Kamil Zyla[1]

ECONOMIC ASPECTS OF USER-ORIENTED MODELING FOR MOBILE DEVICES

Nowadays, when developing an application compatible with the majority of operating systems and devices is hardly possible, it is important to obtain the greatest possible return on investment in its creation. The main goal of MDE is to improve short-term (by increasing functionality of basic software particles) and long-term productivity (by reducing the rate of obsolescence of these particles). This article introduces a method of modeling mobile applications giving the mentioned benefits.

Keywords: *MDE, mobile technologies, software engineering, Aergia modeling language.*

Каміль Жила

ЕКОНОМІЧНІ АСПЕКТИ ОРІЄНТОВАНОГО НА КОРИСТУВАЧА МОДЕЛЮВАННЯ МОБІЛЬНИХ ПРИСТРОЇВ

У статті показано, що на даний час розробка додатків, сумісних із більшістю операційних систем і пристроїв, навряд чи можлива. Важлива максимально можлива віддача від інвестицій у їх створення. Основна мета модульного конструювання - підвищення короткострокової (за рахунок збільшення функціональності основних складових програмного забезпечення) і довгострокової продуктивності (за рахунок зниження швидкості застарівання). Представлено метод моделювання мобільних додатків виходячи зі вказаних переваг.

Ключові слова: *MDE, мобільні технології, розробка програмного забезпечення, мова моделювання Aergia.*

Камиль Жила

ЭКОНОМИЧЕСКИЕ АСПЕКТЫ ОРИЕНТИРОВАННОГО НА ПОЛЬЗОВАТЕЛЯ МОДЕЛИРОВАНИЯ МОБИЛЬНЫХ УСТРОЙСТВ

В статье показано, что в настоящее время разработка приложений, совместимых с большинством операционных систем и устройств, вряд ли возможна. Важна максимально возможная отдача от инвестиций в их создание. Основная цель модульного конструирования — повышение краткосрочной (за счет увеличения функциональности основных составляющих программного обеспечения) и долгосрочной производительности (за счет снижения скорости устаревания). Представлен метод моделирования мобильных приложений исходя из указанных преимуществ.

Ключевые слова: *MDE, мобильные технологии, разработка программного обеспечения, язык моделирования Aergia.*

1. Introduction. Nowadays application development focuses on mobile platforms and the Internet, which has become an easily accessible, commonly used tool and source of information. Also mobile devices (e.g., tablets and smartphones) are very commonly used not only as communication tool. According to the studies conducted by PBS in 2010 for the Office of Electronic Communications, 84.9% of the Poles declare possessing a mobile phone [1] and 50% declared possessing an Internet link

[1] Institute of Computer Science, Lublin University of Technology, Poland.

[2]. Furthermore, according to the GfKPolonia the share of smartphones (devices with open operating system and equipped with a touchscreen and/or QWERTY keyboard) in the overall mobile phones market in 2011 was 27.1% [3]. These factors influence significantly the need for mobile applications, which can be proven by the fact of increasing (basing on the statistics published in "Google Play" distribution system) the number of applications dedicated for Android platform of more than 230 thousands since September 2011 (Fig. 1) [4]. The strength of these trends encourages to search for the ways of applying MDE concepts in the mobile devices domain.

2. Mobile applications overview. Generally mobile applications can be divided into 4 main groups: native applications, server-side applications, hybrid applications and cross-compiled applications [5].

Native applications execute their logic and user interface on a particular mobile device. Programmer has possibility to i.e. build dedicated user interface, access sensors directly and exploit all platform-specific features. A set of good tutorials and tools that support well debugging, testing etc. is also available. The main disadvantage of such kind of applications is the necessity to write and optimize the same application for each mobile platform. This applications can be also distributed/sold using special stores like Android Market.

