A CONTEXT AWARE FRAMEWORK FOR USER CENTERED SERVICES

SIMON HOH
British Telecommunications Plc, 1B-17, Plaza Sentral, Jalan Stesen Sentral 5, Kuala Lumpur Sentral, 50470 Kuala Lumpur, Malaysia.
Email address: simon.hoh@bt.com

ANUSURIYA DEVARAJU
University Technical Malaysia Melaka (UTeM), 1, Jalan TU43, Taman Tasik Utama, Hang Tuah Jaya, Ayer Keroh, 75450, Melaka, Malaysia.
Email address: anusuriya@utem.edu.my

AND

CHIN CHIN WONG
British Telecommunications Plc, 1B-17, Plaza Sentral, Jalan Stesen Sentral 5, Kuala Lumpur Sentral, 50470 Kuala Lumpur, Malaysia.
Email address: chinchin.wong@bt.com

Abstract. In this paper, we introduce a context aware middleware framework that has been developed over the years to serve as an enabler for user centered services. Firstly, we will discuss about a sensory API mechanism developed to allow an abstraction of sensing elements to report information in a structured manner. We will then proceed to discuss how this sensed information is represented in an ontology, replicating a virtual model of the environment. This will facilitate reasoning capabilities, where entities that are inter-related can be resolved and used by the service. And finally we will describe how context specific to a user-centered service could be subscribed from the middleware. The context, once subscribed, will enable actions to be fired off when the particular context is met. The three core components when put together, will allow for services to react more specific to the users needs, based on the user's ever changing context.

1. Introduction

With the introduction of device heterogeneity, humans will be surrounded by intelligent interfaces supported by computing and networking technologies. Intelligence will be incorporated in everyday objects like clothes, vehicles, picture frames, even the cup of which we drink from. To support the level of complexities that will be introduced, information will need to be filtered adequately to provide only specific information relevant to an individual at any point in time. These intelligence built-in in the environment focuses on performing its specific task well. However, this task may not be necessarily attuned to the user at a specific moment in time. Future applications and services needs to synchronously customised to the users' needs at any moment in time, putting the user as focal point of its operation requirements. This behaviour is known as user-centric behaviour.

Research in context awareness is a very important area for user-centric communications. This is because context awareness provides the ability for solutions to be aware to the situation a user is in, thus providing the ability for such solutions to react around the users every need. In order to achieve such goals, many issues that surrounds how context information can be gathered, represented, processed and consumed appropriately by the solution needs to be looked at.

1.1. PROBLEM STATEMENTS

Context aware applications rely on sensors to observe aspects of the context (Henricksen et al. 2006). The basis of adaptive solutions comes from inputs that form data sets for analysis and design of corresponding prediction model. This input information varies as it could be information based on a physical entity, e.g. person, device, place, or non-physical entity (e.g. activity, mood, time of day). A ubiquitous environment contain a diverse range of sensors, each uses its native access mechanisms and output formats, potentially leading to complexity in system design and implementation (Shchzad et al. 2005). The complexities of this diverse set of input types make it very tricky for solutions to use this information. Most context aware applications embed the interpretation logic of context inside the applications. Delegating the data acquisition and context processing task to the application makes them almost impossible for reuse (Davidyuk et al. 2004; Shchzad et al. 2005). There needs to be a standardised manner to represent these data, validate them against recognisable entities, with a standardised manner of which they could be obtained for solutions to use this information consistently.

The information that are captured via the sensors then needs to be modeled in the computing system, where there are issues concerning sharing of these context information. The five issues identified by Nihei (2004) are:

i. Interconnectivity
ii. Operability
iii. Pre-processing of context information
iv. Largeness of scale and real time sharing
v. Rights management, privacy protection and authentication

The ultimate goal of a context aware application is having the ability to act in response to the situation, when certain context is met. To this end, a standard manner of addressing actions can be carried out is required. Similar to sensors, the diversification of what these actions meant that there are potential issues on how the actions can be understood and triggered.

