Ubiq Mobile + QReal – a technology for development of distributed mobile services

Timofey Bryksin, Yuri Litvinov, Valentin Onossovski, Andrey N. Terekhov

St.-Petersburg State University, Faculty of Mathematics and Mechanics
Department of Software Engineering
198504, Universitetsky av.,28, St.-Petersburg, Russia
timofey.bryksin@gmail.com, yurii.litvinov@gmail.com, v.onossovski@ubiqmobile.com, ant@tercom.ru

Abstract

The integrated technology for quick and efficient development of distributed mobile services is described in the paper. The technology includes two main components that are tightly integrated: Ubiq Mobile platform that provides development tools and execution environment for efficient and robust multi-platform mobile services, and QReal – a visual modeling technology that lets to define various domain-specific graphical languages (DSLs) and use them for rapid development of distributed mobile applications in different subject areas.

Index Terms: Distributes systems, Mobile services, Visual modeling, Domain-specific languages, Software development technologies.

I. Introduction

Among the myriad of mobile applications taking information from the Internet consisting of hundreds of thousands of programs and continually expanding, we are interested in those programs that are actually mobile components of larger distributed systems, in contrast with the others that can be characterized as “applications for mobile devices, taking information from the Internet from time to time”. Let’s call such systems distributed mobile services, if they are

- Designed for large number of mobile users including those who work simultaneously;
- Designed for working in continuous mode and have complex server-side business logic, including functionality that doesn’t depend on current users’ activities and can work without active users’ sessions;
- Potentially multiplatform, i.e., support various mobile platforms and types of mobile devices.

The examples of distributed mobile services are multi-user online games for mobile devices (like Tic-tac-toe or Sea Battle from one mobile phone to another), systems of mobile access to various automation systems (like video surveillance), mobile interfaces to distributed business applications etc. Such sorts of applications as, for instance, mobile versions of information or media websites or “find you nearest ATM” services obviously don’t belong to this category.

Most of modern popular technologies for development of mobile applications (like [1]. [2], [3]) are targeted on development of relatively simple applications and can barely help in creation of distributed mobile services. The most suitable technology for this class of applications is still mobile Ajax [4], but it has its own drawbacks and isn't quite appropriate for wide practical use in many cases.

We propose an original approach to creation of distributed mobile services based on two major components. The first component is Ubiq Mobile platform [5] that provides both programming tools and runtime support for creation of multi-platform distributed mobile services that are able to work "always and everywhere" on wide range of mobile devices and in different conditions of mobile connections, including slow and unstable ones. Another component is QReal visual modeling technology [6] that contains tools for creation of various graphical domain-specific languages (DSLs) and provides automatic code generation from these languages into Ubiq Mobile API. The use of domain-specific modeling (DSM) approach for mobile applications development, according to [7], dramatically increases programmers' productivity as well as overall efficiency of the development process. We believe that the combination of DSM technology (QReal) with robust and high-efficient runtime support platform (Ubiq Mobile) will provide a qualitative effect of simplifying and accelerating the development of distributed mobile services.

II. UBIQ MOBILE PLATFORM

Ubiq Mobile is an original platform for creation of modern distributed mobile services with rich functionality. The platform is targeted for a wide range of modern mobile devices – from simple Java phones and low-end smartphones to the advanced hi-end devices on iOS, Android and Windows Phone platforms.

An important feature of Ubiq Mobile platform is its ability to function normally in poorly developed mobile infrastructure and mobile networks with relatively low data transfer (GPRS, EDGE).

The Ubiq Mobile project has been carried out in St.Petersburg State University by the team of specialists and students for the last 3 years. The strategic goal of the project is to give a broad range of conventional programmers an ability to create modern distributed mobile services that can work "always and everywhere" where there is any kind of mobile connection.

Ubiq Mobile platform is focused on the following major types of distributed mobile services:

- Interactive information services with complex business logic inside (smart boards with constantly changed information, displaying information on maps using Google Maps or similar services, support of push mode initiated by server side etc.);
- Mobile interfaces for distributed corporate applications and information systems;
- Tourist information services, information support of events and sports;
- Mobile banking applications;

- Remote interaction with "intelligent" automation systems like video surveillance, smart house automation etc.;
- Multi-user online games;
- Web 2.0 services, taking information from social networks.

