Mobile Multi-Service Smart Room Client: Initial Study for Multi-Platform Development

Andrey Vdovenko*, Sergey Marchenkov*, Dmitry Korzun*†
*Petrozavodsk State University (PetrSU), Russia
†Helsinki Institute for Information Technology (HIIT), Finland
{vdovenko, marchenk, dkorzun}@cs.karelia.ru

Abstract

Smart Room provides a service set to automate such research and educational activity as conferences, lectures, and meetings. For human participants the services are accessible via mobile personal end-user devices. A mobile device hosts a smart room client, which registers the participant in the smart room space, shares her/his personal data and context, and accesses available services in the room. This paper reports recent progress of the smart room client development. Our key objectives are (1) development unification for various mobile platforms, where Windows Phone and Symbian are our reference cases; (2) support in the user interface of personalized access to the important services from a large set. Our initial study shows that objective (1) is an architecture design issue and our multi-platform development is based on tradeoffs between native mechanisms and portable solutions. We achieve objective (2) using HTML5 features, supported by many modern mobile platforms.

Index Terms: Smart room, Mobile programming, User interface, Multi-service, Multi-platform development.

I. Introduction

Smart Room (SR) is a system that provides a service set to automate research and educational activity such as conferences, lectures, and meetings [1], [2]. Cooperation of various computational devices aims at intelligent assistance for events held in the room. The development follows the smart spaces concept [3], implemented with Smart-M3 platform [4] and SmartSlog SDK [5]. The base SR scenario targets automated construction of the agenda with further dynamic adaptation to context, participant interests, and online activity. Collective generation of new knowledge is also possible and uses information from multiple local sources (e.g., sensed parameters of physical environment, discussions on talks, user presence statistics) and external services (e.g., Google Scholar index, media flows from remote experts).

A user interacts in the smart room through personal device [6]. Being in the smart room and connected to the wireless local area network (WLAN), the participant runs a client (SR-client) on her/his device. The client becomes aware of currently available services and allows the user to select and access a narrow subset of desired services from the wide set that the smart room system provides for all participants.

This paper introduces architecture and design of the SR-client. We made an initial study that focuses on (1) peculiar properties and requirements of multi-platform development and (2) design and implementation of the multi-service property in graphical user interface (UI) of the SR-client. These two issues are non-trivial from the point of view of mobile programming due to known specific restrictions of mobile devices. The project is open source, its prototype releases are published at http://sourceforge.net/projects/smartroom/.

A variety of suitable mobile platforms exists to host SR clients. In this paper, we consider Windows Phone and Symbian as reference cases. For them our development has already

resulted in workable prototypes, identifying tradeoffs between native mechanisms and portable solutions that have to be analyzed in SR-client development for a mobile platform. For other popular platforms, such as Android and iOS, the development of SR-clients has also been started, but we do not present it in this paper due to its early stage. We plan to transfer common design solutions (e.g., high-level architecture, GUI concept, tests scenarios), which we initially elaborated for the two selected reference cases.

The multi-service property deals with SR-client usability and user interface. We present GUI design, which easily allows a user to select and navigate most appropriate smart room services. It uses dynamic construction of GUI elements such that the user interacts with a personalized subset of services. HTML5 is the base technology that allows this construction and thus appropriate for implementing the required access features in mobile GUI. The HTML5 technology is supported on many modern mobile platforms. In this initial study we validate our GUI design solution on two prototype implementations—for Windows Phone and Symbian.

Note that issues of information interoperability and semantic processing in the smart space are out of the scope of this paper. In particular, we do not consider algorithms of personalized decision making; we focus on the client design support within which such algorithms can implemented. An interested reader can follow, e.g., [5] for multi-platform development of Smart-M3 knowledge processors and [7], [8] for development of personalized and proactive services in Smart-M3.

The rest of the paper is organized as follows. Section II introduces the Smart Room system and the role of mobile SR-clients. Section III presents the high-level architecture and design rules that any SR-client can follow, despite of the chosen mobile platform. Then we consider platform-aware details of SR-client implementation based on the two reference cases of mobile platform. Section IV concludes the paper.