2. User Centric Solution Requirements

The initial step to provide user centric solutions is to study requirements that are imposed on the solution. To achieve this, one needs to understand the different processes that may happen in an interaction instance.

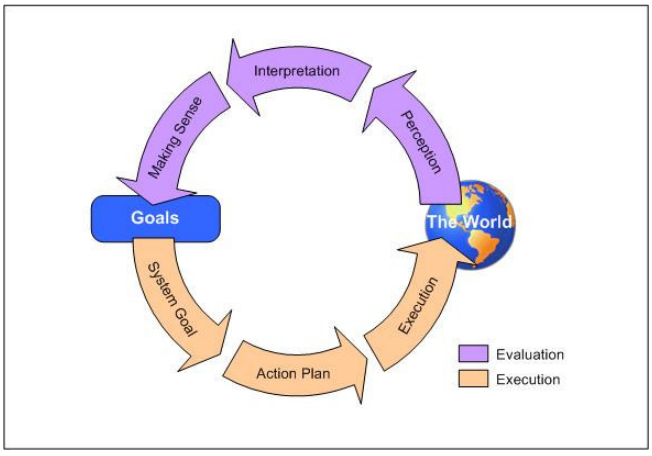

Figure 1. Norman's Stages of Action

As proposed by Arbanowski (2003), the main features of user centric communications of the future are ambient awareness, personalization and adaptability. Ambient Awareness is the ability to sense and exchange information about the current environment an individual is in at a specific

moment in time. Personalisation provides the information necessary to model the preferences of an individual's communication space. And finally, adaptability is the ability to react taking into consideration changes in ambient information and individual preferences.

A more elaborate study of an individual's interaction with objects lead Suttcliffe (2002) to propose 22 generic tasks that describes the different task types involved in the sequence of an individual's interaction. To simplify the discussion of the tasks involved, Montabert (2006) grouped the tasks into Norman's Stages of Action (see Figure 1). The authors shall refer to the Stages of Action mentioned above in the following sections as the core requirements of user centric interaction that may take place.

3. Context Aware Primer and Framework Description

A definition of context awareness is given (Dey et al. 1999) as: '*a system is context aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*'. To simplify the construction of context-aware applications, many context-aware frameworks have evolved during the last years (Baldauf et al. 2007; Singh et al. 2006), where most of them differ in functional range, location and naming of layers, the use of optional agents or other architectural concerns.

3.1. LAYERED ARCHITECTURE FOR A CONTEXT AWARE PLATFORM

The design consideration for constructing context-aware systems is based on the three basic subsystems stated by (Loke 2007): *Sensing*, *Thinking* and *Acting*. Baldauf (2007) describes the three basic subsystems into abstract layered architecture as below (see Figure 2). The *Sensing* subsystem relates to raw sensory information that could be acquired and translated into knowledge. Chen (2004) presented the three different approaches on how to acquire sensor information which are direct access to sensors, middleware infrastructure and context acquisition from a context server. The benefits and drawbacks of the stated approaches have been discussed in our previous paper (Devaraju et al. 2007). A summary of sensing component on existing context-aware systems can be found at Devaraju (2007) and Baldauf (2007).

In the *Thinking* subsystem, the appropriate reasoning technique is then chosen, ranging from simple event-condition rules to sophisticated AI techniques. Some data or knowledge extracted via reasoning might also be stored in this subsystem. Finally, in the *Acting* subsystem, appropriate effectors, hardware and software are employed (Loke 2007).

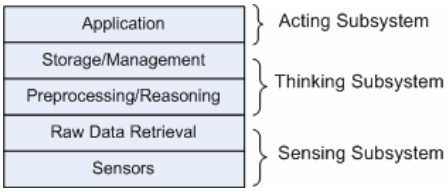

Figure 2. Abstract Architecture for Context Aware Systems

The surveys on context-aware systems by Baldauf (2007), Singh (2006) and Chen (2000) show that existing context-aware frameworks show their similarity concerning the three layered architecture. Yet, this indicates a clear separation of concerns between the context acquisition and user components as proposed by Dey (2000).

