POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Context Management in a Pervasive System

Relatore: Prof. Fabio A. Schreiber

Tesi di Laurea di:

SABATINO RACHELE

Matr. 819191

Anno Accademico 2014 – 2015

Abstract

In recent years, ubiquitous computing aimed at creating intelligent environments by embedding devices that provide connectivity and services all the time hiding their presence, improving in this way the human experience. The ultimate goal of a pervasive system is to be perceived as invisible, adapting to new situations without the direct intervention of the user. Adapting to changes in the environment requires the ability of distinguishing different situations or contexts. Context-aware systems enhance the behaviour of an application by gathering data from the environment, interpreting them recognising the context in which the system operates and adapting the system accordingly. The context-aware behaviour can affect both the execution of specific actions and the tailoring of information on the base of the user's preferences reducing the information noise. The present thesis describes the phases of design of context-aware features and their development to a pervasive middleware called PerLa. Originally conceived only for managing and querying a network of sensors, PerLa has been integrated with functionalities to: 1) define the environment with a suitable model; 2) create a context on the defined model; 3) acquire the sensor readings and external inputs which define a specific context; 4) activate or deactivate a context at run-time depending on the actual values of the context variables; 5) perform the context actions on the system.

Contents

Introduction

1.1 Pervasive computing

The aim of developers in creating applications or systems, is the easiness of use for the final user which depends on a number of factors like the intended user, his experience or the technical constraints; the set of these aspects is called the context of use. In earliest days of computing, the context in which systems were used was strongly limited by the place in which computers were set, they did not change easily location, hence, there was no need to adapt to different environments. With the advent of mobile computers and ubiquitous computing, this limitation disappeared as users carried computers with them and used them in many different situations.
In the early 90s, starting with the initial effort of providing the same services to the user independently of his environment, a new approach started to attract the attention of researchers. Bill Schilit [1] introduced the concept of context-aware computing as a software able to adapt itself according to the location of use, nearby people, devices as well as time. In short, a system that can examine, sense the environment and react to changes detected in the environment. The concept of context-awareness expanded and integrated another computing paradigm, presented a couple of years before, called ubiquitous computing. Widely regarded as the father of ubiquitous computing, Mark Weiser, in his work \The Computer for the 21st Century" [2] envisaged a world where computers would have gradually disappeared into the everyday life background until they would have been indistinguishable from it. Weiser was aware that this vision was, at that time, too far away from becoming reality as in order to achieve such disappearance, technologies had to evolve and produce: cheap, low-power computers, communication networks and ad-hoc software systems.

Nowadays developments in technologies such as wireless communications and networking, mobile computing and handheld devices, embedded systems, wearable computers, sensors, RFID tags and similar, have led to make real Mark Weiser's vision. However, advanced hardware technologies are not sufficient by themselves to fully disappear, technologies have to anticipate the user's need freeing him from a direct intervention and acting accordingly. This means to be aware of the surroundings, to have a context-aware behaviour.
Ubiquitous computing, or pervasive computing has been one of the fastest-growing research areas. It aimed at creating intelligent environments by embedding devices which provide connectivity and services all the time, improving in this way human experience and quality of life. In this environment, the world around us is interconnected as a pervasive network of intelligent devices that cooperatively and autonomously collect, process and transport information, in order to adapt to different situations. An example of a pervasive system is the Wireless Sensors Networks (WSN), a network composed of spatially distributed sensors nodes.

Each node is a low-power device that integrates processing, sensing and wireless communication abilities. Sensor nodes acquire information from the surrounding environment, process locally the data, and/or send them to one or more collection base stations. They are useful for the monitoring of physical environment conditions like temperature, pressure, light or location.

