A Model-Driven Approach for Mobile Business Information Systems Applications

Luís Pires Silva [1], Fernando Brito e Abreu [1,2], Vasco Amaral [2]

[1] QUASAR / ISTAR/ ISCTE-IUL, Av.ª das Forças Armadas, 1649-026 Lisboa, Portugal
[2] CITI, FCT/UNL, Campus da Caparica, Quinta da Torre, 2829-516 Caparica, Portugal

luis.ptds@gmail.com, fba@iscte-iul.pt, vasco.amaral@fct.unl.pt

Abstract.

Context: Mobile BIS apps demand is increasing, with shorter time-to-market requirements, but their production faces problems, such as handling business rules concurrently, multiple platforms, localization and extensibility.

Objective: Propose a generative approach for mobile BIS apps that will mitigate the identified problems.

Method: We adopted the Design Science Research methodology, that helps gaining problem understanding, identifying systemically appropriate solutions, and in evaluating innovative solutions.

Results: We identified the problem and its motivation, defined the objectives for a solution, designed and developed a prototype generative tool for BIS apps, demonstrated its usage and evaluated how well it mitigates a subset of the identified problems in an observational study.

Limitations: Several issues are pending such as distributed business rules enforcement and the formalization of the required transformations from the PIM to several platform-specific models (PSMs).

Conclusion: We intend to contribute for reducing BIS apps time-to-market, while improving the maintainability of those apps.

Keywords: Mobile apps, BIS, MDE, business rules, OCL, transformations, PIM, PSM, Design Science Research

1 Problem

The burst on the availability of smart mobile devices is powering a growing mobile business ecosystem [1]. In this ecological metaphor firms are part of a larger ecosystem, each playing a contributing role and forming symbiotic relationships with customers, suppliers, and competitors. This ecosystem is fueled by the emergence of an "App Economy", enabling new products and services, but also influencing strategies and shaping business models [2]. According to a software market diagnosis for the EU27 region, the apps share is the fastest growing one and will account for roughly half of that market by 2020 [3].

The expanding mobile ecosystem puts an increasing pressure in the demand for mobile business information systems (BIS) apps, therefore enforcing

short time-to-market requirements [4]. However, developing such applications is a challenging task, due to several technical hindrances. Some are generic, such as the need to support localization features, and others are specific to mobile devices, such as the ability to support the diversity of available deployment platforms (e.g. diverse screen sizes, resolutions and orientation), as we had the chance to experience on our previous work [5, 6].

Those mobile BIS apps must perform distributed business constraints handling, since they will be used concurrently by users in both connected and disconnected modes. Furthermore, since business rules change frequently, it is important to make them as much detached as possible from the source code, for maintainability sake. The aforementioned scenario calls for cost-effective techniques to generate mobile BIS apps. This paper proposes a model-driven generative approach that is expected to mitigate the aforementioned problems.

To guarantee business continuity, BIS apps must work online and offline. However, since multiple users will be working concurrently, business constraints violations may arise when switching from offline to online (P1). These business constraints are usually buried in the BIS apps source code, thus increasing maintenance costs and fault proneness (P2). Mobile platforms are very diverse, in terms of size and resolution, requiring BIS apps to perform a dynamic reconfiguration of their user interfaces (P3). The usually small size of those interfaces may be a hindrance in BIS apps, especially when the problem domain is complex, due to the multitude of concepts and their interrelationship that must be handled through their GUI (P4). Users should also be able to work in their native language, but we cannot forecast, at design time, which will be required (P5). Last, but not the least, automatically generated software systems are usually hard to understand, extend and integrate with existing systems, as recognized in a recent survey on MDE adoption in industry [7] (P6).

2 Related work

A good starting point were some systematic secondary studies on model-driven engineering (MDE) generative approaches [7, 8]. There, we could confirm our suspicions that, while many MDE code generation endeavors have been carried out during recent years, most seem to be small-scale studies. We could not find any generative approach supporting distributed constraints satisfaction in the context of mobile BIS apps. Still, we could find evidence of some related approaches targeting comprehensive BIS apps generation.

The Naked Objects pattern, initially defined in [9], advocates that all business logic should be encapsulated onto the domain objects. It also recommends that the user interface should be a direct representation of the domain objects, with all user actions consisting, explicitly, of creating or retrieving

domain objects and/or invoking methods on those objects. The user interface should be created 100% automatically from the definition of the domain objects, for instance, using reflection techniques. There are some code generation tools based on the this approach, such as the *Naked Objects* for *.NET* [10] or the *Apache Isis* for *Java* [11]. Both tools follow the same principle, which is providing automatically a strong base structure, where the programmer can then specify directly in code the domain model and reach other layers through annotations. Another example is *JMatter* [12], a software framework, also based in the Naked Objects pattern, for constructing workgroup business applications, where the domain model can be specified in UML trough *Ultraviolet*, a light UML editor [13]. Regarding the visualization, they are all very similar, being the main difference that the first two are web oriented, and *JMatter* produces a *Java* GUI environment. All aforementioned examples seem to follow a table oriented view style to represent entities and their relationships, which could became a hindrance on small screen phones, due to space restrictions.

