EgoSENSE: A Framework for Context-Aware Mobile Applications Development

Eleonora Milić
Department of Computer Science
Faculty of Electronic Engineering, University of Niš
Niš, Serbia
eleonora.milic@elfak.rs

Dragan Stojanović
Department of Computer Science
Faculty of Electronic Engineering, University of Niš
Niš, Serbia
dragan.stojanovic@elfak.ni.ac.rs

***Abstract*—This paper presents a context-aware mobile framework (or middleware), intended to support the implementation of context-aware mobile services. The overview of basic concepts, architecture and components of context-aware mobile framework is given. The mobile framework provide acquisition and management of context, where raw data sensed from physical (hardware) sensors and virtual (software) sensors are combined, processed and analyzed to provide high-level context and situation of the user to the mobile context-aware applications in near real-time. Using demo mobile health application, its most important components and functions, such as these supposed to detect urgent or alarming health conditions of a mobile user and to initiate appropriate actions demonstrated.**

Keywords-context; context-awareness; mobile services; personal health monitoring;

I. INTRODUCTION

Context-aware systems take into account the current state of a user and, in addition, his environment, enabling a mobile device and the application to adjust in an appropriate way. The development of affordable sensors with low power consumption (such as accelerometer, gyroscope, digital compass, light sensor etc.) and their integration into modern mobile devices coupled with the recent advances in machine learning, enabled the creation of a more extensive model of user context often used for health monitoring [1]. Usage of context in remote medical monitoring is considered as an important innovation for reducing healthcare delivery costs and improving the responsiveness of health care. Remote healthcare monitoring involves the use of smart devices which aggregate data from multiple sensors and transmit the data to the remote server in order to generate high level context.

Existing solutions collect patient's sensor data and transmit that data without any local processing to the remote server for real-time or offline analysis. To significantly reduce the flow of high volumes of collected data and in order to reduce the processing of that data on the remote server we propose a mobile context-aware framework for Android smart devices where context data processing and analysis is performed locally on the device itself and appropriate actions are initiated based on defined changes in the user's context. Only necessary context data is sent to the remote server for further use. Our goal is to collect data from physical and virtual sensors on mobile devices which provide the basic context and enable storage, process and analysis of contextual data locally in order to detect the context of a higher level and the situation of users on mobile devices.

The subject of this paper, and its realization in practice, is the study of methods and techniques for gathering and detecting user context on smartphones, based on sensors integrated in a mobile device (such as GPS, accelerometers, temperature, motion detector and sensors for activity recognition etc.), user's preferences, his/her environment and activities. The framework for context data collection, organization, processing and analysis, needed for generation of high-level context information, is developed. The appropriate techniques and components within the framework are implemented on Android smart devices, as support for the development of context-aware mobile applications and services. To provide usefulness and effectiveness of the proposed framework we implemented a demo mobile health monitoring application for Android platform, based on Funf [1] and Asper[2] open source technologies, as well as sensors for physical activity, pulse, pressure, temperature and motion detection. EgoSENSE framework and Personal Health Monitoring Android application are supposed to detect urgent or alarming health conditions of a mobile user and initiate appropriate actions by using the implemented framework and background services.

II. PROBLEM STATEMENT AND RELATED WORK

The notion of context is defined in numerous fields of science, such as: linguistics, philosophy, knowledge representation and problem-solving in artificial intelligence and communication theory. There are different definitions of context. According to the Free Online Dictionary of Computing, context is something which encircles and provides the meaning for something else. Another definition of context is from the field of distributed and mobile computing, where the person is that "something", and context is comprised of information of that person's environment, such as the location and identity of people and objects surrounding him/her. The

