Preface

It has been more than a decade that Salesforce and Amazon delivered the first on-demand services on the web, starting an age of metered IT services provided to consumers at the pace they need them. Cloud computing is defined by the U.S. National Institute of Standards and Technology (NIST) as a model to enable ubiquitous access, convenient and on-demand to a shared pool of configurable IT services (e.g.: networks, servers, storage, applications and services), and where services can be rapidly provisioned and released without intervention of the service provider and with minimal management effort (Mell & Grance, 2011). This definition entails the unilateral provision of computing services as needed by the client, without intervention of the service provider; the access to the provided services through standard mechanisms and by disparate client platforms; the elastic provision of services, which may scale up and down or in and out, depending on consumer’s demand; the use of a metering capability at a level of abstraction appropriate to the type of service; and, the pooling of computing resources, by the provider, so that several consumers may be served, according to their needs, using a multi-tenant model.

To companies, the arrival of the cloud concept facilitated their internal organization and management as they no longer need to highly invest in resources to achieve their goals. The pay-as-you-go model and the utility computing concept of cloud computing allows companies to use resources as required, avoiding bad and unused investments. The fundamental attributes used to characterize the cloud correspond, at the same time, to the benefits it provides to its users: on-demand computing model, which delivers IT resources to end users in a similar way public utilities - electricity or water – are delivered eliminating the prerequisite of each company needing to have its own datacenter; autonomous services as they hide, from end users, their complexity or technology; predefined QoS to service usage are defined in service-level agreements (SLA) and made clear to clients when they contract them; internet-based meaning all services are delivered through the Internet and publicly accessible; services are easy-to-use through friendly interfaces and GUI forms for administrators; the resources contracted are scalable up and down, or in and out, depending on the organization needs at a given moment; inexpensive solution compared to having their own datacenters and subscription-based model that allows clients to choose what services they are interested in (Hassan 2011).

Likewise, less than a decade ago, the arrival of iOS and Android mobile operating systems, together with the massive deployment of mobile high-speed data networks, such as 3G and 4G, and the proliferation of mobile smart devices (Islam & Want, 2014), enabled the advent of an age where people may be always connected, carry computing power constantly, and IT services may be accessed from almost anywhere, anytime, for both personal and corporate purposes. Over time, mobile technologies have proven to be of great importance for organizations providing mobility, flexibility, reactivity and productivity increase. The wide range of mobile devices and associated technologies and features brought
additional challenges to mobile development. Mobile programmers need to address several issues to ensure a reliable and trustable application including portability, CPU and memory consumption, battery usage, seize devices heterogeneity, data input mechanisms, Wi-Fi or 3G/4G communication and built-in storage. When creating new mobile applications, companies need to target at least the two major mobile operating systems currently in the market. The great disparity in the programming languages used, UI guidelines and the IDEs themselves, make companies face the challenge of choosing the best approach that allows them to have a mobile application targeting more than one platform with a minimum cost, raising the question of when to use native or cross-platform environments.

All these developments have brought mobile and cloud based applications to the fore. As these are gaining momentum, by getting more and more integrated with business process models and expanding in scope and usage, they also become more valuable to businesses. This amplification in business value, together with the corresponding increase in complexity, has led to the development of new software engineering approaches specialized for these kinds of software applications. Existing software engineering methods and techniques have been adapted, and new ones have seen the light of day, so that business processes can rely on mobile and Cloud applications and services, trusting in their ability to respond to business needs, their quality of service, availability and performance.

Modern Software Engineering Methodologies for Mobile and Cloud Environments address mobile and cloud applications and services design and development. They encompass a set of methods and techniques for modeling, constructing and validating mobile and cloud-based applications, analyzing and testing underlying networks and services, and studying and developing new approaches that try to cope with the limitations of one through exploring the opportunities leveraged by the other. These methods and techniques are mainly specializations of existing approaches for desktop and web applications, but also include new proposals specifically tailored for mobile and cloud applications and services. Examples of these latter are domain specific languages for modeling and describing mobile applications’ user interface or access to location-based services, approaches to multi-tenancy adaptation of legacy applications redeployed to the Cloud, or new distributed architectures for mobile-cloud applications. These are amongst many other examples.

The mission of this book on Modern Software Engineering Methodologies for Mobile and Cloud Environments is to discuss and analyze current methodologies, techniques and practical solutions and trends related to this new field of knowledge, in a comprehensive way.

The overall objectives comprise:

- To discuss the importance and the challenges associated to software engineering methods and techniques for mobile and cloud environments;
- To analyze the state-of-the-art of existing software engineering methods and techniques for mobile and cloud environments;
- To propose emerging technological developments and practical solutions for engineering software for mobile and cloud environments;
- To present case studies of application of software engineering methods and techniques specialized for mobile and cloud environments.

