A CONTEXT-ORIENTED SYSTEM FOR MOBILE DEVICES

Ismael Cuadrado Cordero[1], Luis Miguel Soria Morillo[1], Juan Antonio Ortega Ramírez[1] and Luis González-Abril[2]

1: Dept. of Computer Languages and Systems of University of Seville, 2: Dept. of Applied Economic I of University of Seville

ABSTRACT

Our work provides developers of context-based application for mobile devices a framework for developing comprehensive and adaptable solutions that can interacts each other through a set of functionalities and interaction methods for building secure applications easily. On the other hand, our work provides a platform to reduce the energy consumption of the devices; due to the reuse of functionalities. To do this, we have designed a layered architecture that allows interaction between applications and context-oriented services transparently to users. The main layer of our architecture (Core layer) provides a tool that allows communication between adjacent layers. Moreover, using our architecture, developers can design context-based applications in a simpler way, a very important goal in order to increasing number and functionalities of this kind of applications on mobile devices. Furthermore, our architecture allows the reuse of knowledge between developers. During our work, OSGi technology has been used in mobile phones, cutting-edge research in the field.

I. INTRODUCCIÓN

Everyday, we see that the development of mobile device technology has been increased with the aim of improving the user experience. Thus, all progress of these technologies is focused on offering more and more features that are, in most cases, underused. This is because most users do not have the experience and skills necessary to improve the use of the system. For this reason, there is a large segment of population without access to these new technologies. Is in that point when software should offer users the ability to improve the use of those advances. The goal of software technologies for mobile devices must be not interactive - highly functional systems, so user obtains more functionalities with little interaction as possible.

Today's users are looking for a system that can be used in their daily lives without an active control. That's the main reason for the emergence of context-oriented applications. This kind of application is booming and there are a lot of developers specialize in these. The disadvantage of these applications lies that their development is too complex for the average developer. That means that only a small set of applications makes good use of resources. In addition, a user wishing to use several context-based applications should install all these different applications on the same device. That affects the user experience, which should set different front-ends for different applications.

Our architecture is designed with the goal of solving these problems. It can improve system performance and simplifies development. Furthermore, the use of architecture offers great potential for future applications. This is because an application developer can reuse the code generated by another developer, with its layered architecture. For these reasons, we believe that our architecture is important for research in the field of context-orientes systems on mobile devices.

II. RELATED WORK

There are context-oriented systems that use property hardware ([1], [2] and [5]) and those that use general purpose hardware ([3], [4] and [6]). We will focus on the second kind of devices, due to its versatility and price. In the other hand, general purpose systems have the handicap of limitations on data recovery. Our software must wrestle with this handicap.

History of the context-oriented systems begins at year 1991. On this year, Mark Weisser used on the first time the word 'pervasive', meaning the integration of a device in the daily life of users. Following the above definition, Want, Hopper et Al., describe context-oriented systems like "a strong part into 'pervasive' systems". At the same time, they present their application for location 'Active Badge Location System' [18]. This is considered the first context-oriented application. Middle in the '90s, operating the boom of context-oriented systems, thanks to applications like [12], [16] and [17].

The most extended classification of these systems is what Chen [7] made:

- **Direct Access to Sensors.**
- **Middleware based:** Layered architecture. Generally, this typology only handles a kind of context.
- **Context server:** Can handle more than a context, reusing generated data and code.

Independent of the architecture, it is necessary to define an ontology. This is important for applications can access to data context. According to Korpipää and Mäntyjärvi [20], most important goals on an ontology definition must follow four dictates: simplicity, flexibility and extensibility, generic and expressive. Most used ontologies are RFD (used in [8], [9] and [10]) and OWL (used in [11], [13], [14] and [15]).

III. OSGI

Layers communication is possible thanks to OSGi technology. This technology is far used in lots of different fields. However, OSGi has not been regularly used in mobile devices. Integration of this technology on this field is a bet on modernity and future.

The first problem of this technology on mobile devices is because the architecture of these devices is radically different to PCs. Java virtual machine in PCs is common to all applications, that is, there is only one virtual machine. However, smartphones virtual machine behavior is not the same. In this case, each application is running on a different virtual machine, so called Dalvik for Android operating system. Thus, due to OSGi communication protocol needs to share messages between applications, is required that all application run in the same environment. To solve this drawback, there are three alternatives:

- *OSGi Alliance standard.* In this case, we must solve the communication problem by using logs file interconnection.
- *OSGi alternative protocols.* There are several parallel protocols to the original created by *OSGi Alliance*. The problem of these protocols is that haven't a common architecture and functionality.
- *OSGi proprietary implementations.* Some enterprises have developed OSGi implementations, creating its specific environments. The main problem of these solutions is that applications must be run over a proprietary platform and not directly over the JVM.

We was choice *Apache Felix*, an OSGi implementation that can be modified for its use on mobile devices.