[1] http://funf.org/about.html
[2] https://github.com/mobile-event-processing/Asper

context can be used to characterize the state of an entity. The entity is a person, a place or an object that is considered relevant in user-application interaction. It is necessary to identify what kind of information fits this definition, the structure of information, the way to present it and how to use context in a specific application. Context includes information such as: location, time, status of application, resources, network flow, user's intentions and desires, activity and environmental conditions [2]. User's activity is an important part of user context since it directly influences the possibility of the user to interact with a mobile device. Context-aware systems are agents and applications that use different levels of contexts and adjust their behavior according to current contexts. In order to gather the context, a context-aware service has to ask for specific information the context provider or to monitor events sent by the context providers. In order to construct context-aware services, specific actions are most often defined to run according to a set of rules each time when the current context changes. Solution architects can write a set of rules without major problems and to specify which method will be run when a condition is fulfilled. All the rules are memorized in a file and become part of the context understanding process. This is the principle used in services for energy saving and many other services provided by smart devices [4].

Context interpreters are components responsible for context processing based on logical reasoning. Context processing may include: deriving higher level context from a low level context, knowledge discovery, maintaining consistency of knowledge and resolving conflicts. In order to provide an effective infrastructural support for the development of context-aware services in pervasive computing environment, a general architecture of a context-aware framework has been designed and described in [4]. It is a distributed middleware which transfers context from a physical into a semantic space, where the access to context can be easily divided among context-aware services. It is comprised of the following mutually independent components: context providers, context data base, context-aware services and a service for context service locating (a mechanism where context providers and interpreters can sign up their presence so that the application or users can locate them). In [6], authors proposed ContextDroid as a framework for context-aware Android phone applications. It was designed in order to provide developers all the necessary services for easier development of context-aware applications, all the while taking into account the battery usage. ContextDroid integrates several context phrases, so it generally requires applications that use the least resources possible [6]. Queries are transmitted from the application to the service and then forwarded to the sensor. The sensor then sends the acquired data back to the service, which further reasons the received data and notifies the application if needed. It was explained how the contextual information are modeled, acquired, reasoned and, in the end, delivered to the application. Framework is implemented for Android platform.

ContextPhone [8] is a software platform comprised of four different connected modules realized through a set of open source C++ libraries. ContextPhone was running on Symbian OS and was developed using an iterative human-centered design strategy. It was developed for the researches who want to acquire data in a way that is not intrusive for a user and it requires minimal control and maintenance. All the acquired data are stored in local file and periodically updated. Aware [9] is the Android framework developed for context measuring, inference, logging and sharing, and it was built for developers, researchers and regular smartphone users. Aware records hardware, software and data based on user's actions in its local memory. These data are further analyzed using plug-ins and are transformed into information that a user can understand.

Wawelite [11] is a framework for situational awareness in environment monitoring. Its goal is to provide and organize situational knowledge by acquiring data from sensors in the environment. Some examples of such sensors are those that monitor air pollution, aerosols, cloudiness etc. Applications that use this framework, written in Java, observe the data acquired from certain sensors on specific location and for a specific time period. Samurai [12] is an expandable, event-based architecture which integrates and demonstrates well-known software modules for event processing (Esper), machine learning (Weka) and knowledge representation (Parliament), as RESTfull services. It acquires, aggregates, represents and distributes contextual information. Its main characteristics include:

- Conversion of low-level data and events into characteristics which are easier to analyze and compare.
- Aggregation of data from various sources in order to increase accuracy of deduced conclusions.
- Identification of frequent activities in event flow in order to produce patterns that could be of interest.

WildCat [13] is a generic framework for the development of context-aware applications, which is based on sensor-based monitoring and data aggregation. Data are presented hierarchically, as in Unix file system, and are updated dynamically. It acquires, models, represents, understands and delivers context. Data browsing is performed synchronously and asynchronously, using pull and push methods. Esper is used for data processing, more specifically CEP (Complex event processing) engine and language similar to SQL in order to create queries over data. CAMF [14], or context-aware framework for machine learning developed for Android devices, combines machine learning and context-aware computing to provide services based on patterns of mobile devices usage coupled with the context surrounding a specific user. Components that this framework is comprised of are used for acquiring, modeling, representing, understanding and interpretation of the context. CAMF is developed so to hide physical sensors in order to ensure expandability and re-usability, as well as the possibility for application to work offline. Framework consists of three major levels. The first level provides contextual information, the second one records data and the third level provides Weka machine learning used over contextual information. CAreDroid [19] is a framework that is designed to decouple the application logic from the complex adaptation decisions in Android context-aware applications. CAreDroid, as part of the Android runtime system, monitors the context of the physical environment and intercepts calls to sensitive methods, activating only the blocks

of code that best fit the current physical context. CAreDroid allows applications developers to develop context-aware applications without having to deal directly with context monitoring and context adaptation in the application code. System architecture is comprised of three levels: context sensors that provide context information, system layer for monitoring and adaptation and application layer.

