Condor – an application framework for mobility-based context-aware applications

Position paper for the UBICOMP 2002 workshop

W8: "Concepts and Models for ubiquitous computing"

Gothenburg, September 29. 2002
by

Richard Moe Gustavsen (MSc student, University of Oslo)
(Richardg@ifi.uio.no)

Abstract

It is a common understanding that context-aware applications can be difficult to develop [Dey 2000]. Context-awareness is the possibility for an application to use information about a PDA user's environment, including information about the user himself. Usually Context-awareness includes working with sensors, but other methods are also available for acquiring contextual information (e.g. manual input). A PDA-user's context can include physical information such as his position or nearby objects, but also information belonging to categories like system-context, application-context, social-context, historical-context and so forth. A question that arises when developing new context-aware applications is how to make such information accessible for the applications.

To address this issue, I have studied six different (mobile) applications that use context-awareness for presentation of context-relevant information. With context relevant information I mean information that is associated to a contextual situation (like a Stick-e note [Brown 1995], also called tagging of context to information [Dey 2000]), and designed an improved overall application framework for such applications named Condor (Context Document Framework). The most significant innovation of this framework is how context and context relevant information is being addressed: *Rather than trying to separate the functionality regarding context-awareness and context-relevant information into two different frameworks (thereby creating a two-tier application), it sees both context and context relevant information as one common abstraction*. The Condor framework is currently being implemented and tested at SINTEF Telecom and informatics.

Background

Condor was designed as a part of my master thesis (MSc degree) at the University of Oslo [Gustavsen 2002]. With guidance from my advisers at SINTEF (as a part of the AMBiLAB project), I studied six different context-aware applications, all of which has in common that they presented relevant information to the PDA-user according to his contextual situation. Two of these applications were also developed as a part of the thesis. The first, *Mocado* (Mobile Context Aware Demonstrator), is a tourist guide application (similar to The Lancaster Guide [Davies et al. 2001] and Cyberguide

[Abowd et al. 1996]) showing the user's position on a map (as determined from a GPS-sensor), and letting him construct context relevant information according to his position. The latter application, *Mobitras* (Mobile Training Assistant), is on the other hand designed to be a training assistant supporting athletes during their daily trainings. Mobitras is able to announce (by sound) second information as the athlete crosses predetermined checkpoints along a track, and further record the training for an application specific diary. In addition, the recorded trainings can be used as companions (called 'shadows') during subsequent runs, making an illusion of running against other athletes.

Framework requirements

As a result of the abovementioned study, I have established a requirements specification for a generic framework. This specification addresses both functionality regarding context-awareness (e.g. working with sensors) but also requirements for the use of context relevant information:

Req. 1: External sensors – The framework should address a generic system for working with external sensors. An external sensor acquires contextual information from external attached equipment (e.g. a GPS-receiver).

Req. 2: Internal sensors – The framework should address a generic system for working with internal sensors. Internal sensors acquire contextual information without the use of external equipment. This can e.g. be achieved by simple function calls to the underlying API (e.g. time and date).

Req 3: Restricted use of sensors – Only sensors that deliver valid contextual information during execution should be taken dynamically into use by the framework. A thermometer measuring temperatures inside a building should e.g. only be used to describe the PDA-users context when he is inside that building.

Req 4: Dynamic change of sensors – The framework should support dynamic change of sensors. This is the possibility to always use the most accurate set of sensor observations at all times.

Reg. 5: Communication of context – The Framework should be able to handle multiple PDA-users concurrently, and communicate contextual information between them. This stems from the fact that a great deal of a users social context can be made accessible from such communication (e.g. others ID, position, task, history, etc.).

Req 6: Privacy concerns – If the framework has implemented communication of context (req. 5), it should also offer the possibility to maintain such information private. Some of a PDA users context (such as his position) can at different times, and to different people, be of a sensitive character, and as such, not wanted publicly shared.

Req 7: Manual context specification – The framework should enable the use of manual context specification. This is the possibility for the PDA user (via his application) to specify some parts of his contextual situation directly, independent of the underlying sensors (e.g. today I am happy).

