

# XModel: an Unified Effort Towards the Development of High-Quality Mobile Applications

Érika Cota, Luigi Carro, Lucio Duarte, Leila Ribeiro, Flávio Wagner

PPGC - Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

Po Box 15064 - Porto Alegre, RS, Brazil  
{erika,carro,lmduarte,leila,flavio}@inf.ufrgs.br

## ABSTRACT

This paper proposes a model-based line of research and education for establishing new development approaches for mobile applications, where several non-trivial quality aspects of the product must be considered. We first detail our view of the main requirements for a mobile design methodology and discuss why traditional software engineering processes fail to address such requirements. In short, we believe the hierarchical view of a mobile system is not actually available to the software developer and this precludes productivity and overall quality of the resulting systems. We then present our view on how new design methodologies should tackle the existing challenges and actually provide design and implementation layers that can improve productivity and quality. The proposed line of action is based on the definition of appropriate high-level models and on multi-disciplinary knowledge, shifting the focus of education and research to topics that are currently marginal in the software engineer curriculum, such as optimization theory, model-based design and verification.

## 1 Mobile System Development Paradox

Mobile applications seem to converge to a generalization-customization paradox. To see that, let us consider the classic layered view of an electronic-based system, which includes a Business layer, a Software Development Kit (SDK) layer (development software), a Hardware-dependent-Software (HdS) layer and a Hardware layer (execution platform), where one layer is positioned above the next in the mentioned order, creating a hierarchical structure. Unfortunately, this hierarchical view of the execution and development of an embedded system is realistic only for a few classes of applications, namely, the ones with very specific requirements, usually having a single target function, such as automotive supporting systems, instrumentation, etc. Embedded systems targeted to the mass consumer market or, currently, mobile systems, have a more complex structure, as shown in Fig. 1. The underlying concept is the *variability* present at all levels. The reality that is not represented in this view is that hierarchy is indeed accomplished in the execution stack, but not in the development process, which is still "monolithic" in many aspects.

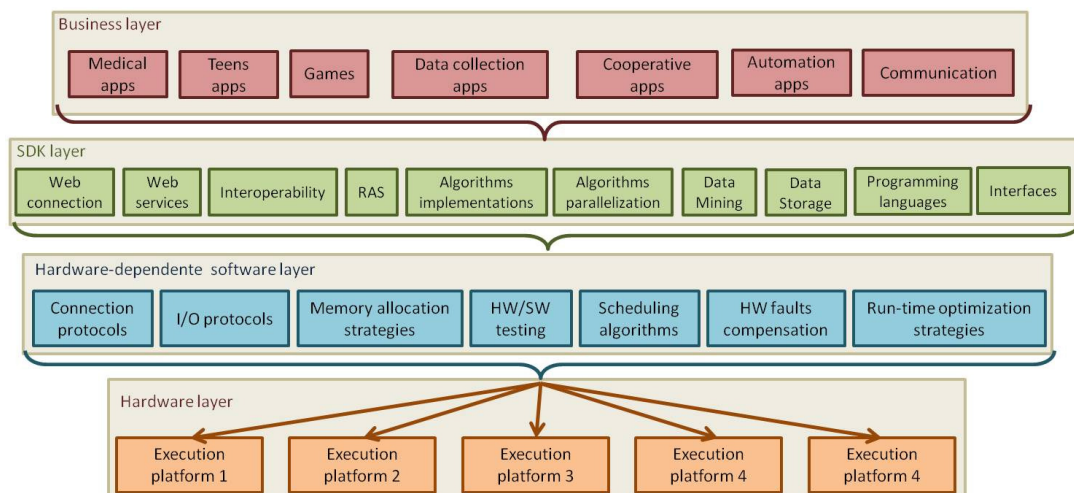


Figure 1: Embedded systems organization

In our view, *actual hierarchy* in the development process is the key to achieve high-quality mobile systems (for any definition and metrics of quality) but current practice of software engineering fails to support this development paradigm. In this paper we elaborate this view and present a possible research and education path towards the goal of a systematic, hierarchical, and flexible approach to mobile application development. Mobile applications are based on and driven by variability. Multiple execution platforms and all its variations (hardware layer) provide a broad range of computational power and development costs, thus allowing the use of an electronic-based system in a broad range of application fields (business layer). Similarly, computer science fields are in constant evolution towards the development of new technologies for the HdS layer (e.g., communication, optimization, and parallelization) and even more for the SDK layer (e.g., algorithms, and data manipulations). This variability at all levels potentially gives the system designer the necessary flexibility and adaptability to design a cost-effective solution that fits the specific application requirements and constraints.