HARMONI middleware [20] includes a light-weight event engine that runs on the mobile device, and processes incoming sensor data streams using rules that are appropriate for the current context. Research suggests that appropriate context-aware filtering can significantly reduce the volume of transmitted medical data. There are four main components: Event Engine which processes the data readings, Data Adaptation component for converting sensors proprietary data formats into a standard sensor stream format, the Rule Manager and Rule Server on the backend interact with each other to dynamically alter the processing rules, Data Transmission component for transmitting compressed data to the backend infrastructure and Local Action Manager component responsible for manipulating settings on the local device.

CenceMe [21] middleware inferres physical and social context and shares information through social network applications. Social context detected locally on the device is transferred to a backend server to match common shared social contexts to raise social awareness. Classification can be done on the phone with the support of the backend servers, or entirely on the phone. Empath (Emotional Monitoring for PATHology) [22] is a middleware to remotely monitor emotional health for depressive illness. Patients' diagnosis and therapeutic treatment planning were supported by reports generated by aggregating context such as sleep, weight, activities of daily living, and speech prosody. The behavior analysis routines run on the server and results would be displayed on the touch screen fixed station at patients' homes. SystemSens [23] middleware captures usage context of mobile phones. Usage context is the collection of users' interactions with research applications. The users' interactions include battery, call, CPU usage, cell location, data connection active and traffic and telephony information events.

Most of the related researches, except [8], are implemented as a framework for Android smart devices. Common to all platforms is the general architecture of context-aware framework where the first level provides contextual information (sensing), the second records data (storing) and the third level usually is the application that reacts to a specific, deducted user's situation or provides Weka machine learning used over contextual information (processing and reasoning). Also common is the usage of built-in sensors, while in [11] data is acquired form sensors in the environment. Then usually sensors send the acquired data to the service, which further reasons the received data and notifies the application if needed. Unlike most of the listed works, context-aware systems/frameworks presented in [8-10] use local memory for storing acquired data. In order to process data in [12] and [13] integration and demonstration of software modules for event processing (Esper) is featured. [12] and [14] propose combination of machine learning and context-aware computing in order to provide services.

In the most of the proposed solutions mobile devices serve only as event sources and event filters, while the complete event processing and machine learning is done on a backend server. They don't include advanced processing and analysis of contextual data, nor generating a higher level of context and situation. The built-in sensors of smart devices provide applications with a continuous stream of context data. Event processing rules are used to aggregate and correlate the sensor data to more abstract and more meaningful situation data. Until a few years ago mobile operating systems were rather inefficient and the computing resources of mobile devices were very limited. Along with the rise of computing power of mobile devices the CEP for processing data streams is developed for mobile devices. The execution of sophisticated event processing rules directly on the mobile device is still a rather new approach. Also performing the event handling of embedded built-in sensors with CEP on the mobile device itself can be considered as a rather new approach. Complete event processing and machine learning should be server-independent and could be processed locally. Our solution showed that collected sensor data should be processed and stored locally on device, and notifications for the application U/I or data transmissions to the remote server are only necessary for events considered as warning or dangerous health conditions.