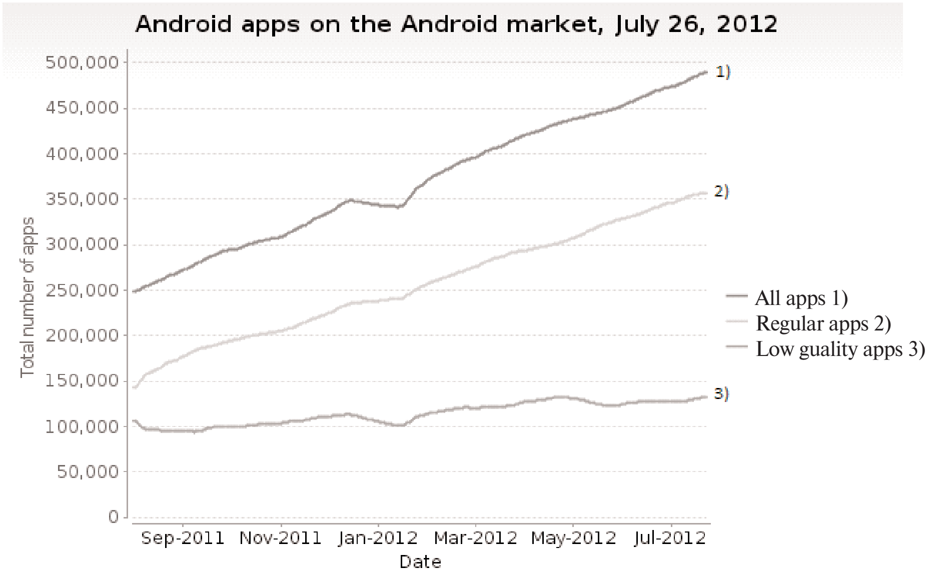
Server-side applications execute their logic and user interface on a special web server or web servers located somewhere in a network. Client application (e.g., mobile web browser) just renders the products of this execution. The main advantage is accessibility by a set of various mobile devices based on different platforms. The main disadvantages are lack of ability to distribute applications via markets and limited access to hardware and programming interface capabilities.

Hybrid applications are mixture of native and server-side applications — some of their logic and user interface is executed on a web server and locally on a mobile device. Their core logic can be programmed just once and run on a server, although significant effort is needed to build platform-specific clients, which can be sold via markets.

Variety of target mobile platforms (Fig. 2) and presented application types causes the demand for searching tools and methods that allows lowering costs and shortening time of developing software regardless of its type. This kind of requirements is met by cross-compiled applications. The main idea is to build a platform-independent model (textual or graphical) of application and transform it to the fully working platform-dependent solution. This process can be seen as a part of the model-driven engineering concept (Fig. 3).

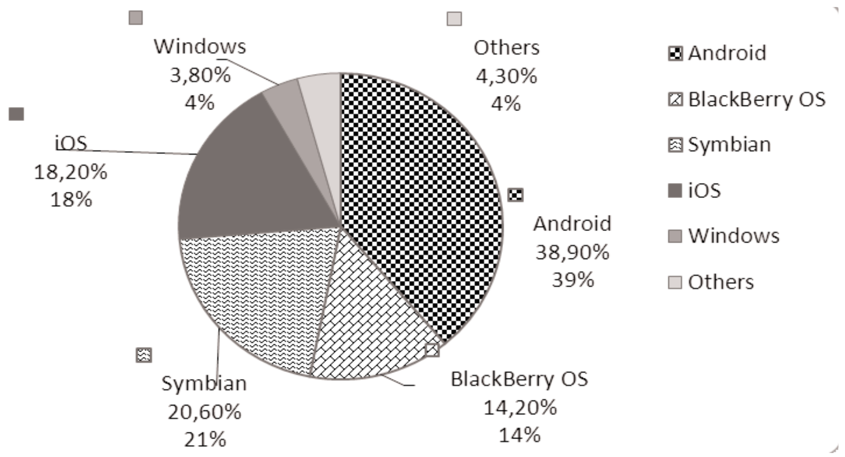
3. Basic notions of MDE. Model-driven engineering (MDE) is a part of software engineering concept aimed at increasing abstraction layer in a process of creating software by introducing different models on different levels and improving automation of this process by introducing model transformations ending on the executable code [6]. MDE is not well standardized and many approaches can be identified within it. The main are model-driven architecture — a formalized approach proposed by the object management group, and model-driven software development — less formalized and aimed at fast implementation [7]. Introducing MDE concepts in different ways depending on the approach is not necessary and misses the main goal of this paper, thus they will be introduced in the most general way possible.

