Design and Implementation of the First Aid Assistance Service Based on Smart-M3 Platform

Nikolai Lebedev*, Ivan Timofeev**, Iuliia Zavialova*
*Petrozavodsk State University (PetrSU), Petrozavodsk, Russia
*P. G. Demidov Yaroslavl State University, Yaroslavl, Russia
{lebedev, yzavyalo}@cs.petrsu.ru, skat.set@gmail.com

***Abstract*—Smart technologies may be successfully applied in healthcare for creation of an IoT-enabled proactive pre-hospital and first aid assistance mobile services. A variety of smart services for the m-Health scenarios may be constructed by interaction of multiple knowledge processors (software agents) running on devices of the IoT environment. Thus, IoT-enabled m-Health applications should provide connection with smart space. It is possible to build such kind of services with Smart-M3 platform. The ontology describes interaction rules and the high-level design of the service. The first aid assistance scenario was chosen as a basic one. According to this scenario, sympathetic people provide first aid to patients in case of emergency. The study is focused on the implementation of the first aid assistance service consists of knowledge processors running on Linux servers and Android mobile devices. Such service should be scalable with adding new modules, sensors or participants. The purpose is to evaluate a possibility of application of a smart spaces approach for implementation mobile first aid services. Besides, implementation issues of server and client sides are discussed.**

I. Introduction

According to recent heart disease and stroke statistics, the survival rate for out-of-hospital cardiac arrest (OHCA) is extremely low and amounts from 7 to 11 in developed countries [1], [2]. Therefore, the sudden cardiac death is an important public health problem and the efforts of governments are directed to increase survival rates after cardiac arrest. In particular, one of the objectives of governmental healthcare development program in Russian Federation is the enhancement of the emergency care.

It is proven that the survival from OHCA is utterly time-sensitive and in case of immediate treatment the chance of survival is roughly 67%. However, it rapidly decreases and after 12 minutes the patient dies almost inevitably [3]. Due to the hospital locations, traffic conditions and lack of free ambulance staff, the ambulance response time may vary over a wide range.

Nevertheless, the first pre-doctor aid in emergency situation can generally be provided by sympathetic people nearby, in other words, by the volunteers. According to law, first aid is defined as a set of simple actions which any person can take regardless of education level or a special training. Laws also affirm the right to provide first aid even without medical training in case of trauma and emergencies directly at the place of the accident before the arrival of medical personnel.

Modern mobile technologies and, particularly, location features provide a decelopment of patients' location control system. It makes possible to introduce a sending panic signals automation, e.g. in cases, when patient feels bad or has no opportunity to send an alarm by himself. Such kind of automation provides more effective care because of the rate of pre-doctor first aid increasing. As an example of this functionality, CardiaCare application was developed in Petrozavodsk State University [4]. To provide this automation, wearable devices and environment sensors are used. Besides, the data may be collected from big variety of such sensors. [5]

Moreover, the first aid quality may be improved by using the personalize treatment. To provide this feature, the first aid service may be extended by using electronic health records (EHR) and electronic medical systems (EMS) [6]. In case of medical personnel involving to the smart first aid service, EHR communicated with EMS allows to get quick access to patients' personal data. According to this data, medical care may be more efficient and amount of medical errors may decrease. Consequently, set of location control system, EHR and EMS allow to build the service that provides a complex first medical aid and a first pre-doctor aid.

All properties described before may be successfully implemented with the idea of IoT and smart spaces approach [7]. The concept was researched and proposed in [8]. But this concept is necessary to be checked for the possibility of an implementation. Moreover, it would be better to provide smart space's participants independence on the OS and devices. In this paper the basic scenario is investigated, the detailed architecture and the implementation approach is proposed.

In Section II, the evolution of the concept of the first aid service based on smart spaces approach, high-level design and needed instruments are described. Section III exposes ontological model which contains shared information and describes the relationship between smart space participants. In Section IV the architecture, its advantages and drawbacks are discussed. In Section V implementation details and difficulties of KPs work in Android applications are considered. In Section VI the conceptual advantages of the proposed architecture and present future work related to this project are described.

II. The Smart m-Health Service Concept Extending

The concept of the smart m-Health service was presented in [8]. According to it, there are several basic scenarios with an evident application of the smart approach. Summarizing, smart m-Health services support a personalized digital assistance in emergency cases for mobile patients.