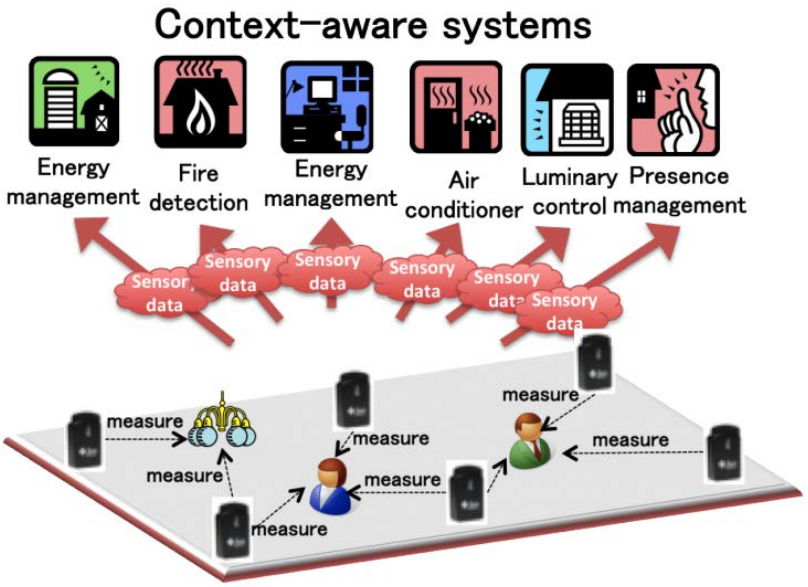

Figure 1.1. An example of a context-aware system

1.2 The notion of Context

In order to consider context effectively, it is important to understand what it is and how it can be used. It can be difficult to give a precise definition; in the years, many researchers have considered context differently according to their field of research. In literature, the first attempt of definition was given by Bill Schilit as the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes over time [1]. A system with these capabilities can examine the computing environment and react to changes in the environment. He developed one of the first pioneer context-aware systems called PARCTAB [3] where he tried to integrate the notion of context to mobile computers used into an office network with the goal of demonstrating the potential for innovation in that area. The device registered the user's id and rough location and on the base of this data, different information and control choices where available.

In [4] the author Dey complained about the lack of a clear definition with the result that it was difficult to have a clear idea of what context was and how it could be effectively used as a source of information in a real operational application. This was due to the fact that the majority of authors [5] defined context by examples or synonyms and, as consequence, the practical usefulness of these definition was limited. For this reason, Dey presented his definition: "Context is any information that can be used to characterize the situation of an entity. An entity can be a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

Context enables to guide the user through the vast amount of information that surrounds him by helping him in discriminating what it is important from what can be ignored. Since the amount of information can grow rapidly and generate confusion, information needs to be filtered adequately so that only relevant information is presented to the user. In addition to prevent useless information to be provided to the user, tailoring the data allows reducing the devices' computational effort and hardware physical constraints in dealing with manageable amount of data. Probably, one of the most significant and widely used context-aware service is Google search engine which returns query results with an order based on many factors like the user's location, web history and navigation, in a word, his context. Another system that can be considered as context-aware are the automatic lights in a building. The contextual parameters taken into account are the current light conditions and if there is motion in the vicinity. The adaptation mechanism is fairly simple. If the situation detected is that it is dark and that there is someone moving, the light will be switched on. The light will then be on as long as the person moves, and after a period where no motion is detected, the light will switch off again.
For a human being, context is often implicit (who I am talking to, where I am, what I am talking about), but an application needs a conceptual model to rely on and translate the information into a meaningful context on which to act. A typical context-aware system acquires data gathered from the real world through sensors, processes the data with some perception algorithm on the base of the context model, recognizes in which context it runs and finally it triggers specific actions.

1.3 Thesis objectives and organisation

Researchers have been investigating many issues of context-aware computing: sensing the environment, context modelling, representation, context inference, knowledge sharing and developing tools and architecture to choose the best way to effectively acquire, represent, and make use of sensed data for providing context-aware services to applications. However, developing context-aware middleware is still a great challenge.
This thesis describes the design and implementation of components which provide
context-aware services to allow an existing middleware for pervasive systems called PerLa, to be context sensitive. PerLa is a project born at the Politecnico di Milano in 2005 when two students worked on a common thesis about the definition of a language, called PerLa Language, for the interrogation of sensors belonging to a WSN. In the following years, other students collaborated in the implementation of the software needed to manage a network of sensors and to query it by exploiting the PerLa Language. As a result, today PerLa can be fully considered as a language and infrastructure for data management in pervasive systems, able to gather data sensed from various heterogeneous devices, to query them and to easily add a new device to the sensor network. Nevertheless, there is still space for improvement with the addition of new features like energy reduction strategies or the expansion to support context-aware applications, the last topic is the main topic of this thesis. The basic idea is to rely on the data sensed from the underlying sensor network already managed by the existing middleware and used this information to recognize a context and change the behaviour of the devices accordingly.

