EXPLOITING CONTEXT IN MOBILE APPLICATIONS

INTRODUCTION

Pervasive computing is nowadays becoming a reality, exploiting the capabilities offered by both computing infrastructure and communication facilities. The pervasive computing environment encompasses a multitude of diverse devices, operating systems, protocols and standards. It includes mobile devices such as cellular phones, smart phones, PDAs and handheld computers for information access, smart cards and smart labels for identification and authentication, smart sensors and actuators that perceive the surroundings and react accordingly. Voice technologies such as Automatic Speech Recognition (ASR), Text to Speech (TTS) and VoiceXML enable the construction of convenient user interfaces and Web Services are a key mechanism for interoperability. Wireless Wide Area Networking allows long distance communication through cellular radio while Wireless Local and Personal Area Networking and standards such as the Wi-Fi, Bluetooth and IrDA allow short distance communication through radio waves and infrared beams.

In the mobile and pervasive computing environment, software engineering should not treat diversity and mobility as problems to overcome, but seek methods of which it could take advantage instead. In these environments, the selection of purpose-oriented and timely information, tailored to user preferences and media characteristics will ensure an optimised information delivery. To this end, the context –the information that surrounds the human-computer interaction– plays a key role and is rapidly changing in mobile settings, and the understanding of it is indispensable for the application designers in order to choose, capture and exploit it. The importance of the context is to use it to make context-aware applications, i.e. those applications that are interested in who, where, when and what, in order to determine why the situation occurs and adapt their behavior accordingly.

BACKGROUND

The Concept of Context

An important dimension of mobile computing is “mobility”, which is primarily concerned about people moving in space and doing their personal, social and professional activities in a wide temporal space. The informative support of the mobile user can be accomplished through terminals, which are movable and operate regardless of the location and time and offer wireless access to information and services. Although mobility –spatial and temporal- is an important aspect of mobile computing, it constitutes only one dimension of the “context”.

The term “context” is defined as “the interrelated conditions in which something exists or occurs” in Merriam-Webster’s dictionary. In the domain of context-aware computing, researchers have defined context as “location, identities of nearby people and objects, and changes to these objects” (Schilit & Theimer, 1994), “location, identities of the people around the user, the time of day, season, temperature, etc” (Brown, Bovey, & Chen, 1997) and “knowledge about the user’s and the device’s state, including surroundings, situation and location” (Schmidt & Laerhoven, 2001). Dey and Abowd (1999) propose a more generic definition according to which context is any information that can be used to characterize the situation of an entity. An entity is a person or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. Dey’s definition is more comprehensive and generic and makes it easier for an application designer to enumerate the context for a given application and choose the appropriate desirables.

The contextual information can be classified, according to which entity it concerns, into the following categories (Schilit, Adams, & Want, 1994):

- *User context:* user identity, location, collection of nearby people, social situation, activity, user's profile, etc.
- *Computing context:* hardware characteristics, software characteristics, network connectivity, communication bandwidth, nearby resources such as printers, displays and other devices and so forth.
- *Physical Context*: lighting, noise level, temperature, humidity etc.
- *Time Context:* time of the day, week, month, season of year, time zone etc.

Orthogonally to the above classification, context can be divided into two broad classes: *primary* and *secondary context.* Primary context derives directly from sensors or information sources while secondary context is inferred from the primary context. For example the name of a city is a secondary context because it derives from GPS coordinates through a relation mechanism.

Context can be also distinguished according to a range of temporal characteristics and it can be classified as *static* or *dynamic*. Static context does not change very quickly (or at all; e.g. a person's date of birth), while dynamic context does (e.g. the location of a person who is driving). When applications are not only interested in the current state of the context (*present context*), but past context is of importance too, context histories are stored; in some situations there is a necessity for context prediction (*future context*).

The interactions between context sources and context sinks can be characterized as "context push", when the context sources update periodically context information in context sinks and "context pull", when the sinks demand information from the context sources.

The types of context, according to the manner of its acquisition, can be divided in three categories:

- *Sensed context:* this context is acquired from the environment by means of physical or software sensors (identity, temperature, time)
- *Derived context:* this kind of contextual information is computed (for example the name of the city from GPS coordinates).
- *Context explicitly provided:* the context that the user provides explicitly (for example the entries in the user profile).