None of the aforementioned generative approaches seem to have synchronization concerns, since they operate within the server database and therefore only solve direct concurrency problems. For instance, the *Apache Isis* targets web apps, server connection is considered to be always available, so a *RESTful* standard approach is implemented [14]. Furthermore, both *Naked Objects* and *Apache Isis* apparently offer the most complete and maintainable approach, since they are based on a reflection type approach. Such a feature allows the developer to fill in the gaps and still make use of the provided API. Another drawback of the previous approaches is that simple naked objects are unable to convey all the abstractions and characteristics of a complex UI, namely its composition and behavior (e.g. navigation paths) [15, 16].

3 Proposed solution

To mitigate the aforementioned problems (P1 through P6) we propose applying MDE techniques to automatically generate fully functional mobile BIS apps with a sound, maintainable, architecture. MDE is about the systematic use of software abstractions – or models – as primary artifacts during a software engineering process [17]. Our generative approach builds upon the following mainstays: (i) *Model centered generation* – everything follows the model, from GUI navigation to persistence; (ii) *Separation of concerns* – there is a clear separation in layers or components, each encapsulating a concern of its own (e.g. presentation, persistence, synchronization); (iii) *Paradigm seamlessness* – the object paradigm is used throughout; by using the same type system, we avoid type conversions (e.g. from object to relational and vice-versa) that hamper maintainability.

Our input is a PIM expressed as a UML class diagram enriched with design-by-contract constraints expressed with OCL. The model is further annotated for expressing presentation and synchronization rules, also in a platform-independent fashion. The non-functional requirements for the generated BIS apps include local persistency capabilities, required for offline usage, and distributed synchronization capabilities to guarantee overall system state consistency. Other required "qualities" with which we are concerned, include portability and maintainability. The aforementioned synchronization across multiple users requires a coordination server to guarantee that business rules consistency is kept system-wide. This issue is similar to the distributed constraint satisfaction problem that has been addressed in the AI field [18] and is probably the greatest challenge to be faced in this research work.

The output of our generative approach is supposed to be a fully functional BIS app for a given target platform. To address this desideratum, we plan to apply language engineering transformation techniques [19] in two steps. In the first step we will go from the PIM to a PSM, corresponding to the desired platform. In the second step we will take as input the PSM and generate source code for the client-side. The server-side code generation process is performed directly from the PIM, since it has no dependencies on the mobile platform used.

4 Preliminary work

In the context of his MSc thesis, the first author, supervised by the second author, has developed the *JUSE4Android*[1] tool. Its main goal was to reduce the development effort of *Android* BIS apps by means of a PIM centered generative approach [5]. The PIM is a UML class diagram, enriched with annotations that set the rules for the UI and persistency layers.

The generated apps follow a model-based navigation scheme, support local persistency, all the CRUD operations and are also capable of simple synchronization. We used DB4O, an object-oriented database management system (ODBMS) that has features that make it particularly suitable for mobile platforms [20], besides being more efficient than relational counterparts [21]. DB4O allowed to avoid unnecessary transformations from the object-oriented type system of the host language (*Java*) to the more recurrent relational model used for persistence in *Android* apps. The resulting code is more maintainable due to this paradigm seamlessness.

The followed model-based navigation scheme allows us to support, in an easier way regarding layout structures, several screen sizes, since we only show one entity type at a time. In order to see associated objects, the user

[1] https://code.google.com/p/juse4android/

must navigate by means of an offered navigation bar. By following the master detail flow design pattern it is possible to accommodate smaller phone screens by showing either the master, or the detail screens. If the screen is large enough, both are shown at the same time. Regarding screen size recognition, adaptation and elements sizing, we use *Android's* size qualifiers when creating the folders which will contain the generated XML files that describe the layouts and views, so that the *Android* operating system is able to select automatically the required ones, depending on the characteristics of the current device. We followed the same generative principle for all the layers (except the model layer), by making every possible view or class as much independent of other entity types as possible.