The platform is targeted on creation of mobile distributed services both for business and for individual programmers and small teams who develop a variety of services for wide range of public users. No special experience in developing mobile applications is required for development of services on Ubiq Mobile platform.

The platform in its basic configuration implements terminal architecture – the majority of applications' business logic is running on server side, while mobile devices act as intelligent graphical terminals and interact with the server side via original binary protocol built over TCP/IP. The use of proprietary protocol instead of one of existing open protocols, on the one hand, increases closeness of the platform. On the other hand, it gives many additional opportunities for optimization of graphical data transfer between server and mobile devices and lets the platform to correctly process mobile connections breaks and make applications more robust against network instability. For enhanced mobile devices with rich functionality (like iPhone, Android-based and Windows Phone-based smartphones), the Ubiq Mobile platform provides an option of integration of server-side application components with specially developed mobile applications.

The platform provides significant savings of mobile traffic for its services. Typically the user of Ubiq Mobile distributed service consumes less traffic in comparison with a process of web surfing through a mobile phone browser during the same period of time. Low traffic consumption, on the one hand, facilitates the work of services on slow connections and, on the other hand, reduces overall cost of using Ubiq Mobile services.

Ubiq Mobile services can function correctly in any kind of mobile networks, including ones with slow and unstable mobile connections. Low traffic consumption allows applications to function normally even on slow GPRS connections. The platform includes special tools for adaptive tuning of parameters of data exchange between server and mobile client depending on connection speed. In particular, compression quality of transferred fragments of graphic images varies dynamically depending on connection speed.

Another subsystem provides correct work of applications in conditions of frequent connection breaks and reconnections, for example, in a moving subway train. When the connection breaks, the platform sends a signal to the application, automatically suspends UI process and saves its state. After that, during some time the mobile client tries to restore connection and, if successfully reconnects (like in subway train), the UI process is being automatically resumed from the saved state.

A distributed service is represented in Ubiq Mobile as a set of concurrent processes that are interacting through mailboxes. Some of them correspond to active mobile users' sessions while the others implement "purely server-side" logic and permanently exist in the system during the whole lifetime of the service. Depending on service architecture, the users' processes can run either on server or on mobile devices. The permanent processes are running only on server side.

There are several models of implementation of mobile users' interfaces in Ubiq Mobile platform:

- Basic model (**B**) – an UI process is running on a server with a mobile device as a remote graphical terminal working with this server through unified thin client program
- Basic Native model (**NB**) – a UI process is running on a server, mobile devices acts as graphical terminal but, in contrast with **B** option, user interface is composed from native controls of this concrete device and therefore has native-like look and feel.
- Native model (**N**) – one or more user processes (including UI process) are implemented in client-side program, running on mobile device and specially developed for this particular service. This program acts as thick client and interacts with other processes of this service through special API provided by Ubiq Mobile for specific mobile platform.

The purpose of **B** model is to provide appropriate quality of graphics and level of functionality for the majority of supported mobile platforms, including the weakest ones (like Java phones and low-end smartphones). On this level, most of rendering work is performed on the server and fragments of rendered images are transferring to the mobile client in compressed mode.

For advanced mobile platforms with rich functionality (like iOS, Android and Windows Phone) one can use **NB** model. In this case UI screen is represented as a tree consisting of generalized controls. Instead of rendering the tree into final image on server side (like in B model), fragments of the tree are transferred to the mobile client. On client side, generalized controls are mapped into appropriate native controls and the whole UI screen is being rendered on the mobile device.

The **N** model could be used if the authors of mobile service want to provide fully native thick client for particular mobile platform. Unlike terminal clients for **B** and **NB** models, such a thick client won't be universal and will serve only specific mobile service (or services) on which it is targeted. The use of **N** model makes sense either for really mass mobile services or for providing smooth interfaces between existing native mobile applications and server-side logic on Ubiq Mobile server.

Ubiq Mobile platform provides transparency of services with respect to the UI models used by different clients. It means that the same service can work at the same time with Java mobile phone using **B** model, and with iPhone or Android device – using either **NB** or **N** models. The transparency is provided on the level of developed applications – all differences are encapsulated inside Ubiq Mobile graphical subsystem that can use different subsets of protocol for interaction with mobile clients depending on their UI models.