Defining Context-Aware Applications

Context-awareness is a concept that consists of two notions: the notion of perceiving the *context* and the adaptivity that derives from the awareness. Adaptivity or adaptability is defined as the ability of a service/application to react to its environment and change its behavior according to the context. Context-aware computing was first introduced by Schilit and Theimer (1994) as the use of software that adapts according to the location of use, the collection of nearby people and objects, as well as to changes to such elements over time. Fickas, Korteum and Segall (1997) define context-aware applications (called environment-directed) as applications that monitor changes in the environment and adapt the operation according to predefined or user-defined guidelines. Dey and Abowd (1999) characterizes a system as context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

The functions that a context-aware application should implement (Schilit, et al., 1994) are:

- *Proximate selection,* a user interface-level technique where the nearby located objects are emphasized or otherwise made easier to choose.

- *Automatic contextual reconfiguration*, a process of adding new components, removing existing components, or altering the connections between components due to context changes.
- *Contextual information and command*, queries on contextual information can produce different results according to the context in which they are issued. Similarly, context can parameterize "contextual commands".
- *Context-triggered actions*, "If-then" rules used to specify how context-aware systems should be adapted.

Dey and Abowd (1999) propose that the features that context-aware applications may support are:

- *Presentation* of information and services to the user or use context to propose appropriate selections.
- *Automatic execution* of a service according to context changes.
- *Tagging of context* to information for later retrieval.

Related Work

The first context-aware system was the Active Badge System developed at Olivetti Research Lab. In case an employee on duty was not in his office, this would direct phone calls to the closest appliance according to the employees' location in the office environment. An evolution of this system is the ParcTab system, which was developed at the Xerox Palo Alto Research Center, relied on PDAs to offer a range of context-aware office applications (Schilit et al., 1994). Besides the above projects, a number of efforts have resulted in context-aware applications that can be categorized as follows (Dockhorn, 2003):

- **Conceptual Frameworks:** they focus on the architectural aspect of context-aware systems and introduce methods of gathering, interpreting and disseminating context to the interested parties. Examples of this approach are the projects: Context Toolkit by Georgia Institute of Technology (Dey, Abowd, & Salber, 2001) and Cooltown by Hewlett-Packard (Kindberg et al., 2002).

- **Service Platforms:** They allow the rapid creation, deployment and dynamic discovery of services, in order to provide the appropriate functionality according to user context. Examples of service platforms are the M3 architecture from the University of Queensland (Indulska, Loke, Ratotonirainy, Witana, & Zaslavsky, 2001) and the Platform for Adaptive Applications from the Lancaster University (Efstratiou, Cheverst, Davies, & Friday, 2001).
- **Appliance Environments:** they try to support interoperability among collections of appliances. Examples of such environments are the Ektara environment from MIT (DeVaul & Pentland, 2000) and the Universal Information Appliance from IBM (Eustice et al., 1999).
- **Computing Environments:** they provide context-aware computing that decouples users from devices and enables applications to perform tasks on behalf of the user. Projects of this category are the PIMA by IBM (Banavar et al., 2000) and Portolano by the University of Washington (Esler, Hightower, Anderson, & Borriello, 1999).

AN ARCHITECTURE FOR CONTEXT-AWARE MOBILE APPLICATIONS

The requirements

The dynamic environments, in which the mobile applications operate, constitute a challenge in order to propose a new software architecture that exploits the context and facilitates the design of applications which enhance the user’s mobile experience with suitable services.

The requirements that have to be met by such an application are:

- *Capture* contextual information from sensors and users.
- *Store* part of the contextual information for later exploitation.
- *Interpret* information, at an abstract level, in order for it to be more meaningful.
- *Transit* the information to the application modules.
- *Adapt* the application behavior according to the context.

