewlett Packard, 2002.

[6] E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, and D. Pesch. A testbed for evaluating human interaction with ubiquitous computing environments. In *TRIDENTCOM '05. Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, 2005.

[7] R. Morla and N. Davies. Evaluating a location-based application: A hybrid test and simulation environment. *IEEE Pervasive Computing*, 3(3):48–56, Juil-Sep 2004.

[8] V. Reynolds, V. Cahill, and A. Senart. Requirements for an ubiquitous computing simulation and emulation environment. In *InterSense '06. Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks*, 2006.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6907
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6907