Req 8: Overriding sensed context – A PDA user should be able to override already sensed contextual information by the framework (with e.g. manual context specifi-

cation). By letting him override e.g. his position, he can place himself at different places, and thereby simulating being in imaginable situations.

Req. 9: Context history as contextual information – The framework should be able to store previously sensed context in a history log, and use this information to describe the PDA users context at present time.

Req. 10: Context interpretation – The framework should be able to convert contextual information between different formats (e.g. from seconds to minutes), and to deduce new contextual information by analyzing the information that is already known (e.g. user ID, time, and calendar can be analyzed to deduce the users task).

Req. 11: Context subscription and query – Applications developed with the framework should gain full access to the users context. This means the possibility for them to both query and subscribe the framework for contextual changes.

Req. 12: Enabling the use of context relevant information – The framework should address a system to both store and use context relevant information, and to activate it (make it accessible to the applications) according the users context.

Req. 13: Acquiring contextual information from context relevant information – The framework should address a system to acquire contextual information from context relevant information. Such information can, when activated, usually describe a part of the PDA users context (e.g. a piece of information that informs the user about a nearby statue, can also inform the framework about a nearby object)

Req. 14: Dynamic information – The framework should address a system to maintain context relevant information dynamic. Such information may change during execution as a result of contextual changes (e.g. information associated to an object on a boat, would change its geographically position according to the boats location).

Req 15: Context information history – The framework should be able to store dynamically changed context relevant information in a history log (as a result of req. 14), and use this information to describe the PDA users context at present time (if req. 13 is implemented).

Req 16: Multiple deactivation contexts – Context relevant information should be able to associate a deactivation context to their information. The deactivation context specifies when context relevant information should be considered invalid (not relevant for the user) after first found valid. If the information contains different pieces of information, multiple deactivation contexts should be allowed.

Req 17: Authoring of context relevant information – After the user has authored context relevant information (during execution), it should immediately be published and made available for other applications. Such information may also be interesting for PDA users other than the author.

Req 18: Privacy concerns – If req. 17 is implemented, the user should also be able to maintain authored information private. Such information may contain personal information, and thereby not wanted publicly shared.

Req 19: Unrestricted access – The framework should allow a user (via his application) to view context relevant information independent of his contextual situation. Context awareness is a tool for irrelevant information filtration, rather than a restriction for information access.

Similarities regarding contextual-, and context relevant information

Based on the requirements specification, I have constructed the *Condor* framework. Condor differs from other similar frameworks and infrastructures in an important way: *Rather than trying to separate the functionality regarding context-awareness and context-relevant information into two different frameworks (thereby creating a two-tier application), it sees both context and context relevant information as one common abstraction*. This stems from an important observation: Quite often it can be rather difficult to decide whether a piece of information should be considered as *contextual* information or *context relevant* information. The answer usually depends on how applications are using it, meaning that it can belong to both categories at the same time. As an example, consider a nearby place of the user such as a bus stop. In one application this information (about the bus stop) can be handled as contextual information (it describes his physical context), and in the next as information associated to a contextual situation (the information is associated with the bus stop's location, which is near the user). As such, contextual-, and context relevant information can be considered as the same abstraction, depending on the usage.

Context documents – extended stick-e notes

Following the abovementioned similarity, Condor is based upon an abstraction called *context document*. A context document is quite similar to Brown's Stick-e notes, as it contains information associated to a contextual situation [Brown 1995]. However, as an extension to the Stick-e note, a context document also contains contextual information meant for further specification of the users context (called the documents post context). This information is almost the same as the information meant for the user, but differs in that it is modified for the framework parser rather than the PDA-user. A context document can be understood in the following way: When the user is in the same situation as the activation context of the document, the associated user information will be presented to the user (similar to the Stick-e note system), and the associated post context will be used to further describe the users situation by the framework (extending his already known context).

Every Condor application that wishes to publish either a sensor observation or some other contextual information does this by wrapping the information inside a context document. This document is sent to a common database structure on a remote server, which is responsible for storing all published documents for later use. Said in another way, Condor does not distinguish between sensor observations and other contextual information (as e.g. a note containing user information). This means that a sensor observation will contain an associated activation context (contained in the document) that tells under which circumstances the observation is to be considered valid. A temperature measurement may e.g. be considered valid only inside the building where it was measured.