II. Software components of Smart Room

The main software components of Smart Room are services and clients [2], [9] as shown in Fig. 1. The services produce information from shared in the smart room space. Information important for all participants is displayed on two (or more) public screens. The clients allows accessing available information on a personal basis for local use by human participants.

A. Services

Smart Room is based on the service set principle implying that a set of services (which can dynamically change their state between active and inactive) is available at the moment and offered to participants. Every service is a smart space agent and can be accessed from SR-client installed on the end-user mobile device. Although the services are potentially available for all participants, the important requirement is that a service subset should be offered for a participant based on her/his preferences and current context [2].

The Smart Room service set consists of several groups (Fig. 1). Conference services are agenda and projector services, inherited from Smart Conference system [10], [11]. Agenda service handles the runtime of conference and according to the schedule starts a speech of particular participant. The conference agenda and related information are displayed on a public screen. Agenda service interacts with projector service (through the shared smart room space), which shows the slides of recent speaker on the second public screen of the room (pure screen or smartboard). Projector service starts a slide show (first slide of the presentation) and then

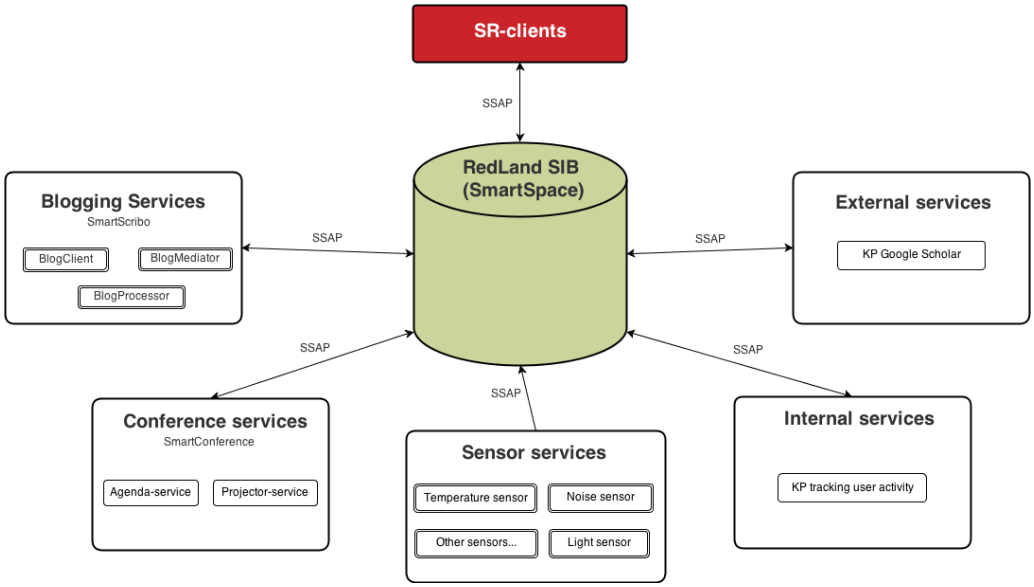

Fig. 1. Vision of Smart Room as a service set. SR-client is a user access point to the services

waits for commands from the speaker. Using her/his client on the mobile device or the sensor smartboard the speaker informs projector about slides switching or presentation ending.

Conference is supplemented with blogging services that support discussions during the talks [10]. The discussion message tree is stored in the smart room space as a cache and synchronized with the external host service such as LiveJournal.

Internal services form or analyze the smart room environment and provide related information to the participants. The services process and represent information from sensors embedded in the room (temperature, lightning, etc.). Another example is services that analyze activity of each participant, e.g., based on her/his personal talks and questions metrics or contribution rate to the discussion with help of cameras and mikes. All content produced in the room is summarized and accumulated by internal services in the content collection.

External services provide additional information to the smart room by the means of other external entities (web sites, data bases, etc.). An example is a service that queries Google Scholar for the citation index of speakers.

The visualization on public screens is dynamic and selective. For instance, the information about current speaker is highlighted on the agenda screen and information on climatic parameters of the room is periodically interchanged with recent comments from the participants discussion.