III. EGOSENSE – A FRAMEWORK FOR CONTEXT PROCESSING AND ANALYSIS

To support the development of mobile context-aware applications we developed a framework for context-aware services and implemented a demo application with impact on health care domain. Developed framework integrates components for: collecting data from sensors, complex events processing, reasoning and storing important data on resource limited devices. Events considered as important are presented to the user as notification of dangerous health condition and sent to the remote server for further processing and presentation. Evaluation and demonstration of the proposed framework is done by the development of a demo Android application for mHeatlh. The framework is designed in a way that the layers are totally independent and customizable depending on user needs. Applications of services in this area are highly interesting now days and particularly relevant for health care applications. Actions taken by the highest level in the architecture in a safe and reliable way are informing the user and other interested parties about the alarming situation related to the user's health condition.

Figure 2 presents the organization and the architecture of the system. The architecture consists of four basic layers, service executed in the background and *U/I* component for adjustment of variable parameters of context-aware service. The architecture of the system is comprised of three major, mutually connected subsystems: sensing, thinking and acting. Sensing subsystem consists of sensors that are being used, situations that need to be recognized and information that is going to be acquired. After that, techniques for context understanding are being realized. Finally, appropriate actions

are defined according to a specific need. Sensors at the bottom of the architecture acquire data which are then sent to the layer for data retrieval and preprocessing. Layer for context understanding as a result produces contextual information as needed by the application and services. Management layer can further perform queries over retrieved context information and distribute updated values to the background service and application.

The first level in the architecture retrieves data from physical sensors via sensor provider. By using Funf framework [15], application gets the information of user's location. Accelerometer and Google Play services send the notification of physical activity and the position of the user. Motion detector provides information that is used to calculate speed of a user's motions, and sensors provide information about the temperature, pulse (heart rate) and cardiac pressure of a user. Sensors data are retrieved over a certain period of time or after the change of values has been detected. Furthermore, it is possible to ask for the current value from a specific sensor using the implemented methods, if there is a need for it. All the data are deserialized in DTO, i.e. objects for data transfer, and are propagated to the next level that gathers them and prepares them for further processing. The next level for acquiring and analysis of contextual data from sensors, accepts retrieved data and classifies them in previously defined objects for storing values that are used for analysis of specified contexts. One object stores contextual data from specified sensors and every time one sensor retrieves and pushes a new value a specified object updates its previously stored values and raises event for the next level in order to notify it that the change in sensor values has occurred.

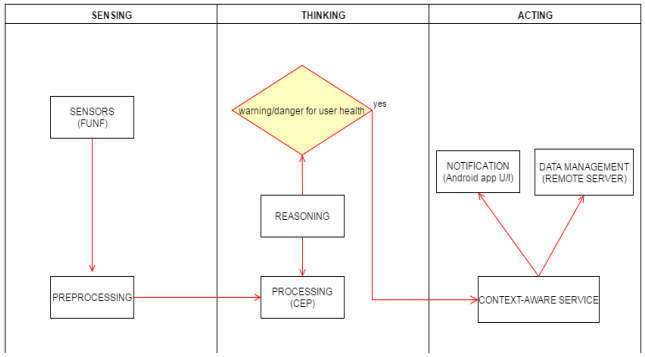

Fig. 1. System Architecture.

CEP (Complex Event Processing) is a software technology that provide processing of continuous streams of data in near real-time. When applied on Android smart phones, CEP use phone sensors and additional contextual information available on smart phones in real time. CEP is used for processing generated data flow from sensors and detected events. This level, after receiving notification and detecting the event, initiates appropriate actions based on received information by using Asper. Asper detects every change on sensors, generates a new event, executes a query on event flow which should check whether any of the conditions are fulfilled and notifies the next level every time when one of the above mentioned user cases is realized.

The highest level (monitoring level), generates a appropriate message if a condition for warning or alarm is fulfilled. It further propagates the message to the background service responsible for sending a notification to a user, sending notification via web service and sending a text message to all interested parties. Service is automatically run after an Android device is turned on and, for variable parameters in cited user cases, sets predefined values stored in device's memory. If there is a need to change parameters using the user interface component, the user can add desirable values and the changes will again be stored in devices memory.