On the other hand, programming within this huge design space can be cumbersome without levels of abstraction. Indeed, mobile software development is making use of abstraction layers. The hardware layer is accessible through development kits and devices drivers provided by specific implementations of the HdS layer (different embedded operating systems and device drivers compiled to specific platforms). The concept of software platforms has also been used to leverage many of the challenges of the development of embedded software, which are normally related to the interaction with the lower layers. Software platforms, such as Android [1], make it easier for an ordinary programmer to access the resources of the hardware platform and interact with other applications or system resources as well. Cross-platform development environments, such as RhoMobile Rhodes [2] and PhoneGap [3], go one step further and help the developer deal with the hardware variability.

From the concept of design platforms (hardware and software) one can conclude that embedded systems tend to be based on generic solutions that are built to be customized at some point later in the development cycle. This approach is in line with the market trend of the mobile field where generic products are offered and further customized by end users according to their own needs. At this point, one can classify the mobile software designer in two groups: the *software platform provider* and the *application developer*. The first group is responsible for the implementation of the services provided by the HdS and SDK layers whereas the other group implements the final applications that will actually interact with an end user. Both groups develop mobile applications, but each one requires different backgrounds.

The application developer customizes a device by developing very specific applications and using the programming resources provided by the software platform. The concerns of an application developer should focus on the business and end-user logic, and not on the translation of this logic into, for instance, a power-efficient, low-level code that must execute over multiple target hardware. For this to be possible, the resources of the development platform should hide the details of the actual implementation and concentrate on the behavior or operations that actually represent the business model. As a consequence, the platform provider must focus on providing a development environment that can both capture the needs of the end user programmer and generate a high quality running code. This group should be concerned with the definition of mechanisms to automate the translation of the services provided by the HdS layer to the plethora of execution platforms available, and the adaptability of those services with respect to the constraints of the execution process (performance, memory, power-management, etc). This group must also find appropriate abstractions to create a high-level access for the resources at each level, in such a way that those can be used by an application developer that is not a computer science specialist. More than abstractions, the software platform must also take advantage of the constant new advances and variations of the lower-level layers to ensure levels of performance, reliability, availability, and overall system quality. For instance, services provided by the platform must be power-efficient, which can change according to the use of the service or even to the data. The platform must also be flexible to easily accommodate new resources or variations in the hardware and future application needs.

We advocate thus that software engineering for mobile systems should focus on the implementation of flexible and hierarchical features in the SDK and HdS layers, which is only

possible if the lower-level layers also provide the correct abstraction and information. Current practice, however, requires that each type of developer accumulates concerns and knowledge of different layers. This implies a longer learning curve for the use of a platform, and may preclude the implementation of high-quality software, since many decisions are taken without proper knowledge and information. Moreover, many decisions taken during the development might need to be adapted during execution or during the application lifetime. The current design model does not support this level of flexibility. Even higher-level platforms such as PhoneGap or Rhodes still assume the application developer has programming and computing knowledge to develop their application. Hence, it seems to us that the developers of the software platforms (HdS and SDK layers) is the one who needs a strong computing and engineering background, so that they can be able to provide an abstract framework of high quality for mobile customization. *Supporting mechanisms for the development of high quality embedded software development frameworks is thus the focus of our research group.*

## 2 MAIN CHALLENGES AND PROPOSED APPROACH

Current software development platforms are tied to a single execution model and hardware platform. Therefore, the end-user developer can usually provide and support their application for a single or very few execution platforms. A few platforms claim that a single software description can be deployed to multiple execution platforms. However, such frameworks are still based on an abstraction level rather low when compared to the needs and the programming skills of the typical application developer. Hence, there is a good chance the developer will not use the full potential of the platform, mainly due to misinformation and lack of deeper knowledge, not to mention the often required tuning to deal with the constraints of the execution platform (power efficiency, restricted memory, etc). Moreover, support for testing, performance analysis, and overall quality evaluation of the resulting application is still very limited, and also requires expertise from the application developer in those topics. For instance, mistakes due to the misuse of available APIs can be hard to be detected. Such mistakes may or may not lead to system failure, performance problems or other non-functional issues, such as power consumption beyond an acceptable range. It is expected that the end-user developer has no freedom (and/or knowledge) in tuning the platform behavior for a given application. Indeed, such a task must be implemented by the platform, in many cases during runtime. Current platform models have little support to such an adjustment.