B. Clients

SR-client is a personal access point of the user to operate in Smart Room. Through the client the user interacts with available services, browsing available information, sharing own information, and injecting control actions.

The base services the client operates with are conference services: Projector and Agenda. Any participant can browse the information from Agenda and Projector locally on her/his client, without need to watch to the public screen. It is especially important when the screen shows information partially, because of the screen size limits and public visualization

preferences. Projector is also under control of the current speaker, who switches the slides during her/his talk.

An essential requirement for the client is support of the unfixed number of services, which can be available simultaneously in the room. That is, a multi-service client has to be able to support new appeared services without re-writing the code. From the GUI point of view the user just navigates from one service to another. For example, such navigation includes the following services: the participant browses the conference agenda, looks through presentations of other participants, edits her/his personal profile and updates own slides, watches the recent temperature and lighting level in the room, participates in blog-based discussion of the current talk, studies locally the recent slide of the ongoing talk.

One of the major features of SR-client is the superuser mode to perform chairman functions. A client switches to this mode by request with username and password. The chairman has additional opportunities for service administration, in particular, she/he controls Agenda and publicly displayed information on the agenda screen. For instance, the chairman can change the agenda manually, terminate the presentation, or switch the information pieces visualized on the agenda public screen. The idea is depicted in Fig. 2.

III. Implementation for mobile platforms

We analyzed available technologies that allow constructing a multi-service SR-client. Our attention is to possible common components used for the multi-platform development.

A. Multi-platform considerations

SR-client consists of a GUI, smart space access, and application logic. The latter locally processes available data from the space and produces information to the user and to share in the space. By selecting certain design patterns for each platform, we can implement client where native mechanisms are mostly used for GUI implementation. Portable smart spaces access can be implemented using SmartSlog SDK [5]. The client application logic can be written on a general-purpose programming language, making the code portable for a wide class of platforms.

We consider the following mobile platforms suitable for implementing SR-client: Windows Phone, Symbian, Android, iOS, and mobile Linux family. The first two are selected as reference platforms for our initial study, which aims at analyzing tradeoffs between native and portable mechanisms and identifying feasible solutions common for several platforms. Windows Phone [12] is characterized with very distinctive features compared with other platforms, so being a good candidate for identification of really common solutions. Symbian supports widespread cross-platform framework Qt, so many developed solution can be easily transferred later to some other platforms.

Both reference platform support object-oriented (OO) design based on C# for Windows Phone and C++ for Symbian. Therefore, we develop a high-level class hierarchy of SR-client. This OO model is clearly a common solution since many other mobile platforms support OO programming (see Table I). Technical details of the results of OO modeling for SR-client can be found in the Smart Room wiki (http://oss.fruct.org/wiki/SmartRoom).

An important problem we consider in this study is the possibility to create a partially unified GUI design of SR-client such that the common part is much as possible and implementation of platform-aware components takes little effort. The common part may include components that are not universally portable but cover a set of mobile platform. Table I summarizes programming languages and development environments that popular mobile platforms provide.