The Ubiq Mobile server-side SDK is currently working on Microsoft.NET. The choice of .NET as environment for the platform has been quite arbitrary and was made mainly due to prevailing development skills in the team at the start of the project. Now there are plans of development Java version of server SDK to avoid use of third-party proprietary software.

The Ubiq Mobile platform includes the following components:

- Server-side SDK, that includes server components, API library and debugging tools. Development of distributed services is assumed to be performed in Visual Studio.
- A set of unified thin clients for all supported mobile platforms that provide **B** and **NB** models of user interface.
- A set of APIs for interaction between native thick clients with server processes. Such APIs are implemented for all mobile platforms where Ubiq Mobile provides **N** model of user interface.

Various services in Ubiq Mobile can be either hosted on one server or distributed among different servers. For the access of mobile clients to services, the platform supports general directory that is replicated among all servers registered in the system. So if the creators of the new service want to publish it for public access, they can host it either on their own server or somewhere in the cloud and then register the new server in Ubiq Mobile directory. Soon after that, the information about all services hosted on this server will be available for all Ubiq Mobile users.

Unified representation of Ubiq Mobile distributed applications as sets of processes together with thought-out API helps to make programmers' work to develop distributed mobile services significantly easier. However, much more productivity boost could be gained by use of graphical domain-specific languages (DSL). Using DSLs to create distributed mobile services allows to

- Considerably increase productivity of developers of mobile services;
- Use different DSL for services, oriented on different domains;
- Broaden and make usage of native thick clients for advanced moble platforms easier (by automated generation of user processes from DSLs into native code for selected mobile platform);
- Easily redistribute user processes between client and server transparently to other pars of the system;
- Simplify creation and support of distributed applications' template library.

To choose an appropriate tool to create DSL and supporting generators into Ubiq Mobile platform, a detailed analysis was performed and some popular technologies were analyzed (both mature industrial and developing projects, [8], [9]). As a result a decision was made to use QReal metaCASE system, which is being developed by a research group led by prof. A. Terekhov in St. Petersburg state university for several years now. This choice was made due to some of QReal's features described below and, most importantly, the possibility of fine tuning to meet the requirements of distributed mobile applications development.

III. QREAL TECHNOLOGY

QReal is a technology and supporting tools, implementing domain-specific modeling paradigm. Sometimes creation of a new language, oriented on the problem that is being solved, and solving it with this new language becomes cheaper and faster than solving it with general-purpose languages. Having adequate tool support, the process of creation of visual languages and appropriate tools like code generators can become cheap enough to make this approach economically reasonable to apply not only in large projects.

QReal metaCASE system consists of tools for rapid development of visual languages, graphical editors for them, code generators, repository, version control, visual debugging and other tools developers are used to have. Besides, QReal offers an environment to work with graphical languages. All created visual editors are built as dynamical libraries, which gives them unified user interface and all QReal's features described above.

A technology like this could be applied in a wide range of projects starting from small companies that are developing a lot of similar applications (specific types of mobile applications, data-oriented information systems etc.) and up to large companies that want to have their own programming tools, specialized on particular project. In addition, visual DSLs often could be easily used even by non-professional programmers due to their expressiveness.

There are some already existing metaCASE tools on a market. Possibly the most successful of them is MetaEdit+[7], developed by MetaCase company and based on results of a research group of Jyvaskyla University. It allows to quickly create a DSL and use it with stand-alone visual editor. For example, in MetaEdit+ a visual language for programming of mobile phones was implemented and used by Nokia. MetaEdit+ is quite expensive, charging more than 9000 euro for one full license, but has quite old-looking user interface, while metamodelling part and engine of QReal is free and open-source, and there is ongoing research of new user interface features.

There are many "ad-hoc" visual programming tools for various applications, for example, for mobile phones ("ad-hoc" in a sense that their visual editors are implemented by hand for specific tasks). They can be quite useful, but lack main advantage of metaCASE systems – ability to quickly change existing languages and create new ones just when the need arises. Also, many of such tools (like iBuildApp and similar services [10, 11, 12]) are not intended to be used for solving complex tasks like non-trivial client-server communication.

QReal applies the domain-specific modeling principle to itself — one of its build-in editors is meta-editor, an editor that allows to describe visual languages by specifying their meta-models. There is also shape editor that helps to describe concrete syntax of created language elements. Besides, some research is going on to create tools for rapid automated development of source code generators.