The main goal of our approach is the adaptation process, which consists of two components: firstly the design of an architecture which supports adaptation at run-time and secondly, the provision of design of the core application itself in order to be context-aware. One evident question concerning the adaptation, is "which are the real adaptation tasks that a context-aware mobile application has to implement?" Placing the end-user in the middle of the concern, thus employing a user-centered approach (Vredenburg, Isensee, & Righi, 2001) the adaptation operation (Kappel, Retschitzegger, & Schwinger, 2001), should consist of the following:

- *content adaptation*: determine which content elements are hidden or revealed according to context changes.
- *operation adaptation*: the choosing of and switching between the functional components of a service/application according to the context.
- *presentation adaptation*: the tailoring of the system's user interface and interactive behavior to the individual needs of the user and system capabilities.

Another issue with regard to the adaptability that should be decided is "where the adaptation process to the context takes place". It could occur (i) within the application and it is *called laisser-faire adaptation*; (ii) exclusively outside the application, called *application transparent adaptation*; and (iii) within the application as well as out of it, called *application – aware adaptation* (Satyanarayanan & Ellis, 1996). Our approach adopts the application-transparent adaptation, since mixing context management code with the application code renders the application code more complex, difficult to maintain and impossible to reuse. This indicates the decoupling of context-independent activities of the application from the contextual concerns, as well as the separation of contextual data from the application’s data.

In the following paragraphs, we first present context modelling, which is a required underpinning for any context-aware application and subsequently present our proposed framework that satisfies the above requirements.

Context Modeling

One of the major issues in context-aware applications is the representation of context, in such a way that facilitates the context reasoning and sharing. Insofar a lot of context models have been proposed, some based on proprietary representation schemes and others on more formal data models.

The existing context models belong to one or more of the following categories:

- **Key-Value Models:** the simple key-value data structures, which were first introduced by Schilit, Theimer and Welch (1993) in PARC’s mobile computing environment. This representation provides a fast and easy way to set and update context but it is only feasible in situations where (key, value) pairs obtained from the environment exactly match those in the model; e.g. the pair (temperature, 50) matches itself only, while it does not match the pair (temperature, 50.1). Moreover, this model lacks any structure and is thus difficult to manage when the number of keys and/or values increases.
- **Web-Based Model:** in this context model, each entity (person, place or thing) corresponds to an unstructured web page, retrieved via a URL. The Cooltown project (Kindberg et al., 2002) relies on this model, which is intended to be used by humans rather than by applications.

- **Markup Schemes Models:** Using markup languages –such as XML- is another method to model contextual information in a flexible and structured manner. Profiles are a typical example of this kind of context modeling approach. For example, CC/PP (Composite Capabilities Preferences Profile) is an RDF-based framework (Resource Description Framework) to describe user preferences and devices' capabilities and characteristics. However the usage of profiles for context representation becomes difficult when the relationships and constraints of context are complex. The models under this category are extensions to the above standard, with no fixed hierarchy and try to cover the higher dynamics of contextual information.
- **Object-Oriented Models:** They are based on an object-oriented approach in which context information is structured around a set of entities. This modeling often uses a variation of the Entity Relationship model and stores the contextual information in relational databases. Although the object-oriented models employ the benefits of encapsulation and reusability, they are usually designed for a specific context domain.
- **Ontology-based Models:** Ontologies are believed to be the most suitable model for the representation and reasoning of context information, for the following reasons (Mokhtar, Fournier, Georgantas, & Issarny, 2005): (i) they enable knowledge sharing through a common set of concepts, (ii) they allow efficient reasoning so as to deduce high-level from low-level context and (iii) they enable interoperability.

The Context – Aware Architecture

The overall proposed system architecture is depicted in figure 1, while its operation and individual modules are discussed in the following paragraphs.

Context Manager

The *Context Manager* is the application module that is responsible for the capturing of the context from different sources, the interpretation of it in a higher level format, the storage of the context and the distribution of it to the *Adaptation Manager*.

The *Context Gatherer* subsystem is in charge of the capturing of context either from hardware sensors (e.g. location sensors, identification sensors, motion sensors etc.) or software sensors (processing power, available hardware components, storage systems, software components, current time etc.) or through the import of relevant profiles. These profiles could be:

- *User Profile*: It contains information about the user and his/her desired application's features. User-pertinent data could be his/her name, date of birth, address, occupation, hobbies, available technical equipment, billing plans, activities over time etc. Computer-relevant data could be application-independent, like font size, preferred language, content characteristics (e.g. text, audio or video), or application- dependent, like the kind of stocks for which the user wishes to be informed when price changes occur.