The rest of this thesis is organised as follows:

- Chapter 2 presents part of the literature concerning the concept of context and some of the most recent and interesting context-aware systems. Then, a full overview of the architecture and main features of PerLa is described which acts as the basis for a fuller understanding of the pervasive system which will be the main actor during the rest of this work.

- Chapter 3 describes the formal context model and language used by PerLa to define and store context data in a machine processable form.

- Chapters 4 and 5 focus on the components which are responsible for the flow of operations required to manage the context-awareness from the building and storing of the context model to retrieve the relevant contextual information which allows to distinguish the context in which the system is operating and finally to execute the actions defined in the active context.

- Chapter 6 contains a simulated and simplified example of a scenario where the context-aware PerLa middleware can be deployed showing the advantages and potential use of this system.

- Chapter 7 draws the conclusion and perspective of possible future works.

OMISSIS

Conclusions

7.1 Final consideration

In this work, we presented the components used to add a context management layer to the PerLa System. The components provide a general and robust supporting infrastructure through which the notion of context can be integrated with the existing PerLa functionalities giving a primordial intelligence. Each of the created new components is responsible for a distinct phase during the execution ow of a typical context-aware system. The main goal of the extended middleware is to be of help in monitoring applications that use dynamic and pervasive data for specific purposes. This lead to the following key design principles:

- Easy abstract model. The most critical development point is at design time, when the designer must carefully consider and analyse the environment to select which characteristics are relevant for the application. The type of context information that is relevant to model and handle varies across applications. The designer must have the right expertise to capture significant aspects discarding pointless ones keeping in mind that as the freedom of design is increased, at the same time the system becomes more complex and more difficult to implement. The environment is then categorized as a list of possible situations, contexts which are represented by a model. In the years, many context models have been proposed, from the simple ones like the key-value model to the most complex ones like ontologies. In PerLa, the Context Definition Tree model (CDT) has been chosen and implemented because it is flexible, easy to use and at the same time, it gives the designer a considerable expressive power.
- Support for generic inputs. The designer has to decide which sensor inputs to expect in a certain context. He has to find parameters that define a context and find means to measure those parameters. The PerLa middleware has the advantage of supporting the use of a wide variety of sensors to acquire contextual information. Important sensors used are GPS (for location and speed), light and vision (to detect objects and activities), microphones (for information about noise, activities, and talking), accelerometers and gyroscopes (for movement, device orientation, and vibration), magnetic field sensors (as a compass to determine orientation), proximity and touch sensing (to detect explicit and implicit user interaction), sensors for temperature and humidity (to assess the environment), and air pressure/barometric pressure. There are also sensors to detect the physiological context of the user (e.g. galvanic skin response, EEG, and ECG). Typically, such measurements can be used to determine reactions such as surprise or fear (lie detectors are based on similar mechanisms). In principle, one can use all types of sensors available on the market to feed the system with context information.
- Separation of context inference from application code. Context inference is the process of interpreting and deducing high level context information from raw data retrieved from sensors or external sources. In PerLa, the inference process is performed during the mapping of the raw data to the CDT's variables. The choice of letting the designer specify the mapping together with the context definition, allows the to separate the context inference from the application code in a manner that opens the way to an easy user's configuration.
- Code tailoring. A large number of possible contexts, due to a rich CDT model, asks the designer to write a large number of actions, queries to be executed when a context is discovered as active or inactive which is an

inefficient time-consuming task and error prone. The solution, called code tailoring, relies on a compositional strategy applied at design time: the designer must associate the CDT context elements with fragments of the final contextual blocks. Afterwards, an automatic composition process is performed that produces the complete contextual block. This strategy has the advantage of relieving the designer from a tedious task and at the same time it gives him the control to change the final contextual blocks if needed.