IV. ARCHITECTURE

We have defined a layered architecture based in a context server. The mainly purpose of the architecture is to reuse data provide by embedded sensors in smartphones, and avoid the computational cost derived from the processing of these data. It means that developers can build applications more complex in an easier way and, of course, reduce the computational cost and hence, the energy expenditure associated. Using this architecture we expect that more applications can use the device, improving its behavior, by using all its resources. On this way, we have defined the following layered architecture:

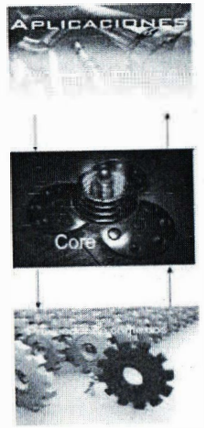

Figure 1: General architecture structure

V. CONTEXT PROVIDER

The context providers' layer is designed to store all objects which goal is to recover all context data provides by the device. When a developer wants to determine a context, firstly it must install a 'context provider' which will recover all context information data. When a 'context provider' determines a new context, it will send to the next layer. This communication between layers is established using an Ontology defined for the purpose.

An example for a context provider is an object that sniffs the location sensors looking for the current location. This can be the definition of a location context. This hypothetical object is developed to compute proximity to a specific location and determines a context based in the distance to the location (High, medium or low).

The development of context providers must follow an established structure. This structure is defined in order to improving the operation of the system.

VI. CORE

On this architecture, the harder computing is processed in the core layer. This layer is in charge of data processing and store. When a context provider sends new data to the core layer, it applies a set of preprocessing steps. Then, it stores those data. At the end, it processes all data and sends a message to the application layer if necessary.

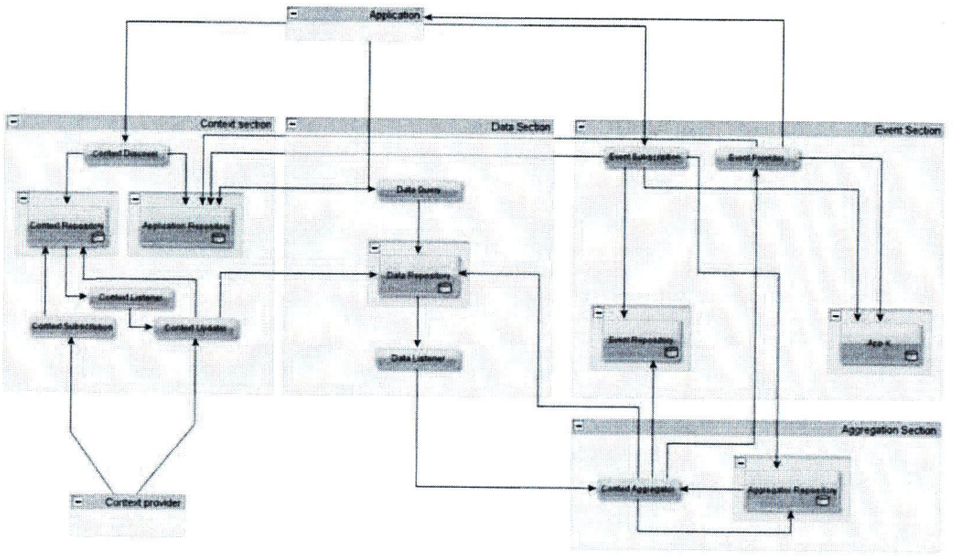

Figure 2: Extended definition of Core layer architecture

On this way, the core layer is divided into four sections.

- *Context Section:* It provides to the system all the functionality directly related to the context. It will store and process context information. It also provides access to the context information for the application layer.
- *Data Section:* This section provides the system usability for data objects. A data object is an object that follows a defined structure. It stores data objects and makes them accessible for the next layer.
- *Event Section:* In this section are stored objects called 'events'. These objects are designed to store simple context information. The goal of 'event' objects is to define the condition when a context is important for the user. An event is defined in a vector with three components: Context, Condition and Value. An example for a valid event is (Speed, >, 120).
- *Aggregation Section:* An aggregate is a set of events. This object is designed to reuse the events stored in the system. An aggregate is active when all its events are active. This is because there are so many applications that use the same event.

VI.1 Context Section

The context section is designed to manage context information. When a Context provider detects a new context, it will be send to context section. The context section picks the new context across the 'context update' component. This component is in charge to check if a context provider is registered in the system. If a context provider is registered in the system, it will be added to the 'context repository' component. This is because of the security, and the component in charge of registry is 'context subscription'.

The 'Application repository' is the component in charge to store the relation between applications and context. This is also because of the security of the system. At last, there is the 'context discover' component, which acts like an access point for the application layer to this section.

VI.2 Data Section

The two main components in data section are 'Data repository' and 'Data query'. The first component is a repository, used for the storage of historical data. The second component acts like an access point. This component allows applications to access the context providers' generated data.