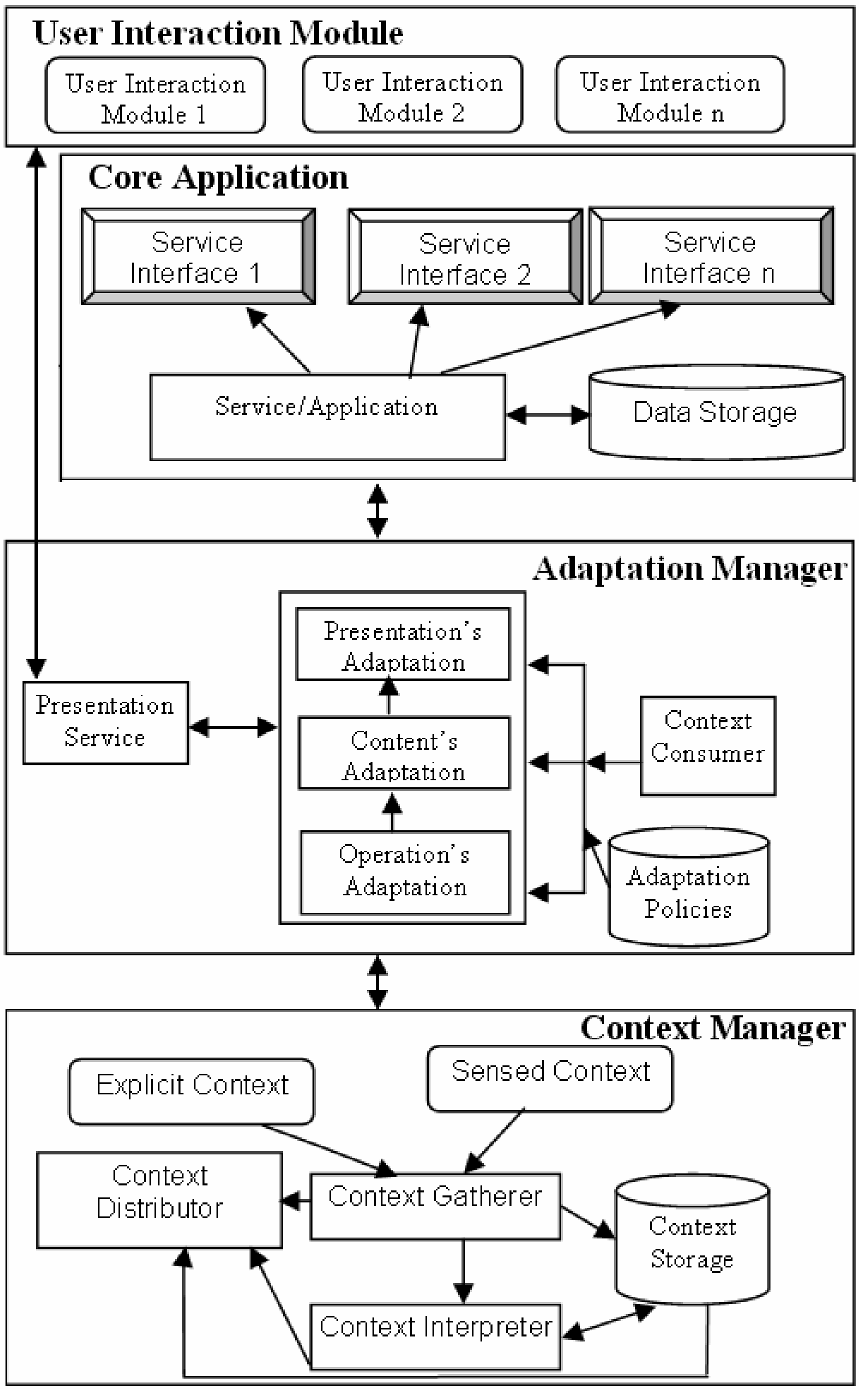

Figure 1 – System Architecture

- *User Agent Profile*: It describes the capabilities of devices e.g. display size, memory, operating system, browser characteristics etc.
- *Network Profile*: It describes network characteristics like supported bandwidths, bearer type and so forth.