IV. IMPLEMENTATION OF EGOSENSE FRAMEWORK

All the values acquired from sensors are joined with the location of the user and current time. In order to make this data readable to the end user and usable by mobile applications, it was necessary to develop a context management framework for acquiring, processing and analyzing these data so that their functionality can be used by mobile applications via the developed API. This framework was developed with the aim to provide context information of a higher level, that would enrich user's location with his/her situation and circumstances. It was shown how to process and analyze the data acquired from various physical sensors in order to generate the context, situation, behavior and activity of a higher level user, which could allow services to send contextual information to the interested parties. A demo Android application was developed using the framework that represents Personal Health Monitoring, and it should detect urgent or alarming health condition of a mobile user and initiate appropriate actions. Android application is based on Funf, Asper and Google Play Services libraries that have been included in this Android project in order to integrate their functionalities.

For gathering sensors data and user's activities which make up the context, Funf - Open Sensing Framework has been used together with sensors for motion, activity, pulse, pressure and temperature detection. For context processing, interpretation and understanding, CEP (Asper as Android port of Esper) has been used. The appropriate techniques and components for user context detection and processing on Android smart devices are implemented, all of which is brought together in a context-aware mobile service based on previously developed components. The following use cases have been selected and if some of these actions are detected, notifications to a user, notifications via web services and text message of a warning to other interested parties are activated:

- The user is still (he/she lies), pulse is higher than P, blood pressure is higher than K, and cardiac pressure is higher than S -> Warning!

  rule : "warning"
  select warningEvent from pattern[every warningEvent = WarningEvent (UserIsNotWalking(isWalking), PulseIsHigherThanNormal(pulse, P), SystolicPresureIsHigherThanNormal(systolicPresure, K),

DiastolicPresureIsHigherThanNormal(diastolicPresure, S))]

- The user is moving, the speed is higher than S, temperature of the environment (body) is above the limit T, and pulse is increased by 30% in the last minute. User's current location is in coordinates: X, Y -> User is in danger!

rule : "danger"
select dangerEvent from pattern[every condition = DangerEvent -> (timer:interval(60 sec) and dangerEvent = DangerEvent(PulseIncreasedBy30Percent(condition.pulse), !UserIsNotWalking(isWalking), BodyTemperatureIsHigherThanNormal(temperature, T), MovingSpeedIsHigherThen(speed, S), CurentLocationIn(location, X, Y)))]

In the demo application we developed a set of methods for organizing and preserving context data gathered from sensors. All processing happens in a background service which is invisible to the user. For example, every time user's blood pressure changes that sensor data is being transferred to the application layer that deals with gathering and context data organization. If that change has an influence on processed use cases then client's Android application generates high-level context in order to inform user if alarming condition has occurred. In case of detecting urgent health condition actions like sending text messages to the interested parties (phone numbers are preconfigured in the application for every user), displaying notification on user's smartphone and sending notification via web service to web server occur one after the other. Interested parties in those notifications can see explanations, with exact time and date, about the problem originated in a given moment regarding the state of health of the user and current monitored health parameters which led to that health condition. Restful Web service is developed in order to be responsible for sending the notifications related to detected events in Android application to the Web server. Web application is developed in order to store alarming health conditions in SQL database for every user. Also Web application is used for visualization and analysis of critical events detected by the user's smart phone. With simple architecture containing application layer (service layer) and data access layer, MVC Web API on server side receives client's notifications in form of HTTP Post requests. Received data are used later for generating all kind of reports for every user (for example how much disturbing health changes the user had during every night in this month). In addition the web server also sends email messages to preconfigured addresses (email addresses are stored in a database for every user separately) for every user when notification of his alarming health condition has been received.

The Android application is tested by several test users on real mobile devices which run Android operating system with API level 2.3.5 (Gingerbread) and above. Server application was hosted on Microsoft Azure cloud server and database was hosted on SQL Server 2012. The system was active and reliable 24/7. With an active internet connection, the application was able to send notifications to the web server and, of course, to send text messages and display notifications to the user. When there was no internet connection text messages were sent to the interested parties and notifications were displayed on the user's smartphone. From the database hosted on Web server reports were generated for specific user for specified date and time intervals in order to monitor disturbing changes in their health conditions.