For behavioral diagrams in QReal there is a possibility to organize visual interpretation of created models. Diagrams are interpreted step by step allowing developer to follow the control flow of the diagram. Besides, for some languages there is also support for debugging of generated source code. In this case the correspondence between elements on diagram and code lines is being made, which makes possible to highlight current diagram elements as debugging is being run, show watch-lists of used variables an so on.

For quick creation of diagram elements and associations between them mouse gestures recognition mechanism is used. A mouse gesture is associated with every element of a graphical language, and when user draws this gesture with pressed right mouse button, QReal recognizes it and creates corresponding element in a workspace of an editor. To create link between elements one shall simply draw a line starting from one element and ending on another with right mouse button pressed.

To support multiple users working simultaneously over the models, QReal uses version control systems as a storage for created diagrams. As yet, QReal supports only Subversion version control system and implements its most commonly used commands with proper support for specifics of storing diagrams. For example, it can graphically show the results of "diff" operation as two diagrams with changes marked on them. Added, deleted or changed elements are highlighted with different colors.

Features of QReal technology described earlier allow creation of not only graphical diagram editors, but full development environment with a set of tools that is expected from common IDEs: version control integration, debugging and so on.

As one case of application of QReal technology, a language for client-server applications was created. It describes applications consisting of a set of processes interacting with each other, where part of them is running on a server, and part — on a mobile client. A program in that language is similar to UML activity diagram and has operators of signal sending/receiving, device native API calls and calls to external web-services, flow control operators such as "if", and so on. An example of such program is shown on Fig. 1.

Most of Ubiq Mobile applications have similar simple structure based on finite automata, so it is quite convenient target for generation. In addition, there is a special API extension for Ubiq Mobile, developed for generator integration. All supporting code for initialization and finalization of processes, message passing and so on is wrapped in this API, so generated code does not need to implement it. Generated program is mainly a set of calls to this API and a control flow connecting them.

Domain-specific language allows to invoke API methods directly, writing their calls in special blocks on diagrams, like assembler insertions in traditional programming languages. This feature makes possible to leave generator unchanged when underlying Ubiq Mobile API changes frequently, and provides developers more fine-grained control over generated application.

Besides the generator we are planning to implement GUI forms editor and a generator of GUI for mobile phones using Ubiq Mobile. It will allow to create all components of the target application inside one tool, and being fully integrated with visual DSL, help developers to be much more productive.

Now we have implemented client-server application for transferring video frames from a webcam to a mobile phone, fully implemented in QReal, part of it shown on Fig.1. It has some non-trivial logic, like working with variables, lists of variables, asynchronous signal sending and receiving, etc. This example can be fully generated into C# code that will not require handmade changes. So it is a proof that non-trivial applications could be developed using visual technology, but complex data flow, data transformations and complex interaction with user require additional visual languages or even tools (like mentioned before GUI editor), and needs further research to make it feasible.