After publication, a context document will be available for every other client connected to Condor (see figure 1).

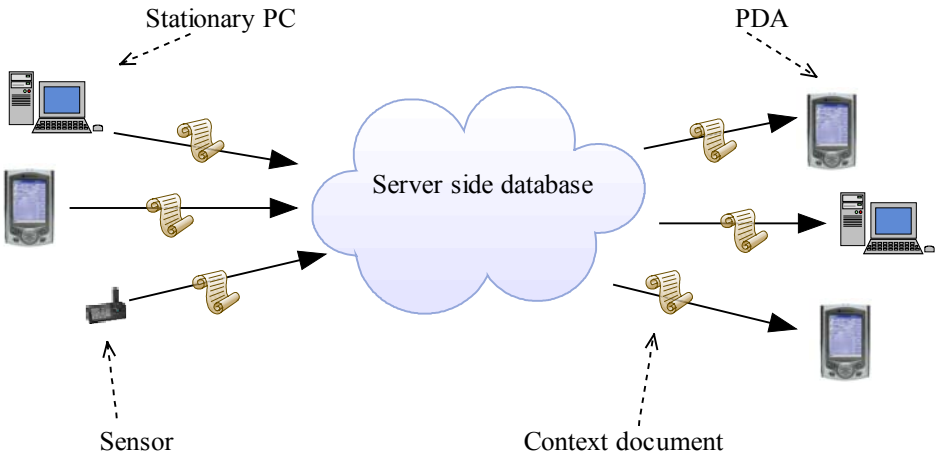

Figure 1: Condor – The figure shows the overall framework architecture. Every client application can publish sensor observations or other information as context documents.

Finally, each application in Condor can construct one or more *entities*. Entities can represent both concrete and abstract physical artifacts (e.g. a person, a dog, a car, a place, a *sensor*, etc.) and have two intended responsibilities: Firstly, they will continually search the server side database for any documents that can be found relevant according to their own contextual situation, and secondly, they will publish documents about themselves to the same place at regular intervals. A person-entity would e.g. contain an activation context specifying the person's location, a post context specifying contextual information back to the triggering entity (e.g. "nearby persons = John") and user-information for the application user (as will be presented on the PDA, e.g. "you are sitting next to John"). Since similar documents about a person will be published at regular intervals (e.g. each time his location changes) he will leave a personal trace [Rahlff et al 2001] on the server side database.

One intended use of the Condor entities is to let PDA-applications model their own mobile user, and thereby let the user "live a life" inside the framework. In this way, the entity will gather relevant information from the server side database (according to the PDA-users situation), and send it back to the owner application at regular intervals.

Conclusion

As a conclusion of my thesis, I found that Condor would have addressed nearly all of the requirements established in the requirements specification. I also stated that the development time and application complexity of both Mocado and Mobitras would have been significantly reduced if Condor had been available when I first started writing my thesis. I believe the abstractions contained in the framework will prove useful for future implementations of context-aware systems.

References

[Abowd et al. 1996] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton.

”Cyberguide: A Mobile Context-aware Tour Guide”.
Georgia Institute of Technology, Atlanta, 1996.

[Brown 1995] P. J. Brown.
”The stick-e document: a framework for creating context-aware applications”.
The University of Canterbury, Kent CT2 7NF, UK., 1995.

[Davies et al. 2001] N. Davies, K. Cheverest, K. Mitchell, A. Efrat.
”Using and Determining Location in a Context-Sensitive Tour Guide”.
Computer, vol. 32, nr. 8, August 2001.

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

[Gustavsen 2002] Richard Moe Gustavsen.
”Condor – et rammeverk for mobilitetsbaserte kontekstoppmerksomme applikasjoner.” (Condor – a framework for mobility based context-aware applications [in Norwegian])
Universitetet i Oslo, Institutt for informatikk, 2002.

[Rahlff et al. 2001] Rahlff, O. W., Rolfsen, R. K., Herstad, J., Using Personal Traces in Context Space: Towards Context Trace Technology,
Springer's Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Context-Aware Computing, Vol. 5, Number 1, p. 50-53, 2001.