MDE can improve short-term productivity by increasing the functionality of basic software particles and long-term productivity by reducing the rate of obsolescence of these particles. Generation of particular software particles is based on appropriate models and improves productivity of developers. Due to this fact, developer should influence the final application rather by changing its model than its generated code. Ensuring the consistency between the model and modernized software/application is a quite important issue, because the longer each software particle remains reusable, the greater is return on investment in its creation. [8]



Source: appbrain.com

Figure 1. **The number of available applications at the Android market**



Source: idc.com

Figure 2. **Market share of significant mobile platforms, 2011**

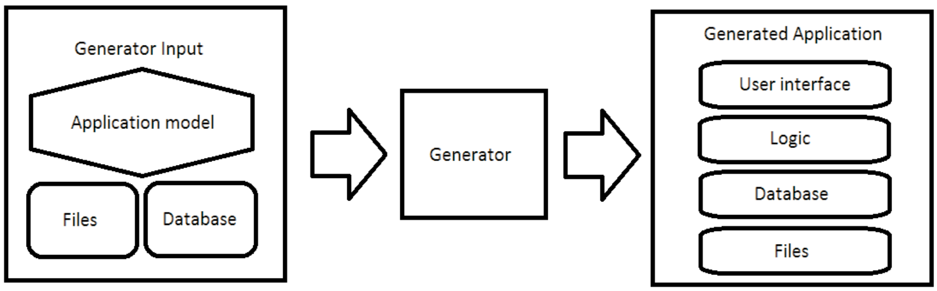


Figure 3. **Idea of cross-compiled applications [5]**

What all MDE approaches have in common is that software has to be described (modeled) using special notation or modeling language (either textual or graphical). These languages are inherently suited to solve problems and represent ideas in a way typical for a particular domain (e.g., mobile applications), so they are called domain-specific languages (DSLS) in opposite to general purpose languages (GPLs) [7]. Domain can be defined as "a bounded area of knowledge or interest" [9].

Properly designed DSL incorporates abstract syntax, one or few "work" syntaxes, description of mapping between abstract and real syntax, and description of semantics (the meaning of particular symbols and their function). Abstract syntax is usually defined by metamodel, which can be also used to describe semantics. Model created using DSL is called domain-specific model (DSM). Applications can be modeled using different DSMs that are translated into one application during code generation phase [7].

Description of method of creating models should be unified and formalized. Such "model of a model", called the metamodel, is a set of concepts (elements, processes etc.) related to a particular domain. If the model is an abstraction of a real-world phenomena, the meta-model is an abstraction showing the properties of such models and defining i.e. basic elements which can occur in the model [10].

4. AML short characteristics. The growing need for new mobile applications causes a demand for cheap, proven and easy to learn methods of rapid app development and for the professionals knowing how to implement them. It also causes a demand for appropriate designing tools (editors) and languages dedicated for modeling specific domains (solving specific problems).

Existing tools are not sufficient to model mobile domain in easy and expressive way taking also into account interactions between user and application and computation/data flows within it. Textual solutions (e.g., mobl, Applause) do not cover whole domain and might be difficult to learn and percept. Moreover, this way of creating model of the same functionality is usually more time consuming and complicated than while using graphical solutions. At last graphical tools are either too general (like Interaction flow modeling language) or designed for other domain (like Web modeling language) or provide only simple abstraction for programming language statements (like Google App Inventor).