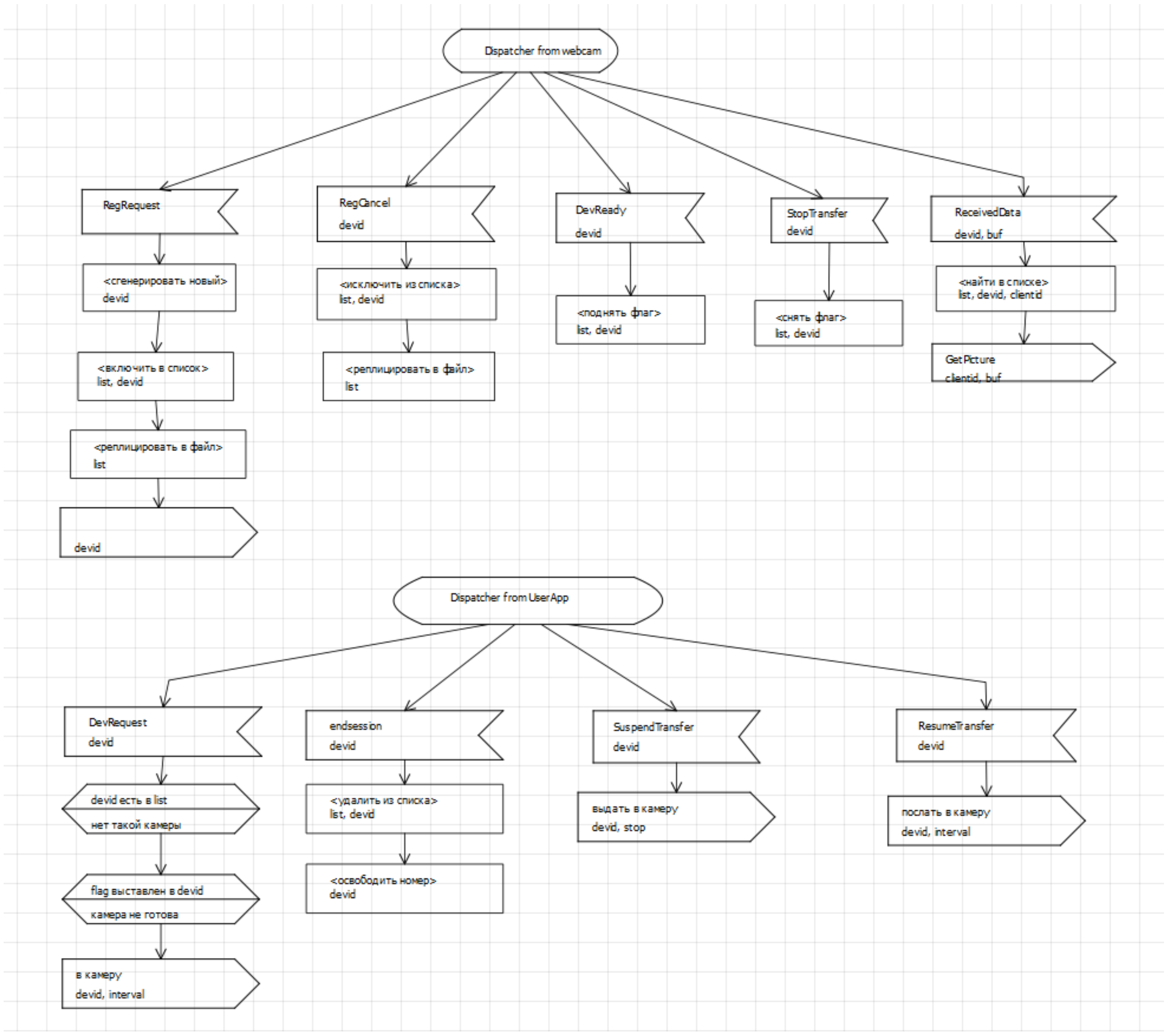

Fig. 1. Example of DSL program – server part of client-server application

IV. Conclusion

DSM approach, despite its obvious advantages [7], is not sufficiently spread in the traditional area of industrial software engineering. We believe that in the field of development of distributed mobile applications the situation will be dramatically different, first of all due to higher complexity of mobile development in comparison with “traditional” software and thus, higher level of skills requirements for developers. In addition, mobile distributed services and applications are more specialized and distributed between domains. The use of DSM approach for distributed mobile services development significantly reduces the level of complexity of development and makes programmers' work easier. On the other hand, natural domain specialization of mobile services makes easier creation of domain-specific languages (DSLs) focused on various classes of applications. In the proposed approach, Ubiq Mobile platform provides robust and high-efficient runtime environment for mobile services while QReal meta-technology provides abilities to create specialized DSLs for various types of mobile services and automatically generate object code for Ubiq Mobile platform from these DSLs.

References

[1] http://ibuildapp.com/
[2] http://www.phonegap.com/
[3] http://www.stackmob.com/
[4] Brian Fling. "Mobile Design and Development: Practical concepts and techniques for creation mobile sites and web apps", O'Reilly Media, 2009
[5] V. Onossovski, A.Terekhov. Ubiq Mobile – a New Universal Platform for Mobile Online Services, Proceedings of 6th seminar of FRUCT Program, 2009
[6] A. Terekhov, T. Bryksin, Y. Litvinov et al., Architecture of QReal visual modelling tool (in Russian) // Software engineering, St. Petersburg university press, 2009, p. 171-196
[7] S. Kelly, J.-P. Tolvanen, Domain-Specific Modeling: Enabling Full Code Generation // Wiley-IEEE Computer Society Press. 2008. 448 pp.
[8] http://www.eclipse.org/modeling/gmp/
[9] http://www.metacase.com/
[10] http://gotiggr.com/
[11] http://ibuildapp.com/
[12] http://www.appmakr.com/