- *Application Profile*: it holds application-specific information, which can be used for dynamic composition of modules from distinct services.

The *Context Interpreter* gathers information from *Context Gatherer* and *Context Storage* and conducts aggregation and inference activities. The resulting context is either passed to the *Context Distributor* or stored to the *Context Storage* for later retrieval.

The *Context Distributor* makes contextual information uniformly available to the *Adaptation Manager*. It provides information in two different modes: *request-response* and *event triggered*, supporting the "push" and "pull" character of the context. In *request-response* mode it grants context only on explicit request while in *event triggered* mode the provision of context is fired from specific events.

Core Application

The *Core Application* consists of a number of services that materialize the desired functionality. These services are not context-aware and their behavior remains the same in the different context situations. Furthermore these services expose their functionalities in a number of distinct and predefined alternatives called *service interfaces* (e.g. multimedia or plain text version) in order to support the diverse preferences during the operation adaptation process.

Adaptation Manager

The adaptation process occurs explicitly in the boundaries of the *Adaptation Manager* and could be accomplished through three concrete steps: the operation, the content and the presentation adaptation. The *Adaptation Policie*s repository contains the adaptation rules according to which the adaptation process takes place (e.g. *If* condition *x* is met *then* message *y* is triggered). The *Presentation Service* is the particular instance of a specific adaptation operation which is forwarded to the *User Interaction Module3*

The *Operation Adaptation Module* carries out the service's composition, which is the process of collecting the suitable application services in the desired version. These services are matched according to the adaptation rules, which depend on the context situation. The context data are passed as input to this process.

The *Content Adaptation Module* materializes the *content adaptation* that follows the *operation adaptation*. It makes the choice of the appropriate content, the language, the compression of data etc, in order to suit the client’s display characteristics, network capabilities and user preferences.

The *Presentation Adaptation Module* conducts the *presentation adaptation* that occurs after the *content adaptation.* It makes the transcoding and the modality transformation with a view to satisfying the clients’ claims.

The *Context Consumer* is the bearer of the context to the *Adaptation Manager*. Additionally the services subscribe to it, in order to be informed by the “push” notifications upon context changes.

User Interaction Module

The User Interaction Module presents data and selection choices to the end user in different modalities. The adaptive user interfaces have been produced as a result of the adaptation process, considering both individual needs and changing conditions in the application environment.

In many existing projects in the area of context-aware computing, capturing and implementation of context have been made in an ad-hoc and per application manner. In the proposed architecture the management of the context is encapsulated in a single module (Context Manager) and it may be reused from any other application; moreover, it can evolve to include other aspects of context that have not yet been foreseen and determined. The implementation of context adaptation logic (Adaptation Manager) can be supported by utilizing the aspect-oriented paradigm and reflective programming techniques, which assist in managing cross-cutting concerns and dynamically discovering and invoking functionality, respectively.

FUTURE TRENDS

Although research has addressed a number of issues related to context understanding, modeling, reasoning and knowledge sharing, further investigation of them is needed to clarify the relevant concepts. Best practices for analysts to elucidate user requirements pertinent to context adaptability and frameworks for designers and implementers that will facilitate the development of context-aware applications should also be surveyed. In the implementation phase, in particular, the ability to separate handling of context-awareness from application logic is highly desirable, since it will decrease the complexity of application development and will make incorporation of context-aware aspects to existing, non context-aware applications easier.

Currently, context-aware services have not been adopted by the masses to the initially anticipated extent. The reasons for this lag need to be surveyed, and feedback from the existing or potential user groups should be collected and analyzed. Social and legal issues as well as privacy and security concerns related to context awareness should be also considered.

CONCLUSION

In this paper we have discussed the issues of context and context-awareness in the provisioning of mobile services, identified context categories, presented modeling and acquisition techniques, as well as methods of context exploitation. We have presented a generic architecture for supporting context-aware applications, which provides facilities for managing context, performing functional adaptation and tailoring the user interface to suit the current context parameters. The profound understanding of context will facilitate the process of choosing, managing and utilizing it, in order to provide context-aware applications, those ones that deliver the correct service, to the correct user, at the correct place and time, and in the correct format, with the minimum distraction of the user.