The most evident scenario to be implemented is when volunteers help needed patients. A patient can ask for a help by pushing a panic button manually or sending a panic signal automatically (in case of some wearable devices for health state control usage). This scenario is more suitable for an implementation because of the difficulties with the service deployment in a real environment if the medical personnel is involved. The goal of such implementation trying is to develop a basic model of the first aid service that may be extending with necessary modules.

The design of the first aid service is based on the smart spaces approach [7]. In general, it is a set of interacting program agents called "knowledge processors" (KP) and the semantic information broker (SIB) that provides the required interaction. In substance, every KP may be implemented as an independent application or it is possible to use a group of KPs in a single program. The platform that provides such functionality is called Smart-M3 [9]. According to it, the interaction goes through the separation of a content represented by the RDF model (Resource Description Framework) [10]. Smart-M3 supports an operation of subscription [11]. It is the constant request to SIB for tracking changes of the data. Also, it provides one-off operations with RDF-triples (read, insert, change, delete) and search queries using SPARQL [7].

The first aid service is assumed to describe a big variety of triples for different situations. The way-out is to use ontologies. The structure of the proposed service ontology is described in the Section III. The library that allows to work with ontologies in a smart space is called SmartSlog [12]. SmartSlog is a software development tool for programming KPs and their interaction in a smart space. It allows easier constructing the program code. It is easy to think in agent domain ontology terms (classes, relations and individuals) instead of RDF triples, as it happens in the low-level development approach of Smart-M3. Besides, SmartSlog Double Api library provides both high level (classes) and low-level (triples) interfaces.

These studies laid the foundation for the implementation of the smart first aid service. In [6] four models of first aid health services are proposed. They provide different high-level architectures of first aid services in depends on infrastructure and relation between Hospital Information System (HIS) and Emergency Medical Systems (EMS). In this paper, the second model is more interesting where HIS and EMS implemented in single organization and can directly communicate with each other. It allows to reduce time for patient medical record retrieving and, hence, reduce time of first aid provision. Also, in [6] authors provide base use case scenario which describes aspects of communication between HIS and EMS in details.

To sum up, the smart m-Health service is developed using ontology and KPs. It's necessary to define the backbone of the service describing basic classes, properties and applications. According to the chosen scenario, there are two participants roles in the service. The first role is a patient and the second one is a volunteer. All participants are suggest to be a real people, so the applications are mobile. As the first platform to examine proposed design the Android OS was chosen. Because the first aid requires special responsibility in the volunteers selection, the authentication provision is necessary. Finally, it's needed to provide a mapping of patients that sent panic signals to free volunteers. The architecture of the service is shown in Fig. 1.

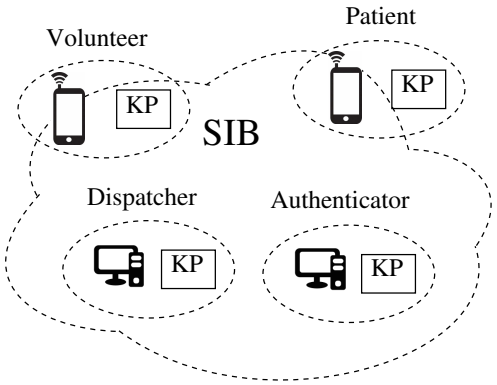

Fig. 1. Architecture of the first aid service

III. Ontological model

To describe and use the proposed scenario in KP agents the ontologies are used. Ontology [13] is the convenient way to present the architecture and use cases of the service. It allows to build a semantic representation and a mapping in the smart space of complex parameters of a patient's health and medical services to the distribution of powers of access to personal medical data service. It represents a data structure that contains all the relevant classes of objects, their links, adopted in the field of a subject domain.

Ontology unites the smart space participant's information: authentication, help requests and responses, locations and other necessary information. It describes relationships between program agents KP. All the features of the service are displayed in the ontology. Classes, subclasses, data properties and object properties of the ontology are shown in Fig. 2.

As a result, the virtual integral representation of current state and processes for patients and volunteers are created in smart space. The information exchange is made possible by the first aid assistance service construction in case of an emergency delivery.