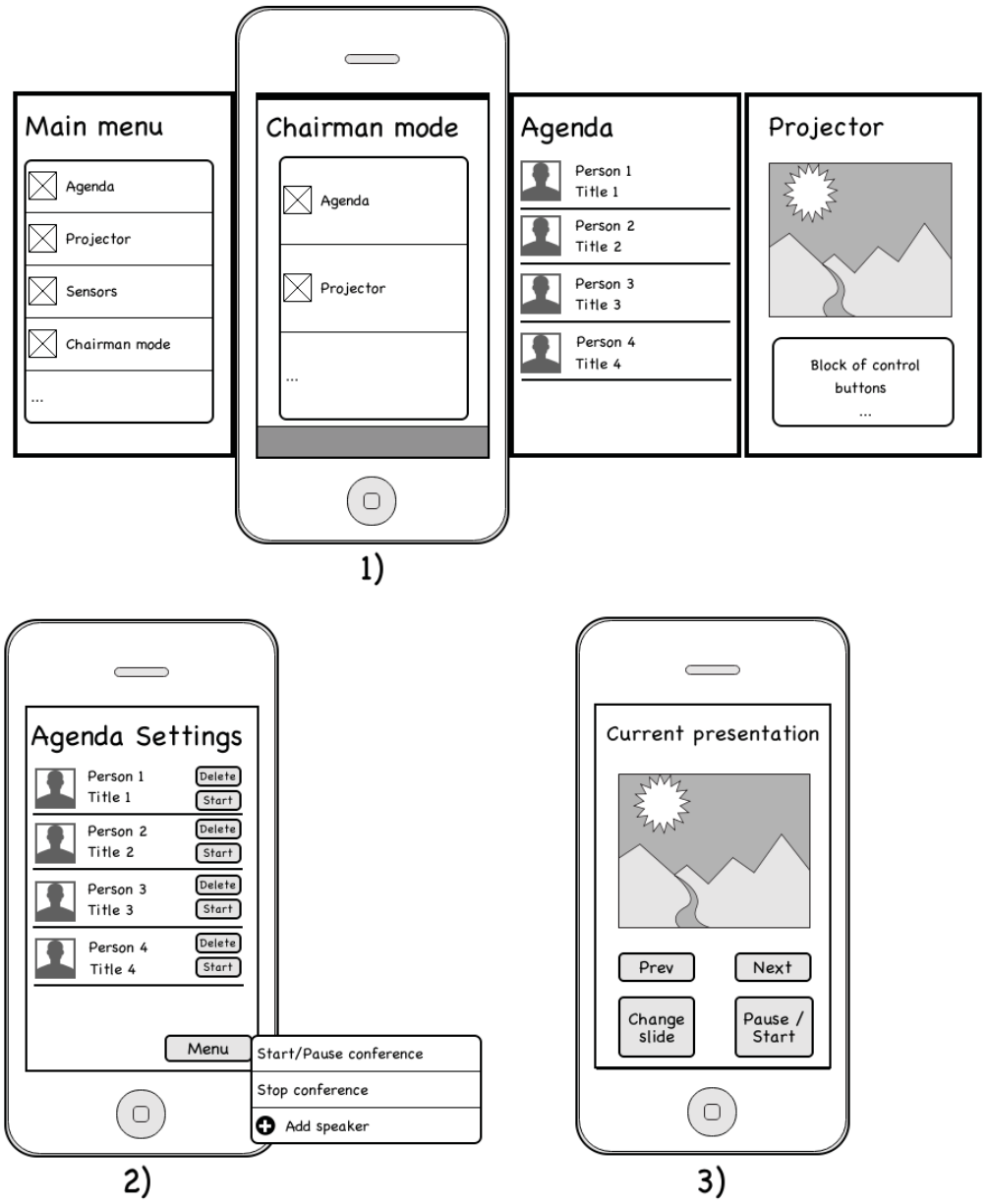

Fig. 2. Schematic view on the client GUI for chairman: 1) entering the superuser mode, 2) changing the order of talks in Agenda, 3) controlling a talk

Basically, it makes impression which kind of native mechanisms the SR-client development has to consider in GUI implementation.

The resultant tradeoff is that implementation of the smart space access module and application logic can be made portable to wide extent. Portability of multi-service GUI can be achieved (partially) by using the HTML5 technology. Nevertheless, many GUI elements have to be implemented using native platform mechanisms.

B. User interface

The GUI structure and some interface elements are common for the all mobile platforms. The GUI design aims at letting the user comfortably take advantage of many services available

TABLE I
MOBILE PLATFORMS: BASE DEVELOPMENT TOOLS

Platform	Programming language	Integrated development environment
Android	Java, partly C, C++	Eclipse
BlackBerry	Java	Eclipse
iOS	Objective-C	Xcode
Symbian	C++, QML	Qt Creator
Maemo, MeeGo	C++, QML	Qt Creator
Windows Phone	C#, Visual Basic	Visual Studio 2010

in the room. The user may select a personalized (and small) subset of them (active services). That is, the user should be able to control the access to services and to navigate them.

Service navigation and control can be based on menu or use an advanced screen with tabs and special controls. Both methods require plug-ins, which become available for each active service at an appropriate point. The user starts a dedicated page on which (s)he clicks for each active plug-in. After that the user has ability to quickly navigate the service features.