REFERENCES

Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., & Zukowski, D. (2000). Challenges: An Application Model for Pervasive Computing. *Proceedings 6th Annual International Conference on Mobile Computing and Networking*, 266-274.

Brown, J., Bovey, D., & Chen, X. (1997). Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications*, 4(5), 58-64.

DeVaul , W., & Pentland, A. (2000). The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications. MIT Technical Report.

Dey, A., & Abowd, G. (1999). Towards a Better Understanding of Context and Context-Awareness. Technical Report 99-22, Georgia Institute of Technology.

Dey, A., Abowd, G., & Salber, D. (2001). A Conceptual framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human- Computer Interaction Journal*, 16(24), 97-166.

Dockhorn Costa P. (2003). Towards a Services Platform for Context-Aware Applications. Master Thesis, University of Twente, The Netherlands.

Efstratiou, C., Cheverst, K., Davies, N., & Friday, A. (2001). An Architecture for the Effective Support of Adaptive Context-Aware Applications. *Proceedings of 2nd International Conference in Mobile Data Management*, 15-26.

Esler, M., Hightower, J., Anderson, T., & Borriello, G. (1999). Next Century Challenges: Data-Centric Networking for Invisible Computing. *ACM/IEEE International Conference on Mobile Computing and Networking*, 256-262.

Eustice, F., Lehman, J., Morales, A., Munson, C., Edlund, S., & Guillen, M. (1999). A Universal Information Appliance. *IBM Systems Journal*, 38 (4), 575-601.

Fickas, S., Korteum, G., & Segall, Z. (1997). Software Organization for Dynamic and Adaptable Wearable Systems. Proceedings of the 1st IEEE International Symposium on Wearable Computers, 56-63.

Indulska, J., Loke, S., Ratotonirainy, A., Witana, V., & Zaslavsky,A. (2001). An Open Architecture for Pervasive Systems. *Proceedings of the 3rd International Working Conference on Distributed Applications and Interoperable Systems*, 175-188.

Kappel, G., Retschitzegger, W., & Schwinger, W., (2001). Modeling Ubiquitous Web Applications:The WUML Approach. *Proceedings of the International Workshop on Data Semantics in Web Information System*.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., et al. (2002). People, Places, Things: Web Presence for the RealWorld, *Mobile Networks and Applications*, 7(5), 365–376.

Mokhtar, B., Fournier, D., Georgantas, N., & Issarny, V. (2005). Context-aware service composition in pervasive computing environments. *Proceedings of the 2nd International Workshop on Rapid Integration of Software Engineering techniques*, 129-144.

Satyanarayanan, M., & Ellis, C. (1996). Adaptation: The Key to Mobile I/O. *ACM Computing Surveys* 28(4es), 211.

Schilit, B., & Theimer, M. (1994). Disseminating Active Map Information to Mobile Hosts. *IEEE Network*, 8(5), 22-32.

Schilit, B., Adams, N., & Want, R. (1994). Context-Aware Computing Applications. 1st International Workshop on Mobile Computing systems and Applications, pp. 85-90.

Schilit, B. Theimer, M., & Welch, B., (1993). Customizing mobile applications. *Proceedings of the USENIX Mobile & Location-Independent Computing Symposium*, pp. 129-138.

Schmidt, A., & Laerhoven, K., (2001). How to Build Smart Appliances. IEEE Personal Communications, 8(4), pp.66-71

Vredenburg, K., Isensee, S., & Righi, C. (2001). User Centered Design: An Integrated Approach. Prentice Hall PTR, ISBN: 0130912956.

TERMS AND DEFINITIONS

Context: Context is any information that can be used to characterize the situation of an entity. An entity is a person, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.

Context-Aware: Context-aware is a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.

Adaptivity or adaptability: It is the ability of a service/application to react to its environment and change its behavior according to the context.

Laisser-faire adaptation: It is the adaptation process to context, which takes place inside the application.

Application transparent adaptation: It is the adaptation process to context, which takes place exclusively outside the application.

Application – aware adaptation: It is the adaptation process to context, which takes place within the application as well as out of it.

Service: It is piece of an information product that materializes a concrete functionality.