First of all, the aim is to define participants' roles in KPs, to provide their connection and authentication in smart space. Each user has their own personal profile. Profile is represented by two ontological classes *Person* and *Profile*. Class *Person* contains the general information about the user. It is based on the ontological model FOAF: *name*, *status*, *img (picture)*, *age*, *e-mail*, *phone number*.

Class *Profile* provides system information about service's users. It defines user's presence status in the service (online / offline). *Username* and *password* properties are used to users authentication.

The next set of ontological classes allows to describe users roles: *Patient*, *Volunteer*. Each role is presented as a separate *Person's* subclass. Consequently, the ontology and the service may be extended by adding new classes similar to these ones. Thus, the power access distribution to personal medical data is implemented directly at the ontological level.

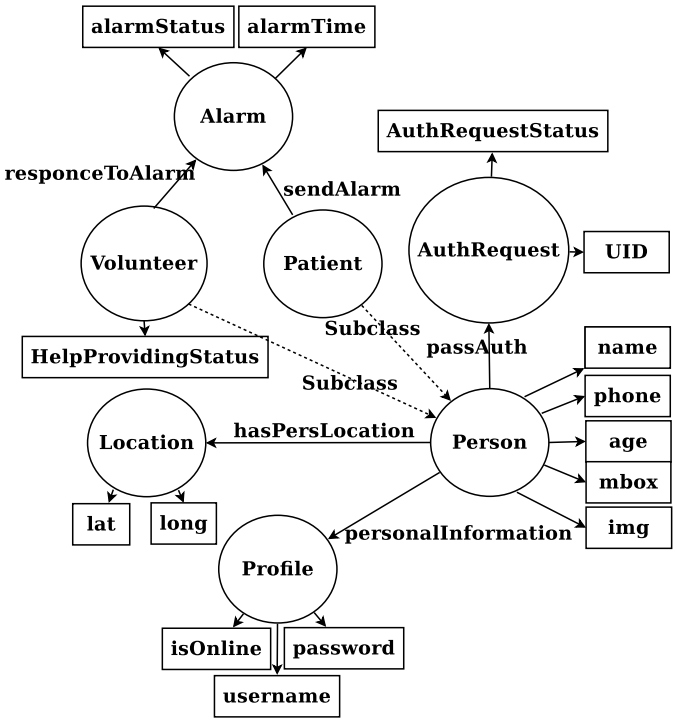

Fig. 2. Ontology

To provide safety of patients data and to control a qualification level of volunteers an authentication methods are necessary. Class *AuthRequest* is required for user authentication. User classes establish the contact with them through the property *passAuth*. Authenticator agent subscribes to a publishing this property in a smart space. Property *UID* is a unique key, which is checked by "Authenticator" KP. After checking the property *authRequestStatus* is set to "PASSED" or "NOT PASSED".

The general use case of first aid service is when patient publishes a panic signal and volunteer gets notification about this. So, "Emergency detection" event is described by class *Alarm*. It is associated with the class *Patient* — patient who have an emergency. The property *alarmTime* is a time when the emergency situation has arisen, and the message about the incident is provided by the *alarmStatus* property.

The *Location* class presents location in the form of geographical coordinates. It is represented by properties latitude (*lat*) and longitude (*long*) and correspond W3C Geo. Ontology defines the location of every user and links Person and Location classes by *hasPersLocation* property. The location definition is necessary for providing an oportunity of route building and for simple distribution of patients in need between volunteers.

Certainly, to provide correct distribution classes and properties for interaction of a patient and a volunteer are crucial. It's needed to provide a choise opportuity to a volunteer. He can confirm or reject a panic signal, then alarm should be forwarded to another KP. According to this scenario the *helpProvidingStatus* and the *helpProvidingStatus* properties were introduced.

Volunteer can deliver helpfulness status *helpProvidingStatus* with the following possible values "READY TO HELP", "HELP REQUEST", "CONFIRMED", "REJECTED". It is necessary to search for the nearest volunteer. It is selected volunteers who set property *helpProvidingStatus* with a value of "READY TO HELP". This property value is changed to "HELP REQUEST" in the case of assistance request and the agent "Dispatcher" chooses this volunteer. Also, class *Volunteer* communicates with the *Alarm* through *responseToAlarm* property. Volunteer sets the value of this property to "CONFIRMED" in case of confirmation, and in case of failure sets "REJECTED".