Potential contexts of use of the present book include the scientific community (scholars and researchers) with research interests in these fields. Moreover, information technologies companies and practitioners,
and graduate and post-graduate level students, can benefit from the advances presented in the book and apply them in real applications and scenarios.

This edited book comprises contributions from several authors focusing the most recent progressive research concerning software engineering methods and techniques applied to the development of mobile and cloud applications and of cloud services-based applications. The book intends to cover the following topics:

- Mobile and cloud technologies and characteristics
- Software architectures for mobile cloud computing
- Collaborative cloud and mobile applications
- Mobile and cloud design patterns
- Modeling of mobile and cloud applications
- Software Testing as a Service
- Polyglot programming
- Model-driven mobile development
- Native and web-based applications
- Energy efficiency in mobile development
- Usability in mobile applications
- Indoor tracking and localization
- Cross-platform mobile development
- Software sizing for mobile and cloud projects
- Mobility management and protocols

**ORGANIZATION OF THE BOOK**

The book is organized into fifteen chapters. A brief portrayal of each of the chapters follows:

Chapter 1 reviews Mobile computing and Cloud computing, two of the most growing technologies in number of users, practitioners and research projects. The chapter surveys mobile and cloud technologies and presents their evolution and characteristics. Then, building on mobile devices limitations and mobile apps increasing need of resources, and on the cloud computing ability to overcome those limitations, the chapter presents Mobile Cloud Computing (MCC), and characterizes it by addressing approaches to augment mobile devices capabilities. The chapter is settled after some views about future research directions and some concluding remarks.

Chapter 2 outlines existing software architectures for Mobile Cloud Computing, and presents Artificial Bee Colony (ABC) as a nature inspired architecture for providing reliable cloud services to mobile requests. The proposed approach intends to pave the way for timely services by using three different agents working in parallel, mimicking the behavior of honey bees namely Employed Bees, Onlooker Bees and Scout Bees. The approach also intends to be seen as a green IT solution for MCC based software Development.

Chapter 3 debates about the capabilities of mobile devices for supporting collaborative applications. Despite the enormous progress in the capacity of computing, storage and data visualization experienced by mobile devices, these still face some difficulties related to energy consumption and the mobile network coverage. This poses serious problems against an efficient and continuous use of mobile collaborative...
Preface

applications and a great challenge for their designers and developers. The author describes design patterns that help modeling mobile collaborative applications to support collaboration through the cloud. Two levels are presented: the first level provides self-control to create clones of mobile devices, manage users’ groups and recover failed clones in the cloud. The second level supports group collaboration mechanisms in real-time.

Chapter 4 addresses estimation of cost, effort and schedule as very important aspects in commercial software development. The authors identify effort as typically being the predominant cost driver in software development, and the size of the software to be developed, as the dominant determinant for effort. Then, the COSMIC method is presented as a standardized technology independent measure for estimating the functional size of software. Finally, the authors argue that the COSMIC method provides a good basis for the estimation of cost, effort and schedule in mobile and cloud-based applications.

Chapter 5 reflects about Software Testing as a Service (STaaS) and the conditions offered by the cloud for supporting it. The authors consider the cloud-based hardware infrastructure and computing resources to perform traditional testing like performance, load, stress, security and compatibility testing for regular on-premises applications. Their opinion that software testing in the cloud (cloud testing) has the capacity to change the way software testing is performed, is however threatened by the fact that the cloud itself poses many issues related to security, compatibility, performance, scalability, functionality, third party dependencies and many more. And, according to the authors, these issues need also be tested. Testing cloud infrastructure and applications has its own peculiarities that demand for novel/tailored testing methods and tools. The chapter distinguishes between ‘cloud testing’ and ‘testing the cloud’, debates why ‘testing the cloud’ is so crucial, and explains the need, objectives, requirements and challenges of ‘testing the cloud’.

Chapter 6 presents polyglot programming, a technique where several languages are used in the creation of a single application. Different programming languages have been designed to solve problems efficiently in different domains, and polyglot programming tries to combine and utilize the best solutions from different programming languages and paradigms in a seamless fashion. In this chapter, the authors examine polyglot programming in the context of web applications, where it has been commonly used to create compelling applications, but where, according to the authors, there is still considerable potential to improve development in various ways.