The menu-based method is inappropriate when a large number of plug-ins appears. Placing them into a single menu makes the GUI cumbersome, especially for mobile devices with small displays (e.g., phones). In this case, the second method is better; Fig. 3 shows an example how an advanced screen can look in the SR-client. The method is implemented in our prototypes of SR-client for smartphones on Windows Phone and Symbian.

On tablets, the layout of available plug-ins can use the same idea as for mobile phones. Since a tablet has a bigger display, the user can simultaneously track many plug-ins on the same window and use multiple tabs for the content. The layout can be implemented with window managers (e.g., Xmonad for Linux and AquaSnap for Windows 7). Another option is specialization for each platform. For instance, in Windows Phone we can use so called docks, which are similar to the working space layout mechanism of Visual Studio.

As an advanced cross-platform development tool, it is reasonable to employ the Qt framework. It allows creating portable client code for certain mobile platforms (namely Symbian, Harmattan, Windows on tablets). Unfortunately, many other platforms (e.g., Windows Phone,

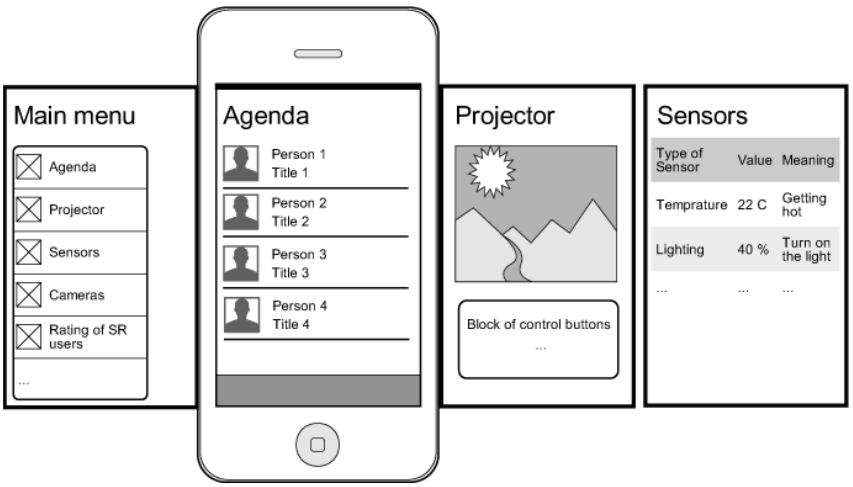

Fig. 3. Advanced screen layout of SR-client: each tab represents a service

Windows 8, Android, and iOS) still have no proper Qt support. (Some work in this direction, however, is in progress.) For them specialized GUI development methods should be used.

Personalized service selection is a challenging problem. It requires flexible and dynamical construction of appropriate multi-service interface elements in dependence on the context and user interests. For this problem we consider the following two solutions.

The first one is a single application written using HTML5 libraries (e.g., PhoneGap, Titanium, Rhodes). This solution is multi-platform since the majority of modern mobile platforms provide HTML5 support. Unfortunately, mobile devices of the previous generation likely meet performance problems and limited functionality.

The second solution is to represent each service as a separate HTML5 plus CSS3 page using JavaScript for internal logic of the device page. The SR-client collects information from services and constructs the associated service pages. The most important services are represented with static pages, where a service-specific layout of control elements is used. Some services can be constructed dynamically, based on a set of smaller services, e.g., representing on one page many parameters from the Smart Room sensors. This solution is in our recent prototypes of SR-client.

C. Windows Phone

Our implementation of SR-client for Windows Phone uses MVVM design pattern [12]. The Model-View-ViewModel (MVVM) is an architectural pattern that originated from Microsoft as a specialization of the presentation model design pattern. Largely based on the modelviewcontroller pattern (MVC), MVVM is targeted at modern GUI development platforms that support Event-driven programming, such as HTML5, Windows Presentation Foundation (WPF), Silverlight, and the ZK framework. Elements of the MVVM pattern include Model, View, ViewModel.