The domain ontology has been represented in the OWL language. The ontology contains 7 classes (concepts), 15 data properties and 5 object properties which indicate the relationships among the 7 classes. From the OWL-representation the ANCI C ontology source code may be generated with the SmartSlog CodeGen instrument. The ontology file in ANCI C contains all classes and properties definitions in the form of SmartSlog structures. This way is used for further KPs implementation described in the next Section.

IV. The service design

First aid assistance service use cases are referred in [8]. They are based on the model of the problem domain according to the ontological model described in the Section III. All applications of the mobile service may be divided to server side applications and client side ones. Server side applications are run on the local machines. Their functions are to provide authentications of all client agents and to distribute patients between volunteers. Client agents work on mobile devices for a provision of a continuous monitoring of a participant state. Next, every side of the mobile service is considered more.

A. General program agent's workflow

The scenario of every program agent of a first aid assistance service consists of several basic steps. Firstly, it's necessary to get smart space access identifier calling smart space connection initializing function. If the authentication is successful, user's profile (ID, name etc.) is published into a smart space. Secondly, a program agent needs to publish its location data to smart space. So the user's location initializing function is called. This function receives coordinates of an user's mobile device, converts them to text format for further transmission to the shared storage SIB. Besides, locations may be updated when "onLocationChanged" is generated. It's important to update locations permanently to provide a correct work of a mapping patients between volunteers algorithm.

Thirdly, programming agent do some actions that it needs according to its purpose. For example, it may be a subscribing for some properties of another agents. For patients it may be a connecting with some sensors. For volunteers it may be an interaction with medical personnel. Finally, before a client application is stopped, a program agent needs to call the smart space connection closing function to close all subscriptions and remove its unnecessary data from SIB.

B. Server applications

The main purpose of server applications in the first aid assistance service is to provide interaction of distributed participants using via some common KPs. First of all, server

applications are for a making decisions about this interaction, e.g. “what volunteers should help this patient”. At present, server application consists of two KP agents: “Authenticator” and “Dispatcher”.

An architecture of the server-side applications is shown in Fig. 3. The main component of the applications is a handler module. It reacts on new events and performs corresponding actions. A handler uses functions from utils module. This module is common for both server applications and contains high-level function to simplify work with smart space. Detailed description of behaviour for each of server-side application is discussed in next sections.

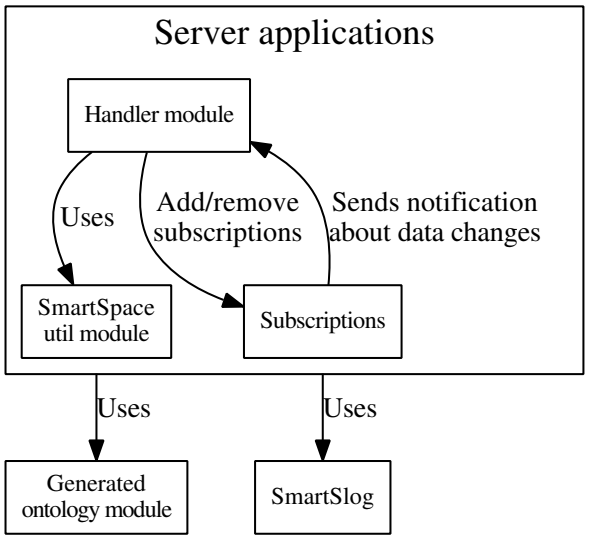

Fig. 3. Design of a server application

Applying a presented architecture approach of building server-side applications new server KPs may be implemented to extend the first aid service functions. It is worth noting that such approach may be applied not only for first aid services: e.g. an authentication provision in multi-agent systems is an important task, but SmartSlog library or SIB doesn’t implement this function. Also, intellectual distribution of notification between participants may be applied in another smart services.

1) “Authenticator”: In the start of operation the agents initialize subscriptions: the agent “Authenticator” subscribes for “AuthRequest” class instance adding and the agent “Dispatcher” subscribes for “Alarm” class instance adding. Corresponding handler is called, when another agent adds new instance of these classes.