7.2 Open points

Besides the contribution of this work to add context-awareness to PerLa, some functionalities and issues can be explored. In this thesis, the problems arising from conflicts between concurrent contexts and the possible decisional strategies have been discussed. The chosen strategy gives priority to the most recently activated context whenever a conflict may appear. This approach was chosen mainly because the other proposed strategies require changing the Context Language with specific clauses to associate a priority value to each context creating a partial or total ordering among contexts. Although this approach is time-consuming for the designer which has to decide the winner context for every situation in which a conflict may arise, it has the advantage of setting a definite criteria and respect user's preference.

Context information can originate from a wide variety of sources, leading to heterogeneity in terms of quality and persistence. Context inputs can roughly be categorized in sensed, static or user-supplied information. Data produced by sensing nodes is usually highly dynamic and prone to noise and sensing errors, while user-supplied information is initially reliable, but easily becomes out of date; as consequence we have to deal with the problem of imperfect context information. Some context modelling solutions address part of this problem by allowing context information to be associated with quality metadata, such as origin, accuracy, certainty and freshness estimates [31]. However, this approach does not address the entire problem. For instance, it does not provide a solution for representing or reasoning about ambiguity or unknowns. These types of imperfection are common when sensors or other providers of context information report conflicting values or fail to report values at all. In fact, the other type of conflict which we have been neglected up to this point is the data conflict. In other words, an input channel reports information that is clashing with the corresponding information provided by another input channel. Such situations are not as exceptional as they may seem. On the contrary, dealing with inconsistencies means adjusting the level of trust associated with each input channel. In PerLa, sensor raw data are previously aggregated thus significantly reducing noise and inconsistencies between sources. Still, the problem is present as a source could produce a completely senseless sample that compromises the final aggregated result in case the aggregation is evaluated on few samples. In short, data conflicts is more damaging when there are few data sources with the consequence that even if the final data is aggregated, the result can be ruined.

A possible solution can be to add a level of trust to data sources which are more vulnerable to faults and devise algorithms or strategies to deal with conflicting inputs which privilege more reliable sources.

Bibliography

[1] Bill N. Schilit, Norman Adams, and Roy Want. \Context-Aware Computing Applications". In: In proceedings of the workshop on mobile computing systems and applications. IEEE Computer Society, 1994, pp. 85{90.
[2] Mark Weiser. \The Computer for the 21st Century". In: SIGMOBILE Mob. Comput. Commun. Rev. 3 (1999). issn: 1559-1662. url: http : //doi.acm.org/10.1145/329124.329126.
[3] Roy Want et al. \An overview of the PARCTAB ubiquitous computing experiment." In: IEEE Personal Commun. 2.6 (1995), pp. 28{43.
[4] A. Dey. \Understanding and using context". In: Personal Ubiquitous Comput. 5 (2001), pp. 4{7.
[5] N. S. Ryan, J. Pascoe, and D. R. Morse. \Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant". In: Computer Applications in Archaeology 1997. Ed. by V. Ga_ney, M. van Leusen, and S. Exxon. British Archaeological Reports. Oxford: Tempus Reparatum, 1998, pp. 182{196. url: http://www.cs.kent.ac.uk/pubs/1998/616.
[6] Harry Chen. \An Intelligent Broker Architecture for Pervasive Context-Aware Systems". PhD thesis. University of Maryland, Baltimore County, 2004.
[7] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. \Modeling Context Information in Pervasive Computing Systems". In: Pervasive '02 (2002), pp. 167{180. url: http://dl.acm.org/citation.cfm?id= 646867.706693.
[8] Thomas Strang and Claudia Linnho_-Popien. \A Context Modeling Survey". In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England. 2004.
[9] Cristiana Bolchini et al. \A Data-oriented Survey of Context Models". In: SIGMOD Rec. 36.4 (Dec. 2007), pp. 19{26. issn: 0163-5808.
[10] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. \A Survey on Contex Aware Systems". In: Int. J. Ad Hoc Ubiquitous Comput. 2.4 (June 2007), pp. 263{277. issn: 1743-8225. url: http://dx.doi.org/10. 1504/IJAHUC.2007.014070.
[11] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. \Context-aware Systems: A literature review and classification". In: Expert Syst. Appl. 36 (2009). issn: 0957-4174. url: http://dx.doi.org/10.1016/j.eswa.2008.10.071.
[12] Pradeeban Kathiravelu, Leila Shari_, and Lu__s Veiga. \Cassowary: Middleware Platform for Context-Aware Smart Buildings with Software-Defined Sensor Networks". In: ACM, 2015, pp. 1{6. isbn: 978-1-4503-3731-1. url: http://doi.acm.org/10.1145/2836127.2836132.
[13] Tao Gu, Hung Keng Pung, and Da Qing Zhang. \Toward an OSGi-Based Infrastructure for Context-Aware Applications". In: IEEE Pervasive Computing (Oct. 2004), pp. 66{74. url: http://dx.doi.org/10.1109/ MPRV.2004.19.
[14] Patrick Fahy and Siobhan Clarke. \CASS { a middleware for mobile context-aware applications". In: Workshop on Context Awareness, MobiSys. 2004.