The element Model, as in the classic MVC pattern, refers to either (a) a domain model that represents the real state content (an object-oriented approach), or (b) the data access layer that represents the content (a data-centric approach). The element View, as in the classic MVC pattern, refers to all elements displayed by the GUI, such as buttons, labels, and other controls.

The View model is a “model of the view” meaning it is an abstraction of the view that also serves in mediating between the view and the model, which is the target of the view data bindings. It could be seen as a specialized aspect of what would be a controller (in the MVC pattern) that acts as a converter that changes model information into view information and passes commands from the view into the model. The view model exposes public properties, commands, and abstractions. The essential relations between the elements are shown in Fig. 4.

Namely the use of this pattern allows migrating the SR-client code from Windows Phone devices to tablets and computers on Windows 8&7. The migration requires a minor version supported by WPF.

On recent development phase we implemented prototype GUI for Windows Phone. The visual representation is given in Fig. 5.

D. Symbian

The SR-client for Symbian is Qt-based [13]. The Qt framework allows cross-platform development, so the code can run with minor modifications on some other mobile platforms. The user interface is built using QML. It provides a declarative way to construct highly dynamic user interfaces. Fig. 6 shows the architecture of Qt-based SR-client, where the focus

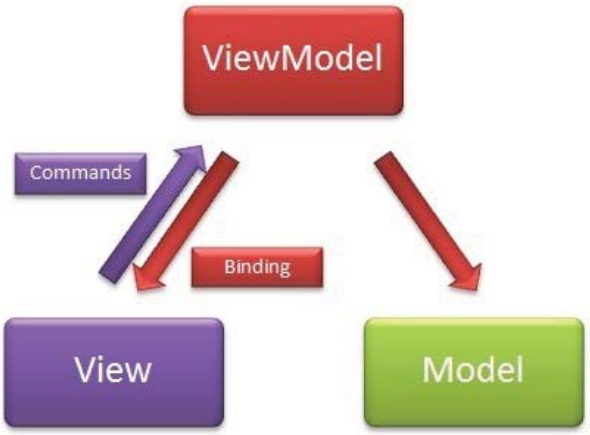

Fig. 4. MVVM relations

is on platform-aware components. We use QML for implementing GUI. The application logic is implemented in C++, and can be ported to some other mobile platforms.

Visual elements within the QML part are responsible for displaying the Smart Room services. Each is represented with a web page built using HTML5. The pages in turn are displayed from their URL by the WebView QML element contained in the QtWebKit module. Online Internet connection for the client is not essential, since SR web services uses the local storage capability of HTML5.

Smart Space data access module is within the C++ Engine part and provides the basic functions to operate with the shared smart room space. In particular, it includes caching in the local structure of SmartSlog SDK [5]. This module also extracts the URL web services, user photos, and presentations to download them on mobile device.

On the client side, the HTTP file sharing engine creates an HTTP server by QtNetwork Module. Hence other clients can access media files (e.g., photos, presentations) over HTTP protocol, herewith each client has a unique IP address. This method is an effective mean of exchanging files between distant devices.

IV. Conclusion

The paper presented results of our initial study for development of multi-platform and multi-service Smart Room clients. We identify feasible common parts of the client design. It includes separation onto basic modules: GUI, smart space access, and application module. For each module a common high-level object-oriented model can be used. Although basic GUI must be implemented separately for each mobile platform, the common solution is representation of a service as a separate HTML5 page. In the multi-service case, service navigation and control uses an advanced screen with tabs. We implemented the above solutions in prototypes of SR-client for Windows Phone and Symbian.