To authenticate “Volunteer” and “Patient” agents service applies the following algorithm: firstly, volunteers publish a request for authentication and the unique identifier (UID). The agent “Authenticator” subscribes for “AuthRequest” class instance adding. Then it verifies the identity against a list of valid UIDs. If the identifier is found, the agent “Authenticator” confirms the passage of the agent. Otherwise, the agent “Authenticator” exposes the authentication request property to the state “Failed”. The detailed diagram of the authentication process is presented in Fig. 4.

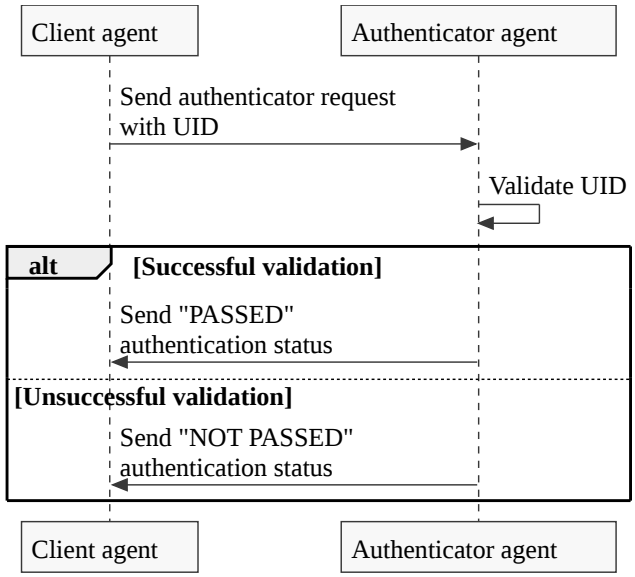

Fig. 4. Authentication process

2) “Dispatcher”: The service implements a publishing and a subscription to assistance requests algorithms. In the start of operation agent Dispatcher subscribes for Alarm class instance adding.

When a patient reports about feeling unwell, agent “Dispatcher” checks clients authentication during the process of a request for help. If the client has been authenticated, the agent “Dispatcher” continues working with it, otherwise skips the client. Next, “Dispatcher” absorbs information about free volunteers, then determines the nearest volunteer , who is more suitable to come to the patient. After that, “Dispatcher” agent sends a help request to the selected volunteer. Then he can accept an invitation to go for help or refuse it. In case of positive response, a volunteer is reserved for a patient who called for help. Then volunteer receives the coordinates of this patient.

In the case of the volunteer’s consent agent “Dispatcher” finishes work. Volunteer may be busy or helps to other patient, the “Dispatcher” agent continues search for the next nearby volunteer and repeat the above procedure. If there are no free volunteers, the “Dispatcher” agent completes its work.

C. Client applications

The client application in the first aid assistance service is a set of KP agents (one or many) to provide an interaction of a participant of the first aid treatment scenario with other ones. It may be a patient, a volunteer, an ambulance, a pharmacy application and others.

Actually, both of client-side applications are very similar, except of some specific functions. Every client-side application has a module responsible for location control, because each participant of first aid scenario should publish its coordinates into a smart space. To provide basic security methods function for publishing and receiving authentication requests should be implemented in any client application.

Besides, every KP should interact with a smart space according to a defined scenario that was described above, so basic procedures for working with SIB are similar. In this way, such abstract design of client application for first aid assistance service may be applied not only in existing programs, but in any new ones. Two client applications for the first aid service were developed using the described techniques. At present, prototypes of a "Volunteer" application and a "Patient" one were developed. They are respectively based on KP agents "Volunteer" and "Patient". Next, every application is considered more. The current model of a client application's architecture is shown in Fig. 5.

