# Programming Framework for Sensor-data Driven Context-Aware Applications

Devdatta Kulkarni (dkulk@cs.umn.edu) *
Advisor: Dr. Anand Tripathi
Department of Computer Science, University of Minnesota Twin Cities MN 55455, USA

## Abstract

*Context-awareness is one of the central characteristics of pervasive computing applications. We have built a programming framework for constructing context-aware applications from their high-level speci£cations. This framework automatically constructs the runtime environment for an application to enforce application de£ned policies for dynamic composition of resources and services based on context events derived from ambient sensors.*

## 1 Introduction

Recent trend in number of application domains such as, assisted living [3], hospital information systems [2], tour guides [1], smart environments [6], is towards integrating context information to dynamically adapt an application's behaviour to provide enhanced functionality to users. Typical examples of context information include user location, co-location of users, co-location of a user with a device or an object, devices being used by a user, and so on. The typical characteristics of context aware applications include context-based recon£guration, context-triggered actions, context-based information access, context-based access control and multi-user coordination [5].

The main theme of my Ph.D. research has been on developing a programming framework to support rapid construction of context-aware applications from their high-level speci£cations. The middleware generates the runtime environment of the application from the application's speci£cation. Middleware provides services for resource registration and discovery, location-independent naming, and context agents for detecting and aggregating sensor data. The main advantage of this approach is that the task of developing context-aware applications is simpli£ed because of the following reasons. First, the application programming efforts are limited only to developing the design speci£cation and the required application components. Second, the runtime environment for the application is automatically generated and maintained by the middleware, based on the design speci£cation.

One of the crucial requirements of context-aware applications is real-time collection and aggregation of sensor data from sensors distributed in the environment. For this purpose I have developed a reusable context information collection and aggregation system using a distributed agent-based event aggregation system. In this system, one can de£ne agents for monitoring and collecting context information by tracking user Bluetooth devices, RFID tags, and GPS enabled devices. Also, agents can be de£ned to represent physical spaces, such as rooms. Such agents maintain state information associated with the corresponding physical space. All agents support query and noti£cation interfaces and generate context events, which may be subscribed to by context-aware applications.

## 2 Hospital Information System

I will discuss in this workshop a medical domain application that exhibits novel context-based requirements. It supports storage, retrieval, and access of patient information. Doctors and nurses may access this information through their mobile personal devices. We consider the following context-based requirements for this application, some of which have been raised by others [4].

A nurse may access medical records only if some doctor is also present in the ward where she is present. Moreover, only those nurses who are on a patient's medical assistance team may access the patient's medical records. We may also require that a nurse's mobile device automatically binds to the patient's records to whom she is currently attending. The application also allows nurses to post alerts for other nurses in a ward. Such alerts are noti£ed to a nurse when she enters a ward.

For realizing above application, one needs to tackle following interdisciplinary research problems.

• Designing models for real-time sensor data aggregation to support application-speci£c context detection requirements.

• Mechanisms for policy speci£cation for adapting an ap-

*This work was supported by NSF grant 0411961.

plication in a given physical space based on ambient conditions.

• Mechanisms for context-based access control policy speci£cation and enforcement for an application.

## 3 System Architecture

In our programming framework, a context-aware application is programmed using an abstraction called *activity*. An activity de£nes a shared object space, and a set of user roles. Various resources/services required by the application are accessed as *objects* within the activity. Operations are associated with a *role*, through which a role member may perform application tasks. Context-based access control requirements are programmed using two mechanisms, *precondition*, and *access constraint*. We provide the *reaction* mechanism for programming context-triggered actions. A reaction is similar to a role operation, the only difference being that it is triggered by an event and executed by the runtime system, rather than executed by any role member.

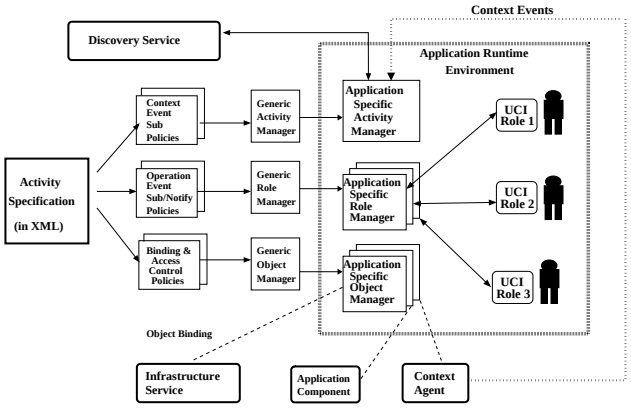


**Figure 1. Process of generating a context-aware application's runtime environment**

In Figure 1 we present the main elements of the middleware. The middleware provides three generic components, an *activity manager*, a *role manager*, and an *object manager*. The runtime environment of an activity is constructed by *deriving* policies from its XML speci£cation, and integrating them with the generic managers to construct application speci£c managers.

The policies that are derived include object binding and method level access control policies for object managers, operation execution and event subscription/noti£cation policies for role managers, and context event subscription policies for the activity manager. An object manager maintains a reference to the service to which the object is currently bound. These managers are run on a set of *trusted servers*.

## 4 Lessons Learned

I would discuss the following lessons that emerged as part of designing this programming framework and implementing a number of context-aware applications using it.

• Distributed agent based architectures can be effectively utilized for real-time processing of sensor data streams for detecting context conditions required by an application.

• If a context event triggers dynamic binding of multiple objects, then the order in which the event is dispatched to these objects can be crucial for correctness of the application's behavior.

• Concurrent executions of reactions triggered by asynchronous context events can lead to incorrect behavior of an application.

• During the course of execution of a context-dependent task, it is possible for the related context condition to become false. This can be crucial for the correct enforcement of context-based access control requirements. We refer to this as the *context invalidation problem*.

• The precondition mechanism is inadequate for specifying context-based access control conditions in which a role member's access to a resource needs to be restricted based on the relationship of certain attributes of the resource to the role member's current context. We refer to this as the *context-based resource view control problem*.

## References


[1] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: a Mobile Context-aware Tour Guide. *Wirel. Netw.*, 3(5):421–433, 1997.

[2] J. E. Bardram, T. R. Hansen, M. Mogensen, and M. Søgaard. Experiences from real-world deployment of context-aware technologies in a hospital environment. In *Ubicomp*, pages 369–386, 2006.

[3] S. Consolvo, P. Roessler, B. E. Shelton, A. LaMarca, B. Schilit, and S. Bly. Technology for care networks of elders. *IEEE Pervasive Computing*, 3(2):22–29, 2004.

[4] M. Evered and S. Bögeholz. A case study in access control requirements for a health information system. In *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 53–61, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[5] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, US, 1994.

[6] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, and F. Liu. The smart classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 02(2):47–55, 2003.