VI.3 Event Section

Using this section, core layer can manage all events registered at the system. Like all sections, it has a repository used for storage. This repository is the component 'Event repository'. At this layer, it can be found all the registered events. The relation between events and applications is stored at the component 'App X'. This component is used for optimization and security of the layer. At the end, there are two components used for the interaction between applications and core.

VI.4 Aggregation Section

This section is designed for reuse of the stored events. It is composed of two components. The first component is called *Aggregate Repository*, which stores aggregates information, for instance, events that is composed by, logical functions that relate these events and last value of aggregate evaluation. The second component is *Context Aggregate*. This component stores all the related logic for the aggregate. With this component it can be made sniffing of new data and detect new changes on stored aggregates.

VII. Applications

The application layer is the layer in charge of giving functionality to the system. On this layer there are stored all objects designed to give the user a service. These applications must follow a predefined structure. Applications do not need to know anything about context providers (use, name, etc.). An application just has to registry under the system looking for a service. Then, the core layer will search for the appropriate context provider. If this context provider "dead", the core layer will look for another context provider that can make the same work.

Development of applications is so easier now. Developers just have to design user functionality and registry at the system.

VIII. Conclusion

The main problems of context-oriented applications are complexity of development and reutilization of resources. On one hand, our architecture improves the usability of device. This is because of reuse of generated data made at core layer. Using this middle layer, different applications can use same data on different purposes. Due to this improvement, we can remove need of different applications using same resource. So, we can optimize use of the device.

On the other hand, our architecture simplifies development of applications. This is because of separation between context providers and applications. This means lots of new developers making applications. Thanks to this new population, use of the device will be increased. Our expectations with this architecture are to provide the developers population a new tool that increased use of context-oriented applications.

At last, using OSGi technology in a mobile device increased functionality and optimize messaging. Use of this technology allows system dynamically install new applications or context providers. This increased functionality of system and make it more attractive for final users. Users will see a dynamic system which does not need to reboot when actualizing.

References

[1] Bill N. Schilit and Marvin M. Theimer. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5). 1994

[2] Nick Ryan, Jason Pascoe and David Morse. Enhanced reality fieldwork: the contextaware archaeological assistant. Gaffney, V., van Leusen, M., Exxon, S. (eds) Computer Applications in Archaeology 1997

[3] Anind K. Dey. Context-aware computing: The CyberDesk project. Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments. Menlo Park, CA: AAAI Press.

[4] Richard Hull, Philip Neaves, James Bedford-Roberts. Towards situated computing. In Proceedings of International Symposium on Wearable Computers 1997

[5] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann. Context-Awareness on Mobile Devices – the Hydrogen Approach. 2002

[6] Jay Budzik, Kristian J. Hammond. User interactions with everyday applications as context for just-in-time information access. Proceedings of Intelligent User Interfaces 2000. ACM Press, 2000

[7] Harry Chen. An Intelligent Broker Architecture for Context-Aware Systems. PhD. Dissertation proposal. 2003

[8] Panu Korpipää, Jani Mäntyjärvi, Juha Kela, Heikki Keränen, Esko-Juhani Malm. Managing Context Information in Mobile Devices. IEEE Pervasive Computing. 2003

[9] Panu Korpipää, Jani Mäntyjärvi. An Ontology for Mobile Device Sensor-Based Context Awareness. Proc. Context '03, LNAI no. 2680, 2003

[10] Resource Description Framework (RDF). www.w3.org/RDF/

[11] OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/

[12] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. Wireless Networks, 3(5). 1997

[13] Tao Gu, Xiao Hang Wang, Hung Keng Pung, Da Qing Zhang. A Middleware for Context-Aware Mobile Services. IEEE Vehicular Technology Conference. Milan, Italy, 2004

[14] Harry Chen, Tim Finin, Anupam Joshi. Using OWL in a Pervasive Computing Broker. Workshop on Ontologies in Agent Systems, AAMAS 2003

[15] Harry Chen, Tim Finin, Anupam Joshi. An Ontology for Context-Aware Pervasive Computing Environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review. 2004

[16] Sidney Fels, Yasuyuki Sumi, Tameyuki Etani, Nicolas Simonet, Kaoru Kobayashi, Kenji Mase. Building a context-aware mobile assistant for exhibition tours. The First Kyoto Meeting on Social Interaction and Communityware. 1998

[17] Nigel Davies, Keith Cheverst, Keith Mitchell, Alon Efrat. Developing a context sensitive tour guide. Proceedings of First Workshop on Human-Computer Interaction for Mobile Devices, Glasgow, UK. 1998

[18] Roy Want, Andy Hopper, Veronica Falcão, Jonathan Gibbons. The Active Badge Location System. ACM Transactions on Information Systems, 10(1), 1992