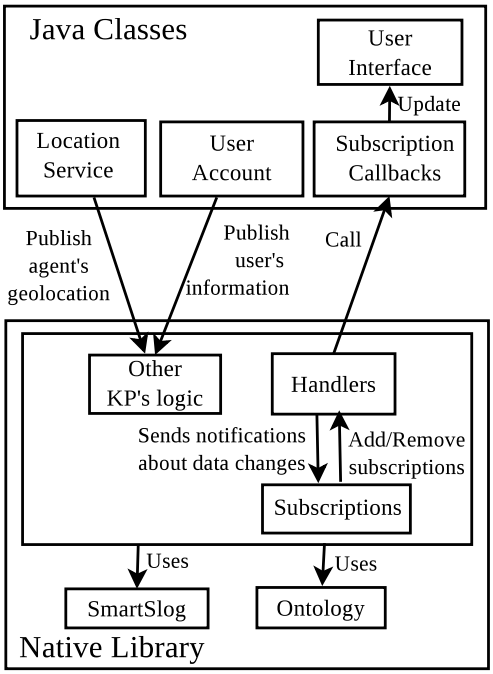

Fig. 5. Design of a client application

1) "Patient": The "Patient" application based on the scenario of a help signal sending from a patient to a volunteer. There are two specific modules in this application. The first one is the "Help request" module. This module is responsive for publishing of a help request to a smart space when an "Alarm" button was clicked. Secondly, it is a "Help response" module. When an "Alarm" button is clicked with help request published to smart space, KP need to subscribe to a volunteer's response. According to the threads model described above, it is needed to create a function for handling response receiving.

2) "Volunteer": The "Volunteer" application has a more complex structure because of different background tasks with subscriptions. Firstly, "volunteer" agent has to subscribe to help requests. When the help request is received, the "volunteer" agent needs to publish his response. If the response is positive, agent has to get a patient's location. Also, it's needed to subscribe to location's changes. Finally, the route is generated using the Location Service of a "volunteer" application. This route is rebuilt when patient's location is changed.

V. The Client Applications Implementation Details

In general, each of client applications consists of Java part and native part. All user interfaces methods, location's control function are implemented in Java part. All KP logic is implemented in the native part. Android NDK and ANSI C to develop native parts of clients applications are used. Now Android SDK with NDK provides tools for editing, running and debugging C and C++ code, but with some issues.

Certainly, a KPs development for Android OS is associated with some difficulties. Two main issues that were investigated are the code organizing to provide KPs cross-platform and scalability, and working with subscriptions as a background tasks. Next, these problems are discussed more.

A. Integration of KPs into Android application

The smart space interaction functions are provided with a native library of a client application. The native library is a set of several modules. Every module implements a group of functions, e.g. "connection to smart space" or "handling alarm subscription". All functions are divided between modules depending on their purposes.

Besides, KP's logic is placed in several files too. The main file contains all needed function to work with a smart space using SmartSlog library. All handlers for subscriptions are described in another file. So, if it's needed to change an application behaviour after subscription's notification.

One of issues of the implementation of a Smart-M3 platform-based application for Android is keeping some identifiers permanently while the application is running. For example, it may be a JVM pointer, some global references to Java Methods or some SmartSlog identifiers such as the smart space Node ID. It is convenient to keep such kind of variables as global ones in separate file.

Last module plays role a JNI wrappers above KP's Logic. So, in this way all KP's behaviour is coded on pure ANCI C and it may be use detached from Java part. Moreover, this method of building KPs for Android OS is convenient for further testing of application methods. All in all, a scalability is also provided and it is important feature, because first aid service may be extended by new modules implementation, e.g. for wearable devices or environment sensors support.

In this way, an instrument to compile all of this modules to get a dynamic library is needed. This problem was solved with describing all files and catalogues in Android.mk file and running "ndk-build tool". It's not the best way, but the Gradle plug-in cannot manage compiling of library consisted of several files correctly in current version of "Android Studio".

B. Subscription control

There is another important problem in the developing of a client application for smart spaces using JNI. It is needed to provide an optimal way to subscriptions control. On the one hand, it is needed to implement a separate thread to work with subscriptions in Java code. On the other hand, the SmartSlog library provides callbacks to handle notifications of subscriptions. Such callbacks or handlers are run in separate

thread too. As a result, there are some issues with correct controlling of threads described in C and in Java independently. For example, subscriptions handlers run by SmartSlog library in native part can't get an access to JVM instance and Java methods pointers. Also, Java callbacks called from SmartSlog handlers are run not in UI thread, so there are threads collision and application error as a result. Some ways to solve this problem were analysed.

The first and the easiest way is to call subscription functions directly from User Interface (UI) thread. Obviously, such method is not appropriate, because almost every subscription generate an infinity loop that hangs application out. This way is suitable just to get some data from shared space once.

The second way is to use the Android AsyncTask. The concept of this way is to describe a new AsyncTask instance for every subscription. This way is not appropriate either because it is impossible to run various AsyncTask's instances without an additional controlling mechanism. This method is used for only to 1-2 subscriptions. Then it can control their switching.