The keyword for mobile software development is *flexibility*. Flexibility allows the system to cope with the variability in many levels (of the hardware, of the application needs, of the programming languages, etc). It has been long known that one achieves flexibility by *modularization* and *abstraction*. In this aspect, current knowledge of software engineering remains valid. *Reuse* is another important concept, which is also already present in the current development models. These concepts potentially imply higher-quality software, but this must be ensured by methods and tools. What is still missing in the mobile field is the *quality-ensurance-by-methods-and-tools* concept. Much has been said and done about software quality, but this is still not in the mainstream in the mobile field. The second concept that is still missing is *flexibility-in-the-long-run*. How can the platform be flexible enough to realize a change in the application behavior and/or in the hardware behavior?

In this context, we can define five main challenges for the development of a high-level and high-quality mobile software development platform:

1) Definition of appropriate abstractions for each development layer in such a way that: i) the developer of the higher layers can use the provided services without further knowledge; ii) interaction between layers is facilitated to support variability and provide flexibility;

2) Definition of synthesis processes capable of transforming the semantics of the end-user application into cost-effective runtime code while taking into account the non-functional requirements of the application and of the execution platform;

3) Support for application testing and fault-tolerance mechanisms, including mechanisms to deal with uncovered faults in the platform (hardware or software) and misuse of the platform API;

- 4) Capture some hardware-related issues that have deep impact on the software development process, such as the availability of multicore devices and of distributed computing environments;
- 5) Support for system evolution in several aspects: execution platform, requirements, programming languages, and so on.

We propose a model-based approach to tackle the mentioned challenges. Current research and results on models are still limited to traditional applications and few tools are available for the support of a complete model-based design. Furthermore, software engineering models are still not capable of capturing behaviors or adapting to the variability in the lower levels. One must find the most suitable models to deal with the specificities of the mobile system: run-time adaptability, variability, constant system evolution, verification requirements, and so on. For instance, a successful model, such as UML, still seems under-utilized due to the lack of supporting automation tools that allow the interaction of the model with the actual runtime code. Our research group proposes a development cycle based on models as a means to achieve both flexibility and quality supported by tools. Models can be verified, can suffer transformations that can be tracked and verified according to different goals, and can serve as a basis for a number of analysis. In our work, we propose the use of models in every abstraction interface, which will require extreme modelling capabilities at all levels, hence the term XModel. Specifically, a research plan towards such a powerful development platform includes the following topics:

- Investigation of distinct abstraction models and the mechanisms for implementing model transformations that can be applied in the different design levels;
- Definition of a formal basis for the different models to enable verification and transformation processes;
- Co-synthesis of fault-tolerant systems, combining hardware and software techniques;
- Model-driven specification and management of platforms;
- On-line and off-line monitoring strategies for verification and optimization;
- Quality metrics applied to embedded software.

### **3 REQUIRED SKILLS FOR A SOFTWARE ENGINEER**

In order to provide adaptive development frameworks capable of abstracting its behavior in favour of expressiveness without losing the possibility of adjustment to the execution platform, a multi-disciplinary approach is required. It should combine traditional computer science knowledge (algorithms, data structures, network, compiling, operating systems, and programming languages) with additional knowledge that is normally not in the mainstream of a computer science course. This additional knowledge would be based on the following ideas:

- Model-based design must be the focus of education: i.e., the software engineer must be capable of defining and using different models as well as relating them through transformations;
- New abstraction models must be explored to capture the semantics of the end-user application. To this end, natural language processing, visual languages, and agent systems theory, for instance, can be considered;
- Verification, Analysis, and Testing concepts are required to support the software synthesis process and evolution;
- Optimization theory can be applied in the synthesis process to deal with the execution constraints and to the run-time adaptation;
- Fault tolerance is also a required knowledge for the platform provider as a means to cope with the escaped faults both in the hardware or in the softwaersoftware platforms;
- Metrics and assessment techniques, which are quite different in the mobile domain w.r.t. the traditional software development;
- Parallel programming for MPSoC platforms, since this looks like the future of mobile platforms.

### **REFERENCES**

- [1] Android Developer site. <http://developer.android.com>. Accessed on 16 September 2011
- [2] RhoMobile site. <http://rhomobile.com/>. Accessed on 16 September 2011
- [3] PhoneGap site. <http://www.phonegap.com/>. Accessed on 16 September 2011