[15] Weijun Qin, Yue Suo, and Yuanchun Shi. \CAMPS: A Middleware for Providing Context-aware Services for Smart Space". In: Proceedings of the First International Conference on Advances in Grid and Pervasive Computing. Springer-Verlag, 2006. url: http://dx.doi.org/10.1007/11745693_63.
[16] Jakob E. Bardram. \The Java Context Awareness Framework (JCAF) { a Service Infrastructure and Programming Framework for Context-aware Applications". In: Proceedings of the Third International Conference on Pervasive Computing. PERVASIVE'05. Springer-Verlag, 2005. url: http://dx.doi.org/10.1007/11428572_7.
[17] Adrian Fitzpatrick et al. \Towards a Sentient Object Model". In: In workshop of engineering context-aware object oriented systems and environments. 2002.
[18] F. A. Schreiber et al. \Perla: A language and middleware architecture for data management and integration in pervasive information systems". In: IEEE Transactions on Software Engineering 99 (2011).
[19] Marco Fortunato and Marco Marelli. \Design of a declarative language for pervasive systems". Master Thesis. Politecnico di Milano, 2007.
[20] F. A. Schreiber et al. \Pushing context-awareness down to the core: more flexibility for the Perla language". In: Electronic Proc. 6th PersDB (2012), pp. 1{6.
[21] Cristiana Bolchini, Elisa Quintarelli, and Letizia Tanca. \CARVE: Context-aware Automatic View Definition over Relational Databases". In: Inf. Syst. 38.1 (Mar. 2013), pp. 45{67. issn: 0306-4379.
[22] F. Schreiber et al. \Towards autonomic context-aware pervasive systems: the PerLa context language". In: Electronic Proc. 6th NetDB (2011), pp. 1{7.
[23] JavaCC reference. url: \url{https://javacc.dev.java.net/}.
[24] E. Gamma et al. Design Patterns Elements of Reusable Object-Oriented Software. Ed. by Addison-Wesley. 1995.
[25] K. Sheykh-Esmaili et al. \Changing Flights in Mid-air: A Model for Safely Modifying Continuous Queries". In: ACM SIGMOD International Conference on Management of Data (SIGMOD'11). Athens, Greece, 2011.
[26] G. Pirotta, G. Pisati, and D. Pozzi. \Resolution of Context Conflicts in Pervasive Systems". Master Thesis. Politecnico di Milano, 2011.
[27] Carv project. url: \url{https://www.kickstarter.com/projects/333155164/carv- the- worlds- first- wearable- that- helps- you-ski}.
[28] Snow Googles Airwave. url: \url{http://www.oakley.com/en/mens/goggles/snow-goggles/airwave/product/WGOO7049i}.
[29] Poc Sports. url: \url{http://www.pocsports.com/en/product/1213/receptor-bug}.
[30] Avalanche center. url: \url{http://www.avalanche- center.org/Education/glossary/}.
[31] Hui Lei et al. \The Design and Applications of a Context Service". In: SIGMOBILE Mob. Comput. Commun. Rev. 6.4 (Oct. 2002), pp. 45{55. issn: 1559-1662.