The third way is to use an Android Background Service. This method is the preferable but it's the most labor-intensive task. In the production mode of application all subscription should be implemented in this way because a patient and a volunteer need to be on-line every time to send and receive panic signals. On the other hand, his way requires to investigate a power consumption when several subscriptions are run.

The temporary trade-off was found to solve a problem of synchronous work of a several subscriptions in a background. The solution is shown in Fig. 6. Handlers both in Activity (Java handler) and native part of a client application are created. The C-handler is linked as a callback to a subscription using SmartSlog library functions. When C-handler is initialized, it calls a Java-handler. Further, it calls "runOnUIThread()" method in Java-handler and do other actions. A "runOnUIThread()" method is needed here to avoid errors with running UI tasks out of a context. This method doesn't provide an application work in a background but it works well for a prototype.

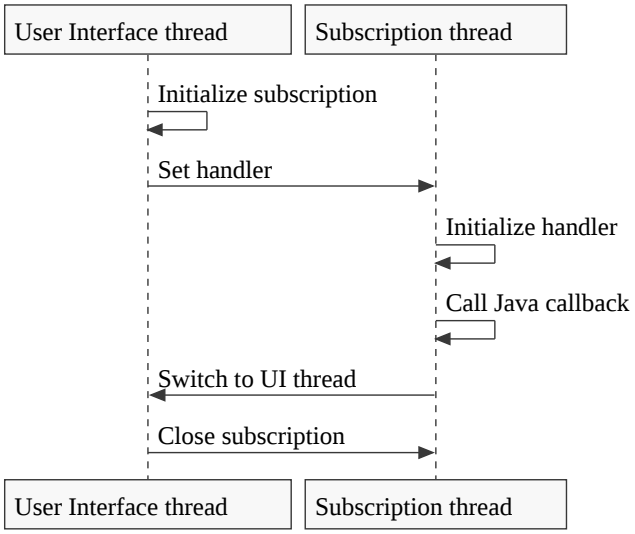

Fig. 6. Threads interaction in client applications

The experience described before may be implemented in Android applications based on Smart-M3 platform. Depending on the way of receiving data from a smart space, any of methods above may be used, but for long-term tasks the using of a background service is the preferable one.

VI. Current results and future work

Design of mobile smart applications based on KP agents and their interaction with server-side ones was developed. Program agents were written on ANSI C language and may be compiled and launched at any Linux-powered machine with use of the GCC compiler. However, the agents may be used in any other platform and be compiled by another compiler.

The ontological model prototype was developed and integrated into the service. At present, all KPs have to use the static variant of an ontology file to provide a mutual compatibility. The program agents use generated ontology module where relations between ontology classes present.

It is assumed that smart spaces development approach based on Smart-M3 platform provides an extending possibility with new KP agents implementation. KP agents may be a health information systems, wearable sensors or other participants of this interaction, e.g. pharmacies. To provide new agents' correct work the ontology scaling is required. It may be due to some problems such as all parts updating of the service with new ontology module. It's needed to research a problem of a dynamic ontology provision.

Besides, it is necessary to investigate working with wearable devices. The main purpose of using such kind of devices is an automatic panic signal sending when patients can't do it by themselves. But there are some problems that needs to analyse. For example, it is a limit of a number of devices connected to smartphone via bluetooth at the same time and ways to avoid collisions. Also, it's important to provide a plug and play way of wearable devices usage.

Server-side applications may be improved either. For example, agent "Dispatcher" may provide a hospital coordinates publishing for ambulance workers. To get it done, the "Dispatcher" should choose appropriate hospital. As noted in [6], there are several parameters of hospitals which should be reviewed: a location, a specialization, an availability of beds, equipment etc. Moreover, new agent should be implemented in a health information system (e.g. in a hospital) to provide patient's medical records for ambulance workers during first aid provision. According to this data, it's needed to make a mapping algorithm improvements to take into account such information as road traffic, a severity of a patient's health state, etc.

In "Authenticator" agent's turn, it may be improved by modernizing algorithm of token's distributions to avoid chances of non authorized access and several kinds of collisions when the server will be loaded with participants. For example, two patients try to authorize at the same time. Also, a situation takes place when a participant wants to use not only one device to use the service. Another example of collision may be a scenario when an user try to authorize in the service with UID used before or with another device.