Thus, the author of this paper developed Aergia modeling language (AML) — a new domain-specific graphical language for modeling mobile applications. The components introduced by AML cover the variety of domain-specific areas, including:

database operations, geolocation services, social services, multimedia services, GUI, content computation, device sensors, communication services etc.

Figure 4 presents a model of a mobile application screen for sending messages. The numbers of recipients are provided by the Contact picker component. The content of the message is provided by the Message form component. Both components are rendered as parts of GUI. Data gathered from the user is passed by the links to the Send message component which sends the message. Figure 5 presents very small part of AML metamodel, which is developed using the tools provided by the Eclipse modeling project.

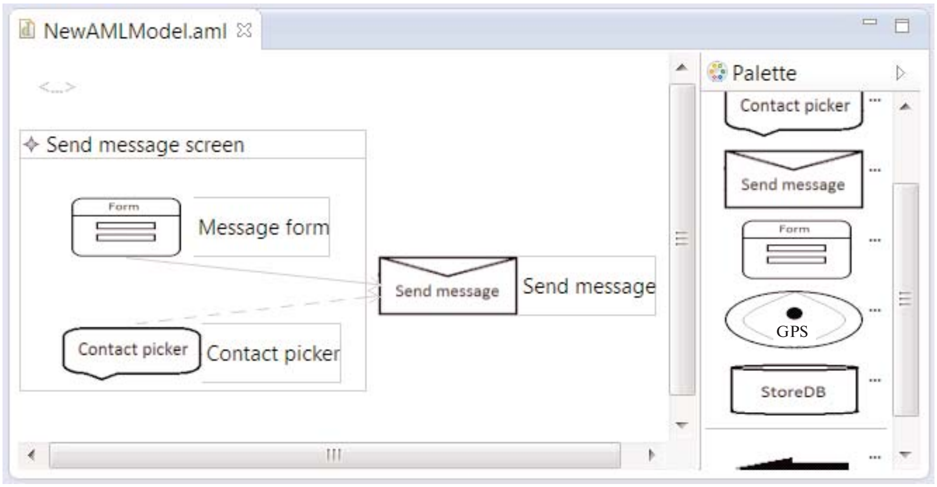


Figure 4. **Example of model made using AML**

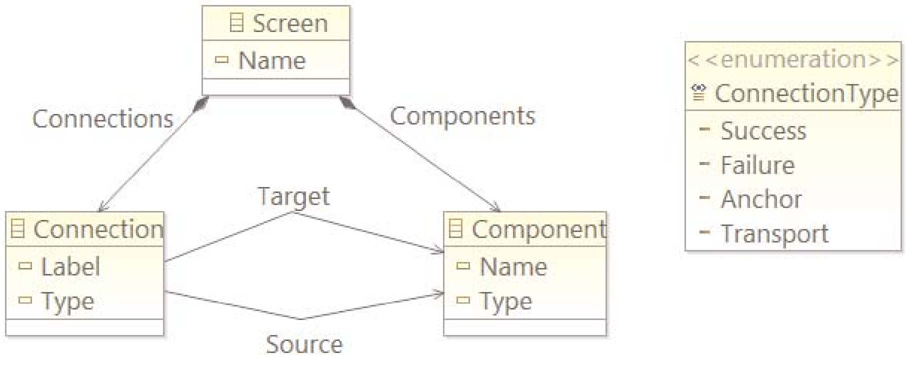


Figure 5. **Fragment of the AML metamodel**

5. Advantages and disadvantages of MDE concepts usage. Despite considerable resistance in the IT environment, associated with a complete change of the usual ways of developing software, MDE technologies are in the scope of the growing group of institutions responsible for creating lines in construction and maintenance of software. Nevertheless, all MDE solutions (including Aergia modeling language), regardless the domain, share some similar advantages and disadvantages.