Chapter 7 debates the growing need for collaborative platforms for software modelers to work together. It also analyses the challenges posed to solutions for the collaborative model-driven development of complex systems, which typically involve many stakeholder groups working in a coordinated manner on different aspects of a system. The authors also present the architecture of their prototype of a cloud-based multi-view modeling environment, AToMPM.

Chapter 8 reflects on the main characteristics of mobile applications and mobile devices, since they have a great impact on mobile applications. It also presents the classification of mobile applications according to two main types: native and web-based applications. Then, the authors identify the most relevant types of quality attributes for mobile applications, and show that the relevant quality attributes for mobile applications are usually framed in the Usability, Performance, and Maintainability and Support categories.

Chapter 9 discusses the limited resources of mobile devices when compared to personal computers, especially energy. In mobile devices, battery consumption is always under the user’s eye. The authors also argue that object-orientation, which is currently the main paradigm in applications’ development, has some inherent features that consume important amounts of energy in the context of mobile development.
According to the authors, these are however needed features, which are responsible for offering maintainability and flexibility, among other software quality-related advantages. The chapter aims to present an analysis to evaluate the trade-offs between object-oriented design purity and battery consumption, so that developers can design mobile applications taking into account these two issues, giving priority to object design quality and/or energy efficiency as needed.

Chapter 10 presents the results of testing the usability of a web-based learning management system (LMS) within Android mobile devices. The authors claim that usability testing is vital as the user usability satisfaction is the leading point in market for mobile web applications. Ten usability factors have been chosen, on the basis of which proper experimentation by users accessing the LMS through their smart phones is conducted. Initial requirements for the experimentation were measured in a pre-test process, after which the detailed experimentation has been performed to measure the usability factors presence in the LMS when accessed through android based systems. The results are statistically summarized.

Chapter 11 proposes a domain independent Pedestrian Dead Reckoning System that can be applied to any indoor environment. The chapter describes the adopted architecture and the developed prototype. The authors state that within the proposed system, the user can create new indoor spaces, define reference points in it, positions for future access and also track his current location. The chapter presents some solutions for solving several walking detection false positives, in order to track the user’s position. The results of conducted tests are also presented, and show a 98% accuracy of the system when tracking the user’s current position.

Chapter 12 proposes a model-driven development approach for generating data-oriented mobile applications from a domain model, which represents the information structure of the application domain. The authors present an approach for generating a platform independent user interface model from a domain model, and then propose to have different code generators for each different target platform. The approach responds to the increasing adoption of mobile smart devices, the demand for a growing number of mobile applications, and the need to deploy each mobile application to different mobile platforms, such as Android, iOS or Windows. The approach presented is applicable to data-oriented mobile applications. A generator prototype for Android apps is also presented.

Chapter 13 also tries to respond to the increasing demand of mobile applications, and the need to deploy each mobile application to different mobile platforms. The authors propose to combine techniques based on MDA (Model Driven Architecture) with the HaXe language. The main ideas behind MDA are separating the specification of the system functionality from its implementation on specific platforms, managing the software evolution, increasing the degree of automation of model transformations, and achieving interoperability with multiple platforms. HaXe is a modern high level programming language that allows generating mobile applications that target all major mobile platforms. The main contributions of this chapter are the definition of the HaXe metamodel and the specification of a model-to-model transformation process between Java and HaXe. These contributions enabled also the definition of an MDA migration process from Java to mobile platforms.

Chapter 14 presents the notion of model execution within the model-driven engineering field. Model execution is about interpreting the model through a dedicated execution engine instead of executing code generated from the model. The modeling languages able to be executable are called i-DiML (interpreted Domain-Specific Modeling Languages). The authors assert that model execution tends to abolish the boundaries between modeling and programming. This chapter then illustrates how parts of an Android mobile application can be modeled and executed by leveraging a well-known i-DiML, namely UML 2 State Machine Diagrams and the PauWare engine. The proposed installation of PauWare on Android
Preface

OS sets up the foundation for a whole range of mobile apps, provided that they are modeled with State Machine Diagrams.

Chapter 15 surveys mobility management and protocols, which play an important role in heterogeneous networks in providing a seamless mobility support for both devices and users. The chapter then presents SIP-PMIP Cross-Layer Mobility Management in order to provide a seamless mobility in heterogeneous wireless networks. In effect, the authors propose a Cross-Layer Mobility Management Scheme, which can handle terminal, network, personal and session mobility. To demonstrate, video conferencing is included in the modeling, simulation and implementation of the module using Riverbed Modeler.

António Miguel Rosado da Cruz  
Instituto Politécnico de Viana do Castelo, Portugal

Sara Paiva  
Instituto Politécnico de Viana do Castelo, Portugal

REFERENCES