V. EVALUATION

The client application has been developed with the Android application framework that provides access to the local device resources like hardware sensors of the device and processing the data by using CEP. So if the screen consumption is taken into consideration, and having in mind that the core of Android operating system consumes 34.9% of processor resources, it can be said that by testing developed context-aware service on four different smartphone devices which run different versions of Android OS we concluded that implemented activity recognition system does not influence consumption and processor usage in any significant way. It is proved that the presented architecture of context-aware service, implemented through a demo application for the Android operating system, could be implemented, without any difficulty, for any other operating system running on smartphones by using similar open source technologies specialized for use on the selected operating system. The proposed architecture offers advantages like reduced network traffic and exploiting local processing power. To be more specific, sensor data is processed directly on the mobile device. Cloud server is used for preserving logs related to alarming health conditions. Because the sensors of potentially many users may produce high volumes of data, the overall network traffic is reduced significantly. Also processing data on the smart devices exploits only the processing power of mobile devices.

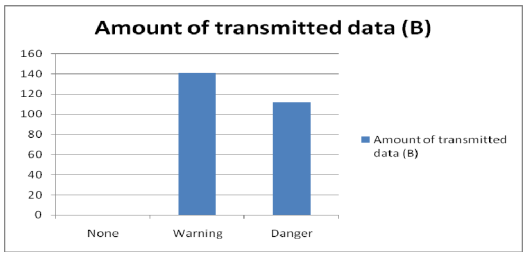

Fig. 2. Amount of transmitted data to the remote server

From Figure 2, we see that appropriate filtering of sensor data results in a significant reduction in the amount of data transmitted to the remote (cloud) server. Thus, we conclude that context-aware events filtering can lead to a significant reduction in the amount of data that must be transmitted by a mobile device. In many sensor environments, a large fraction of the collected data conforms to expected patterns, and data transmissions are only necessary for notification about the warning or danger events for user's health condition. That behavior can allow locally performing an accurate and continuous analysis of the users' health status by minimizing network transmission and maintaining appropriate levels of security and privacy by using HTTPS protocol for communication with the cloud server.

CPU utilization regarding developed context-aware service seems to be appropriate and it doesn't have a significant influence on the performance of smart devices because events processing and reasoning use very small fragments of available resources. Also, we observed the memory usage of the reasoning system on the four mobile devices considered. As further consideration, it is possible to note that memory usage does not significantly vary depending on the mobile device used. As the number of collected sensor data increases by time, the overall memory usage of application grows up which increase the pattern matching time and the overall reasoning time.

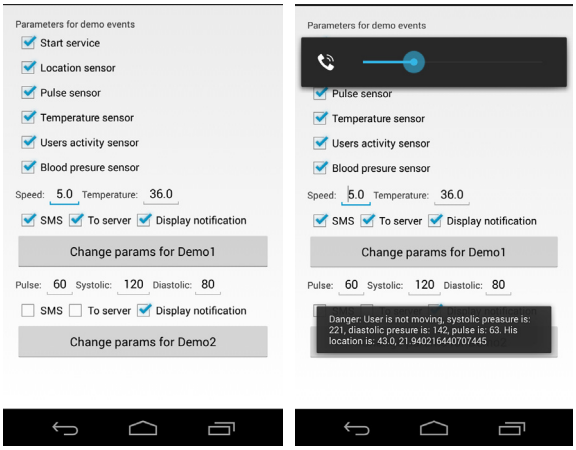

Fig. 3. Personal Health Monitoring Application screen shots.

In the health care domain, the presented platform proved to be secure and reliable, because it operates in the background and it is very reliable from the standpoint of the values obtained with built-in sensors. The technologies used to handle sensor data are free and open source, so they can be adapted for handling users' specific requests. The user does not need to interact with the service or to have any technical knowledge, because the service will perform all the work and inform him of the alarming situation regarding his health condition.