Core benefits of using MDE solutions in a process of software development are as follows [11]:

1. Speeding up software development — a single component corresponds to many lines of code (increased abstraction level), development process focuses on application functionality and not on the implementation details.

2. Good influence on quality of software — good programming patterns implemented in the generator are reproduced in all of generated applications (utilization of proven software particles), some technical issues might be corrected by modifying the generator (applications are corrected by generating their code once again), validation can be performed on a model level.

3. Decreasing costs of software development — shorter time of delivering fully functional solutions, lower costs of application modifications, lower costs of creating software for different platforms.

4. Possibility to involve domain experts in a process of software development — ability to utilize knowledge of domain experts without programming skills and to facilitate synchronization between business needs and IT solutions.

On the other hand, usage of MDE might be dangerous because of the issues like[12]:

1. Decreasing flexibility of programming — necessity to adhere to the rigid forms and limits caused by automation of software development and increased abstraction layer, loss (to some extent) of full control over the resulting application code.

2. Changing requirements for participants of development process — changing usual ways of developing software by reducing large number of programmers to small group of experts (responsible for DSL editor and generated code) and moving burden of creating complete business solutions to the so-called "business engineers", who need to gather business requirements and express them in a formal model (quite demanding task).

3. Risking usage of immature MDE tool (without needed functionality implemented) or mistakes made by inexperienced team, e.g., possibility of accepting client's requirements hard or even impossible to meet with particular MDE tool resulting in loosing most of MDE benefits.

6. Summary. Nowadays, when developing one application compatible with the most of operating systems and devices is hardly possible, it is important to obtain the greatest possible return on investment in its creation. Model-driven engineering concept gives unique opportunity to meet these requirements (at least in case of some standard tasks), additionally speeding up development process and involving in it non-tech people.

The main effort needed, in order to apply MDE to the mobile domain, is to develop platform-independent method of describing mobile applications (like AML) and to create generator translating application model into platform-dependent solution. Return from that investment is obtained due to the two core benefits of MDE-like solutions: one application model can be the basis for generating applications on many platforms and changes (depending on their type) can be introduced to applications by modifying generator or model and generating the code again.

References:

Maj, M. (2012). Telefonia komorkowa: wieksze nasycenie i roznorodnosc, http://di.com.pl/news/30118, 26.07.

Laskowski, M. (2011). Czynniki zwiekszajqce jakosc uzytkowq interfejsow aplikacji internetowych, Logistyka, nr 6

Hatalska, N. (2012). Penetracja smartfonow w Polsce — dane za 2011, http://hatalska.com /2012/02/13/penetracja-smartfonow-w-polsce-dane-za-2011/, 26.07.

AppBrain. (2012). http://www.appbrain.com/stats/number-of-android-apps, 30.07.

Friese, P. (2012). Cross-platform mobile development // https://speakerdesk.com/peterfriese.

den Haan, J. (2012). MDA, Model Driven Architecture, basic concepts, http://www.theenterprisearchitect.eu/archive/2008/01/16/mda_model_driven_architecture_, 30.07.

Kesik, J. Zyla, K. (2011). Wspolczesne Technologie Informatyczne - Technologie MDE w projektowaniu aplikacji internetowych, Wydawnictwo Politechniki Lubelskiej, Lublin.

Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering, Computer 2(39).

Volter, M. (2012). Model-Driven Software Development Tutorial, http://www.voelter.de/data/presentations/mdsd-tutorial/01_Introduction.pdf, 30.07.

Stahl, M. T. (2006). Volter, Model-Driven Software Development: Technology, Engineering, Management, Wiley.

den Haan, J. (2012). 15 reasons why you should start using Model Driven Development, http://www.theenterprisearchitect.eu/archive/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development, 30.07.

den Haan, J. (2012). 8 reasons why Model-Driven Development is dangerous, http://www.theenterprisearchitect.eu/archive/2009/06/25/8-reasons-why-model-driven-development-is-dangerous, 30.07.

Стаття надійшла до редакції 11.09.2012.