Acknowledgment

This research is a part of grant KA179 “Complex development of regional cooperation in the field of open ICT innovations” of Karelia ENPI programme, which is co-funded by the European Union, the Russian Federation and the Republic of Finland. The article was published with financial support from the Strategic Development Program of Petrozavodsk

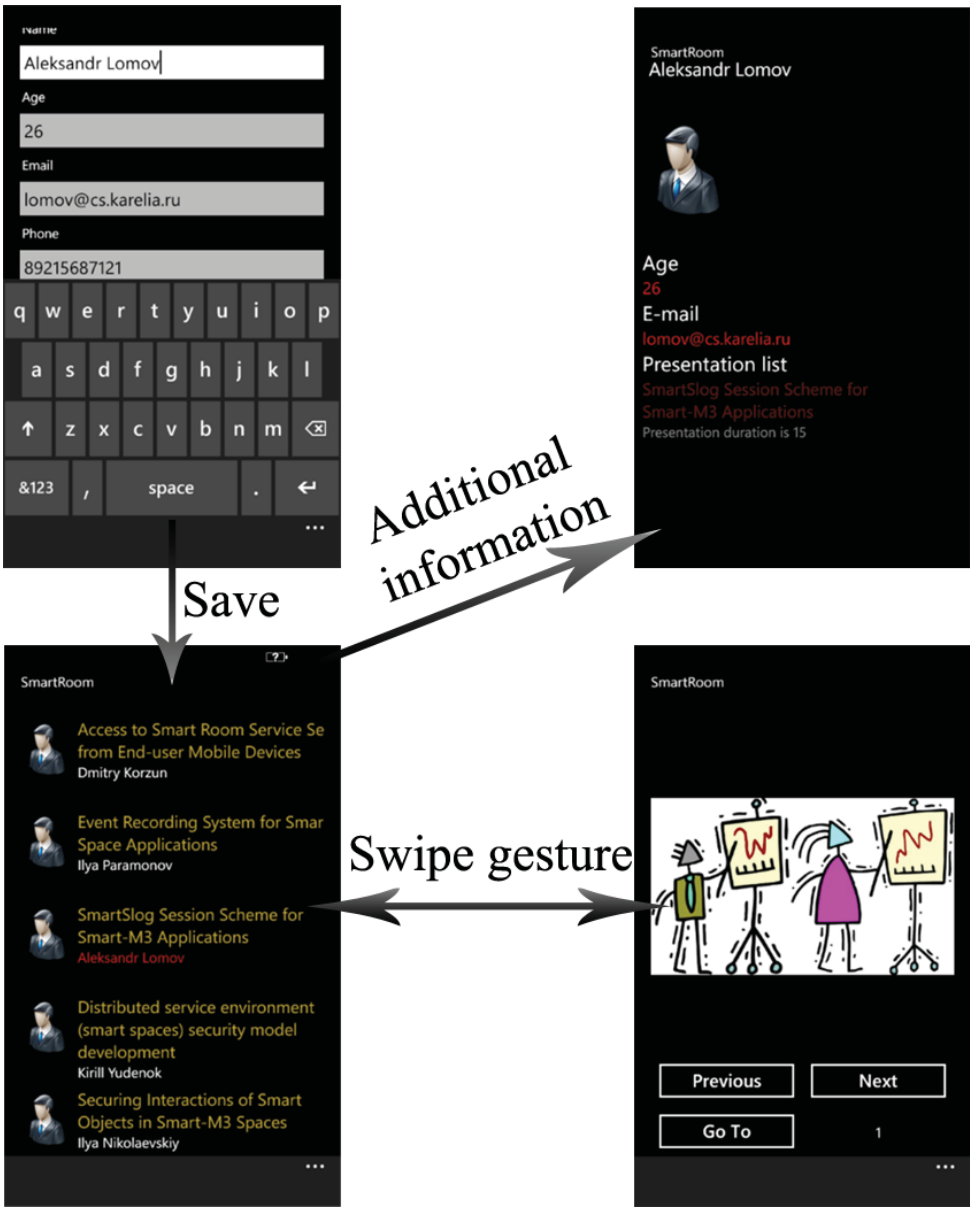

Fig. 5. Windows Phone beta GUI

State University. We are grateful to the Open Innovations Association FRUCT for its support and R&D infrastructure. We would also like to thank Sergey I. Balandin and Iurii A. Bogoiavlenskii for their feedback and expertise.

REFERENCES

[1] I. Oliver, E. Nuutila, and S. Törmä, "Context gathering in meetings: Business processes meet the agents and the semantic web," in *The 4th Int'l Workshop on Technologies for Context-Aware Business Process Management (TCoB 2009) within Proc. Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems*. INSTICC Press, May 2009.