Testing and conducting experimental research of KPs prototypes is planned. Finally, the described prototype of the first

aid assistance service requires an approbation in live situations (or in situations closed to it). On the one hand, it will allow to detail hardware and system requirements of the service for further deployment in a real environment. On the other hand, it will give an opportunity to get a patients' and medical personnel's feedback to improve usability of the service in future.

VII. Conclusion

This paper presents the ongoing work that is conducting for the development of the first aid assistance service based on Smart-M3. The main purpose of this service is to provide an interaction of different participants of first aid treatment such as patients or volunteers. In this paper it is aimed at developing design of mobile smart applications based on KP agents run on Android devices and their interaction with server-side ones. As a result, 4 prototypes were developed. Further work is required for scaling current ontology by adding new classes. The development of the existing KPs will be continued too by adding new modules, sensors support implementation and approbation in a real environment.

Acknowledgment

This research is financially supported by the Ministry of Education and Science of the Russian Federation within project # 14.574.21.0060 (RFMEFI57414X0060) of Federal Target Program "Research and development on priority directions of scientific-technological complex of Russia for 2014–2020".

The study is financially supported by Russian Foundation for Basic Research 16-07-01289.

References

[1] D. Mozaffarian, et al. "Executive Summary: Heart Disease and Stroke Statistics2015 Update A Report From the American Heart Association", *Circulation*, 2015, pp. 434-441.

[2] European Society of Cardiology, "Out-of-hospital cardiac arrest survival just 7 percent.", ScienceDaily. ScienceDaily, 1 September 2013.

[3] B. Sund, "Effect of response times on survival from out-of-hospital cardiac arrest: using geographic information systems", 2010.

[4] A. V. Borodin, A. Pogorelov and Y. V. Zavyalova "CardiaCare. Mobile System for Arrhythmia Detection", in *Proc. of 13th Conf. Open Innovations Association FRUCT*, Petrozavodsk, Russia, 22-26 Apr. 2013

[5] A. V. Borodin, Y. V. Zavyalova, A. Zaharov, I. Yamushev "Architectural Approach to the Multisource Health Monitoring Application Design", in *Proc. of 17th Conf. Open Innovations Association FRUCT*, by ITMO University, Yaroslavl, Russia.

[6] I. V. Paramonov, A. V. Vasilyev A., I. A. Timofeev, "Communication Between Emergency Medical System Equipped With Panic Buttons and Hospital Information Systems: Use Case and Interfaces", in *Proc. of the AINL-ISMW FRUCT*, Saint-Petersburg, Russia, 9-14 November 2015.

[7] D. G. Korzun, S. I. Balandin, A. V. Gurtov "Deployment of Smart Spaces in Internet of Things: Overview of the design challenges", in *Internet of Things, Smart Spaces, and Next Generation Networking*, Springer Berlin Heidelberg, 2013. pp. 4859.

[8] D. G. Korzun, A. V. Borodin, I. A. Timofeev, I. V. Paramonov, S. I. Balandin "Digital Assistance Services for Emergency Situations in Personalized Mobile Healthcare: Smart Space based Approach", in *Biomedical Engineering and Computational Technologies (SIBIRCON)*, 2015, pp. 62-67.

[9] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, A. V. Gurtov. "Overview of Smart-M3 principles for application development." *Proc. Congress on Information Systems and Technologies (IS&IT11), Conf. Artificial Intelligence and Systems (AIS11)*. Vol. 4. 2011.

[10] O. Lassila, R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation*, February 1999.

[11] A. A. Lomov, D. G. Korzun, "Subscription operation in Smart-M3". *Proc. 10th Conf. of Open Innovations Association FRUCT and 2nd FinnishRussian Mobile Linux Summit*, 2011, pp. 83-94.

[12] A. A. Lomov, D. G. Korzun "Evaluation of Program Code of Smart-M3 Knowledge Processors Developed Using the SmartSlog Tool", in *Proc. of the 16th Conference of Open Innovations Association FRUCT*, ITMO University, St-Petersburg, Russia.

[13] D. Korzun, A. Lomov, P. Vanag, J. Honkola and S. Balandin, Generating Modest High-Level Ontology Libraries for Smart-M3, UBICOMM 2010. Florence, Italy, October 2010. pp. 103-109.