4. Mapping the Framework

Using Norman's Stages of Action as requirements for a user centered solutions, we can surmise that *Perception* requires sensory entities implemented that sense and perceive information in real environment. This information is needed to be *interpreted* to provide its meaning to an individual, where definition of context would *make sense* of what information interpreted means. A user centered solution would then be required to define its *goal* when a certain context is met. The reaction of a user centered solution consists of *action plans* declared in its adaptability service taking into consideration preferences of the individual at hand. The *execution* of the action plan is then carried out to provide the necessary user centered behaviour as defined by the solution itself.

5. The Implementation

5.1. CONTEXT AWARE SERVICE PLATFORM (CASP)

The CASP is a middleware-based infrastructure designed to simplify the development of context-aware services by providing a common set of functionalities, which services can simply utilise. It defines a framework for creation of services to facilitate use of pre-built infrastructure to define specific contexts which are relevant to the service, and specific actions that need to be carried out when these contexts are met. It also provides a knowledge base which is a virtual representation of the environment with

the use of ontology, and provides mechanisms to query them. This reduces the efforts required to develop context sensitive services, enabling the service developer to focus more on development of the core service itself.

Figure 3 shows how the layered architecture described earlier suggests subsystems than can be abstracted into our infrastructure.

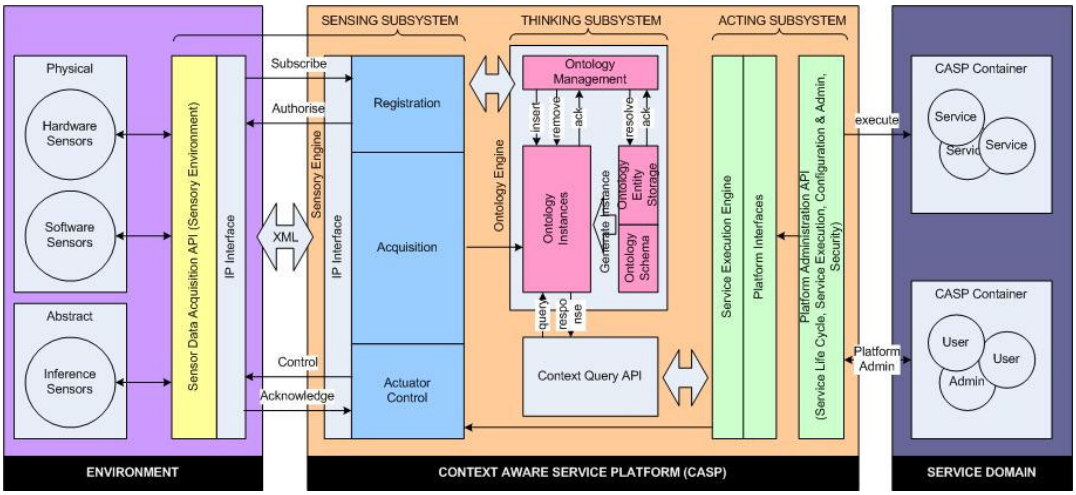

Figure 3. CASP Architecture

5.2. CONTEXT SENSING

The sensing subsystem abstracts sensor types that exist in the physical or non-physical world provide a consistent interface method for sensory information to be programmatically fed into the CASP via its Sensor Data Acquisition API. The sensors used for raw data retrieval can be any hardware, software, or their combination. In similar manner, Indulska (2003) has classified sensors into three groups – Physical sensors, Virtual sensors and Logical sensors. Physical sensors refer to the hardware sensors, while Virtual sensors source context data from software application or services. Logical sensors make use of couple of information sources, and combine physical and virtual sensors with additional information from various other sources. The implementation of the sensing subsystem includes the development of an API that captures abstracted sensory information that is registered and reported to the CASP.

5.3. CONTEXT MODELLING AND STORAGE

The thinking subsystem is made up of an ontology engine that attempts to map sensory information in the real world to object instances in a virtual world that computer systems can interact with. It resolves ownerships and relationships of the sensed information to create a knowledge model of the information in ontology. In addition, the thinking subsystem exposes the query capability for the knowledge model via the Context Query API. The API allows the Service Manager to resolve contexts that are subscribed by a user for a specified service. Here, the Ontology Manager is designed in a manner that ontologies specified by OWL standards could be extended to support different solution domains.