JUSE4Android uses the UML Specification Environment (*USE*), developed at Bremen University[2], which is used to parse the supplied PIM and validate it. The *JDOM*[3] package was also embedded into the *JUSE4Android*, since it provides a simple API to read, manipulate and create XML type files. It follows the visitor pattern to facilitate extensibility, being the model classes the common argument of the visitor. For each supplied PIM, *JUSE4Android* generates, two applications: one for the client-side and another for the server-side. The client-side has several layers: the *View layer* where the views are describe in XML files; the *View-Model layer* (control layer) which acts as a middle man between the *View* and *Model layers* and controls and reacts to user inputs; the *Model layer* holds the domain logic, as described in the domain model; the *Persistency layer* offers the required API to communicate with the DB4O persistency engine; and lastly the *Synchronization layer*. The *Model layer*, besides the usual getters and setters, provides CRUD operations for all navigations that are possible to perform from a given domain entity, and was built upon the experience gathered in the open-source project *JUSE for Java*[4] developed for educational purposes in the context of MDE, by the second author of this paper.

5 Expected contributions

Business rules can be specified with OCL in the PIM and then compiled by the *USE* component. We will survey existing proposals to generate *Java* code from OCL business rules before implementing our own, to obtain the best of the breed. Here we plan to deliver (i) a systematic review on OCL to *Java* transformation and (ii) a new version of the *JUSE4Android* tool, capable of generating the code for business rules enforcement upon a local DB4O data-

[2] http://sourceforge.net/apps/mediawiki/useocl/
[3] http://www.jdom.org/
[4] https://code.google.com/p/j-use/

base instance. That code can be embedded in the client-side *Android* BIS app, required for offline work, or in the one running in the cloud server.

We will research the problem of distributed constraints satisfaction, through another systematic review, which will provide a better understanding of the advances in the field and its players, along with corroborating the research niche where our expected contributions will fit. We expect to devise a solution that will allow orchestrating business rules enforcement, namely resolving inconsistencies that arise when concurrent users of the mobile BIS apps shift from offline to online usage. The existing DB4O synchronization engine component, which we tested, only provides distributed data updates, therefore not guaranteeing the required business rules consistency. This is probably the biggest research challenge to be faced in this project and the expected results will be: (i) a systematic review on distributed constraints handling, (ii) our proposed approach for distributed rules orchestration, and (iii) a new version of the *JUSE4Android* tool supporting that orchestration.

We aim at generating mobile BIS apps for other mobile platforms. Although there are *Java* virtual machines (JVMs) available for all of them, the rendering mechanisms for the GUIs are platform-specific. We plan to apply language engineering transformation techniques [19] in two steps, to address this issue. In the first step (model-to-model transformation) we will go from the PIM to a PSM, corresponding to the desired platform (e.g. *Android*, *iOS*, *Windows Mobile*). In the second step (model-to-code transformation) we will take as input the PSM and generate source code for the client-side (the BIS app running on the users' mobile device). The code generation process for the server-side will hopefully be performed directly from the PIM, since we cannot envisage dependencies on the platform. The portability of the DB4O component on all the required mobile platforms is also an issue here, but we expect the *Versant* open-source community[5] will find a solution for that problem. The expected outcomes of this thread are: (i) the formal specifications of the PIM to PSMs and PSM to code transformations and (ii) the new *JUSE4Mobile* tool, supporting those transformations. Several components of *JUSE4Android*, described in section **Error! Reference source not found.**, are expected to be reused here, with some adaptations.

6 Plan for evaluation and validation

To assess the outcome of the previous threads, we plan to conduct several qualitative and quantitative validation experiments. The latter will be performed for the two encompassed roles: the *developer* that generates mobile BIS apps and the *user* that installs and runs those apps. Those experiments

[5] http://community.versant.com/

will address different quality characteristics such as usability, maintainability, portability and efficiency, as defined in the *ISO/IEC SQuaRE* standard [22]. A usability validation experiment targeting the *user* role, in the context of the preliminary work, can be found in [5]. That experiment provided evidence that our model-driven navigation paradigm is easy to understand and promotes incremental conceptual learning. Since we will always make the tool available as an open-source project, we expect to collect usage data and feedback from those that will use it. That feedback will hopefully provide the ultimate proof of feasibility for our proposals. We will also strive at receiving feedback from the automated software engineering and MDE research communities, by preparing demos for presentation in scientific meetings[6].

7 Current status

We have proposed a MDE generative approach for mobile business apps that is expected to mitigate several problems such as guaranteeing distributed business rules fulfilment, support multiple platforms and handle localization, while reducing the development and maintenance effort. Its input is a PIM model, where business rules are expressed in OCL. The support to several target platforms will be achieved through transformation techniques, namely to fulfill the required GUI reconfiguration requirements which are platform-specific. A model-driven interaction paradigm is also briefly introduced. The generated mobile BIS apps have a layered architecture, with paradigm seamlessness, which is expected to increase program comprehension and maintainability. The planned timeline for completion is represented in Figure 1.