VI. CONCLUSION

Possibilities of applying context-aware systems are perhaps the greatest in the field of health care, where we can see an increasing interest in the systems that can detect human activities. The methods for activity tracking currently used, often require a lot of time and resources, and are usually in the form of an additionally paid person who is in charge of monitoring the patient and reporting his/her condition or in the form of making a patient independently notify a doctor about his/her condition in prescribed time intervals. An automatic activity recognition system would lower the risk of mistakes and, in addition, it would increase the quality of health care, since it would save the time that medical staff usually spends in acquiring and sorting the data. Furthermore, non-invasive monitoring will enable people to live their everyday lives undisturbed while, at the same time, they would offer their doctors a more precise presentation of their activities. Another possibility of application is social networks which are becoming an important part of everyday life. Existing communication services allow a simple exchange of textual messages, photos, videos, etc. With the use of sensors, a richer user context could be shared with friends in a much more natural and easier way. Automatic activity recognition would enable users to share their current physical activity with their friends via social networks. The results of this research work show that it is possible to develop mobile context-aware applications which would adapt to a user using his/her current context and which would, in accordance with the general context, access services in order to acquire or deliver adjusted information and services.

REFERENCES

[1] A. K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, Vol. 5, No. 1, pp. 4-7, 2001

[2] S. Loke, Context-Aware Pervasive Systems: Architectures for a New Breed of Applications. Auerbach (CRC Press), 2006

[3] M. Baldauf, S. Dustdar, F. Rosenberg, "A Survey On Context-Aware Systems", International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, pp. 263-277, 2007

[4] T. Gua, H. K. Pung, D. Q. Zhang, "A service-oriented middleware for building context-aware services", Journal of Network and Computer Applications, Vol. 28, pp. 1–18, 2005

[5] J. Koolwaaij, A. Tarlano, M. Luther, P. Nurmi, B. Mrohs, A. Battestini, R. Vaidya, "Context Watcher ─ Sharing context information in everyday life", IASTED Conference on Web Technologies, Applications and Services (WTAS), 2007

[6] B. van Wissen, N. Palmer, R. Kemp, T. Kielmann, H. Bal, ContextDroid: an Expression-Based Context Framework for Android, PhoneSense 2010, Zurich, Switzerland, 2010

[7] T. Hasu, ContextLogger2- A Tool for Smartphone Data Gathering, HIIT Technical Reports 2010-1, Aalto University, Finland, 2010

[8] M. R. H. Toivonen, R. P. A. Oulasvirta, "ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications", Pervasive Computing, Vol. 4, No. 2, pp. 51-59, 2008

[9] AWARE Android Mobile Context Instrumentation Framework, 2014, http://www.awareframework.com/

[10] J. Dunkel, R. Bruns, S. Stipković, "Event-Based Smartphone Sensor Processing for Ambient Assisted Living", 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), Hannover, Germany, March 6-8 , 2014

[11] M. Stocker, M. Ronkko, M. Kolehmainen, "Abstractions from Sensor Data with Complex Event Processing and Machine Learning", 7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA, 2014

[12] D. Preuveneers, Y. Berbers, "SAMURAI: A Streaming Multi-tenant Context-Management Architecture for Intelligent and Scalable Internet of Things Applications", International Conference on Intelligent Environments, Shanghai, 2014

[13] Wildcat, http://wildcat.ow2.org/

[14] A. I. Wang, Q. K. Ahmad, Camf – Context-Aware Machine Learning Framework For Android, Institutt For Datateknikk Og Informasjonsvitenskap, 2010

[15] Funf open sensing framewok, http://www.funf.org/about.html

[16] Android Activity recognition, https://tsicilian.wordpress.com/2013/09/23/android-activity-recognition/

[17] Low-power sensors, https://developer.android.com/about/versions/kitkat.html

[18] Event Processing with Esper and NEsper, http://esper.codehaus.org/tutorials/tutorial/tutorial.html

[19] S. Elmalaki, L. Wanner, M. Srivastava, "CAreDroid: AdaptationFramework for Android Context-Aware Applications", MobiCom '15 The 21st Annual International Conference on Mobile Computing and Networking, 2015