[2] I. Galov and D. Korzun, "Smart room service set at Petrozavodsk State University: Initial state," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*, Nov. 2012, pp. 239–240.

[3] D. J. Cook and S. K. Das, "How smart are our environments? an updated look at the state of the art," *Pervasive and Mobile Computing*, vol. 3, no. 2, pp. 53–73, 2007.

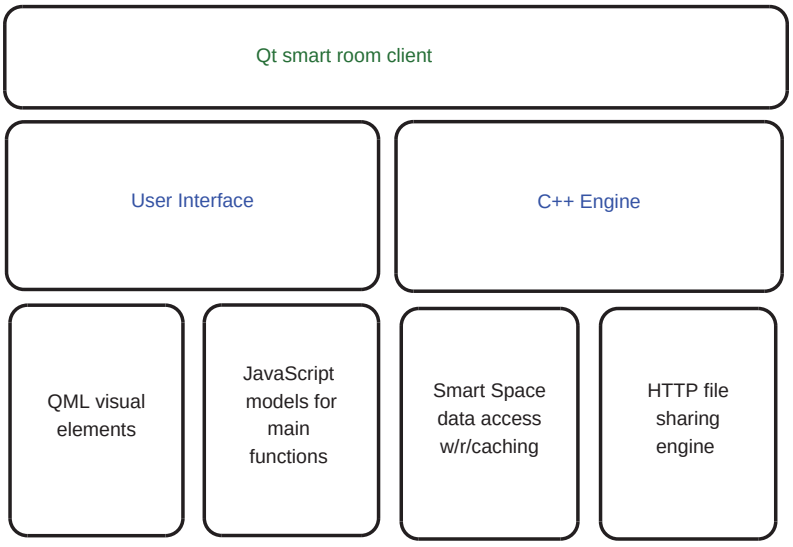

Fig. 6. Architecture of Qt-based SR-client

[4] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proc. IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Computer Society, Jun. 2010, pp. 1041–1046.
[5] D. G. Korzun, A. A. Lomov, P. I. Vanag, J. Honkola, and S. I. Balandin, "Multilingual ontology library generator for Smart-M3 information sharing platform," *International Journal on Advances in Intelligent Systems*, vol. 4, no. 3&4, pp. 68–81, 2011.
[6] A. Vdovenko and D. Korzun, "Access to smart room service set from end-user mobile devices," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*, Nov. 2012, pp. 300–302.
[7] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, and A. V. Gurtov, "Overview of Smart-M3 principles for application development," in *Proc. Congress on Information Systems and Technologies (IS&IT'11), Conf. Artificial Intelligence and Systems (AIS'11)*, vol. 4. Moscow: Physmathlit, Sep. 2011, pp. 64–71.
[8] D. G. Korzun, I. V. Galov, and S. I. Balandin, "Proactive personalized mobile mutliblogging service on SmartM3," *Journal of Computing and Information Technology*, vol. 20, no. 3, pp. 175–182, 2012.
[9] K. Rustam, C. Evgeny, and D. Korzun, "Sensors in a Smart Room: Preliminary study," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*, Nov. 2012, pp. 37–42.
[10] D. Korzun, I. Galov, A. Kashevnik, K. Krinkin, and Y. Korolev, "Integration of Smart-M3 applications: Blogging in smart conference," in *Proc. 4th Conf. Smart Spaces (ruSMART'11) and 11th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'11)*. Springer-Verlag, Aug. 2011, pp. 51–62.
[11] I. Paramonov, A. Vasilev, N. Kozhemyakin, I. Timofeev, A. Subbotkin, E. Krylov, D. Korzun, and I. Galov, "Multiplatform client and pecha kucha support for smart conference system," in *Proc. 11th Conf. Open Innovations Association FRUCT*, Apr. 2012, pp. 197–198.
[12] Y. Kiriaty and J. Rodriguez, *Learning Windows Phone Programming: Building Apps with Silverlight and XNA Framework*. O'Reilly Media, 2011.
[13] F. H. P. Fitzek, T. Mikkonen, and T. Torp, *Qt for Symbian*. Willey, 2010.