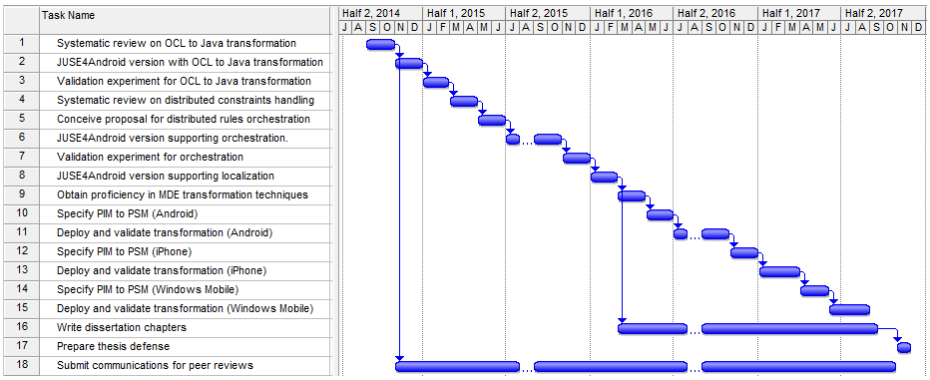

Figure 1 - Planned timeline for completion

[6] e.g. in http://ase-conferences.org/ or www.modelsconference.org/

References

1. Basole, R., Karla, J.: On the Evolution of Mobile Platform Ecosystem Structure and Strategy. Business & Information Systems Engineering 3, 313-322 (2011)
2. Page, M., Molina, M., Jones, G.: The Mobile Economy. Kearney, AT (2013)
3. Aumasson, A., Bonneau, V., Leimbach, T., Gödel, M.: Economic and Social Impact of Software & Software-Based Services. BE: European Commission (2010)
4. Parada, A.G., Brisolara, L.B.: A Model Driven Approach for Android Applications Development. Brazilian Symposium on Computing System Engineering (SBESC'2012), pp. 192-197, Natal, Brazil (2012)
5. Pires Silva, L.: A Model-Driven Approach to Generative Programming for Mobile Devices. DCTI, vol. MSc. Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal (2014)
6. Pires Silva, L., Brito e Abreu, F.: Model-Driven GUI Generation and Navigation for Android BIS Apps. In: Lisbon, P. (ed.) 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD'2014), pp. 400-407. SCITEPRESS Digital Library, Lisbon, Portugal (2014)
7. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. Proceedings of the 33rd International Conference on Software Engineering, pp. 471-480. ACM, Waikiki, Honolulu, HI, USA (2011)
8. Schieferdecker, I., Hartman, A., Mohagheghi, P., Dehlen, V.: Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. Model Driven Architecture – Foundations and Applications, vol. 5095, pp. 432-443. Springer Berlin Heidelberg (2008)
9. Pawson, R.: Naked objects. Department of Computer Science. University of Dublin, Trinity College (2004)
10. Pawson, R.: Naked Objects. IEEE Software 19, 81-83 (2002)
11. Haywood, D.: Apache Isis. (2012)
12. Suez, E.: JMatter. (2013)
13. Ramage, R.: umlc - UML modeling with code generation and UMLcompiler. (2006)
14. Fielding, R.T.: Architectural styles and the design of network-based software architectures. University of California, Irvine (2000)
15. Kennard, R., Steele, R.: Application of Software Mining to Automatic User Interface Generation. In: SoMeT, pp. 244-254. IOS Press, (Year)
16. Xudong, L., Jiancheng, W.: User interface design model. Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 3, pp. 538-543. IEEE, Qingdao, China (2007)
17. Kleppe, A.G., Warmer, J.B., Bast, W.: MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional (2003)
18. Yokoo, M.: Distributed constraint satisfaction: foundations of cooperation in multi-agent systems. Springer Publishing Company, Incorporated (2012)
19. Santiago, I., Jiménez, Á., Vara, J.M., De Castro, V., Bollati, V.A., Marcos, E.: Model-Driven Engineering as a new landscape for traceability management: A

systematic literature review. Information and Software Technology 54, 1340-1356 (2012)
20. Falsken, E.: Enabling the Mobile Enterprise with db4o Versant (2013)
21. Roopak, K.E., Rao, K.S.S., Ritesh, S., Chickerur, S.: Performance Comparison of Relational Database with Object Database (DB4o). In: Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on, pp. 512-515. (Year)
22. Esaki, K., Azuma, M., Komiyama, T.: Introduction of Quality Requirement and Evaluation Based on ISO/IEC SQuaRE Series of Standard. Trustworthy Computing and Services, pp. 94-101. Springer (2013)