Chapter 33
Creating, Debugging, and Testing Mobile Applications with the IPAC Application Creation Environment

Kostas Kolomvatsos
National & Kapodistrian University of Athens, Greece

George Valkanas
National & Kapodistrian University of Athens, Greece

Petros Patelis
National & Kapodistrian University of Athens, Greece

Stathes Hadjiefthymiades
National & Kapodistrian University of Athens, Greece

ABSTRACT

An important challenge in software development is to have efficient tools for creating, debugging, and testing software components developed for specific business domains. This is more imperative if it is considered that a large number of users are not familiar with popular programming languages. Hence, Application Creation Environments (ACEs) based on specific Domain-Specific Languages (DSLs) can provide an efficient way for creating applications for a specific domain of interest. The provided ACEs should incorporate all the functionality needed by developers to build, debug, and test applications. In this chapter, the authors present their contribution in this domain based on the experience of the IPAC system. The IPAC system provides a middleware and an ACE for developing and using intelligent, context-aware services in mobile nodes. The chapter fully describes the ACE, which is a key part of the overall architecture. The ACE provides two editors (textual, visual), a wide functionality spectrum, as well as a debugger and an application emulator. The ACE is based on an Application Description Language (ADL) developed for IPAC. The ADL provides elements for the description of an application workflow for embedded systems. Through such functionality, developers are capable of efficiently creating and testing applications that will be deployed on mobile nodes.

DOI: 10.4018/978-1-4666-4301-7.ch033

INTRODUCTION

In Computer Science applications, software components should be developed in order to provide more intelligence in the produced systems. However, users, lacking experience with programming languages are not able to write productive software components.

In such cases, *Application Creation Environments* (ACEs) play a critical role in building and testing software components. With these tools, developers can efficiently design software components as they can utilize a number of editing facilities without the need of using a conventional programming language. In general, ACEs contain: a) a source code editor, b) a debugger, and c) application building automation tools. Software components are developed for a specific domain. Therefore, the vast majority of programming languages do not provide an efficient solution. In this case, *Model-Driven Engineering (MDE)* (Schmidt, 2006) can provide a number of advantages. MDE is a software development methodology, aiming to increase efficiency in developing applications for a specific domain, through the creation of appropriate models. *Domain-Specific Languages (DSLs)* (Mernik et al., 2005) follow the principles of MDE development and can provide a number of advantages in cases of limited programming knowledge. A DSL is a language designed to solve problems that arise in a particular field of application, targeting more specific tasks than classic programming languages. DSLs provide the means for describing parameters for a domain of interest having a concrete syntax. Several semantic models are used for the description of the problem. These semantics lead to the automatic generation of specific tools used for the creation of the final code, which could be in a general purpose programming language (e.g. Java). In DSL tools, there are specific methodologies for the definition of the semantics of each language. Compared to general-purpose languages they offer better expressiveness in their specific focus domain. The most significant advantage of DSLs is that they provide users with the capability to write domain specific programs more easily. These programs are independent of the underlying platform, which is another advantage.

However, developing applications with a DSL also presents some disadvantages. The most important is that there are not any commonly used debuggers for DSLs. A primary reason for this fact is that DSLs are oriented to specific domains and generic debuggers cannot be used. Hence, the development of debugging facilities for DSLs is necessary. Based on such tools, users will be capable of debugging the source code of their applications. The debugger should be DSL-oriented, covering all of the language elements. Another disadvantage is that applications developed for embedded systems should be emulated prior to final deployment. Through emulation, developers can identify possible errors, as well as performance issues. However, as in the case of debuggers, there are not any generic emulators that can be adapted to every DSL.

In this chapter, we present our system for creating and testing applications developed with a DSL, fully integrated into ACE functionalities. The *Integrated Platform for Autonomic Computing* (IPAC) (Tsetsos et al., 2010) ACE provides two editors: a textual, and a visual editor. Each of them provides a number of functionalities for the application creation process. Such applications are created for deployment in mobile nodes. The provided ACE aims not only to experienced developers but also to non – experienced programmers. The IPAC ACE depends on an *Application Description Language* (ADL) created in the framework of the IPAC project. The ADL is the basis for the creation of a number of s/w productivity tools. We describe the code generation component responsible for producing the target code in a certain language (e.g., Java). We developed the IPAC *Debugger,* responsible for accepting logging messages that follow a pre-defined format and present them in a user friendly interface. Such logging messages