5.4. CONTEXT SERVICE CREATION AND EXECUTION

The acting subsystem consists of the Service Execution Engine, its service creation API and the services itself. Each of these services is provided with knowledge acquisition capability via the Context Query API. Each service conforms to a service wrapper framework requirement that requires it to define the context which would consequently trigger a response. The service wrapper framework is made up of rules that specifies context. Each context is represented by one or more conditions, whereby, if these conditions are met, in would be said to be in the said context. When the context situation is met consequently, an action defined by the service via the service wrapper will be triggered.

6. Conclusion and Future Work

From the implementation of the framework discussed, the authors have demonstrated a system that provides the necessary mechanisms to facilitate easy creation of user centered solutions. However, it is envisaged that services in the future will be automatically subscribed by a user when a user is in a certain context. As such, more work needs to be done to provide an exchangeable dialect which depicts an individual's preferences to the many actions that can be performed automatically by a solution. This would form an extensible preference profile that would form the basis for individuals to manage the level of service automation and profile access.

References

Arbanowski, S.: 2003, I-centric Communications, Technical University of Berlin, Berlin.

Baldauf, M., Dustdar, S., and Rosenberg, F.: 2007, A Survey on Context-Aware Systems, *International Journal of Ad Hoc and Ubiquitous Computing* **2**(4), pp 263-277.

Chen, G., and Kotz, D.: 2000, A Survey of Context-Aware Mobile Computing Research (Technical Report: TR2000-381), Dartmouth College, Hanover, NH, USA.

Chen, H.L.: 2004, An Intelligent Broker Architecture for Pervasive Context-Aware Systems, University of Maryland Baltimore County, Baltimore.

Davidyuk, O., Riekki, J., Rautio, V.-M., and Sun, J.: 2004, Context-Aware Middleware for Mobile Multimedia Applications, Third International Conference on Mobile and Ubiquitous Multimedia, ACM, College Park, Maryland, USA.

Devaraju, A., and Hoh, S.: 2007, A Context Gathering Framework for Context-Aware Mobile Solutions, Mobility Conference.

Dey, A.K.: 2000, Providing Architectural Support for Building Context-Aware Applications, Georgia Institute of Technology, Georgia.

Dey, A.K., and Abowd, G.D.: 1999, Towards a Better Understanding of Context and Context Awareness, Technical Report GIT-GVU-99-22, Georgia Institute of Technology.

Henricksen, K., and Robinson, R.: 2006, A Survey of Middleware for Sensor Networks: State-of-the-Art and Future Directions, Proceedings of the International Workshop on Middleware for Sensor Networks ACM Press, Melbourne, Australia, pp. 60-65.

Indulska, J., and Sutton, P.: 2003, Location Management in Pervasive Systems, Proceeding of the Australasian Information Security Workshop (CRPITS '03), pp. 143-151.

Loke, S.: 2007, *Context-aware Pervasive Systems: Architectures for a New Breed of Applications* Auerbach Publications, New York.

Montabert, C.: 2006, Supporting Requirements Reuse in a User-centric Design Framework through Task Modeling and Critical Parameters, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Nihei, K.: 2004, Context Sharing Platform, *Journal of Advanced Technology* (Special Issue: Advanced Technologies and Solutions toward Ubiquitous Network Society).

Shchzad, A., Ngo, H.Q., Lee, S.Y., and Lee, Y.-K.: 2005, A Comprehensive Middleware Architecture for Context-aware Ubiquitous Computing Systems, *4th Annual ACIS International Conference on Computer and Information Science*, pp 251-256.

Singh, A., and Conway, M.: 2006, Survey of Context aware Frameworks – Analysis and Criticism, University of North Carolina, Chapel Hill.

Sutcliffe, A.: 2002, *The Domain Theory: Patterns for Knowledge and Software Reuse* Lawrence Erlbaum Associates, Mahwah, NJ.