Ubiquitous Computing Environments and Its Usage Access Control

Hua Wang [1] Yanchun Zhang [2] Jinli Cao [3]
[1] Department of Maths & Computing, University of Southern Queensland
Toowoomba QLD 4350 Australia
Email: wang@usq.edu.au
[2] School of Computer Science and Mathematics
Victoria University of Technology, Melbourne City, MC8001, Australia.
Email: yzhang@csm.vu.edu.au
[3] Department of Computer Science & Computer Engineering
La Trobe University, Melbourne, VIC 3086, Australia
Email: jinli@cs.latrobe.edu.au

Abstract

Ubiquitous computing aims to enhance computer use by utilizing many computer resources available through physical environments, but also making them invisible to users. The purpose of ubiquitous computing is anywhere and anytime access to information within computing infrastructures that is blended into a background and no longer be reminded. This ubiquitous computing poses new security challenges while the information can be accessed at anywhere and anytime because it may be applied by criminal users. The information may contain private information that cannot be shared by all user communities. Several approaches are designed to protect information for pervasive environments. However, ad-hoc mechanisms or protocols are typically added in the approaches by compromising disorganized policies or additional components to protect from unauthorized access.

Usage control has been considered as the next generation access control model with distinguishing properties of decision continuity. In this paper, we present a usage control model to protect services and devices in ubiquitous computing environments, which allows the access restrictions directly on services and object documents. The model not only supports complex constraints for pervasive computing, such as services, devices and data types but also provides a mechanism to build rich reuse relationships between models and objects. Finally, comparisons with related works are analysed.

1 Introduction

Ubiquitous computing is a broad semantic definition. In many cases, researchers define ubiquitous computing in their research projects through examples. We use Weisers definition of ubiquitous computing [20], that is, ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.

The main future of ubiquitous computing is to create a user centric and application oriented computing environment. Such environments are different from the traditional computing models since a physical space within the environments supported by associated hardware and software facilitates interactive information exchange between users and the space. The availability of cheap computing devices and wireless networks are making such spaces possible. A user does not require to log into a single personal computer as in traditional computing environments, but communicates with a variety of computing devices in the space. Scalable configuration is an important aspect in such space since the same space is often used for different tasks at different times. Contextual information, such as the current users in the space, or the current activity, is important for the configuration.

The widely use of advanced technology in computing, network and sensor has facilitated the development of ubiquitous computing, enabling various convenient applications. Resources and information in ubiquitous computing environments are shared by users, heterogeneous sensors and so on. Security issues are vital in the environments since contextual information such as sensor locations and applications become an integral part of the system authorization. On the other hand, a variety of applications and users interaction with the pervasive environment poses new security challenges to the traditional user-password approach for computer security. The heterogeneous devices and mobile users in such dynamic pervasive computing environments make security management difficult, especially the access to authorized users since it is a basic security requirement for guaranteeing user's privacy, information confidentiality, integrity and availability.

The security architecture of ubiquitous computing and its applications are in infant stage, but an active research area [3, 21, 13]. Several papers have analysed the security requirements for ubiquitous computing [11, 15, 13, 7]. An active space is proposed in [11] which can be configured for different types of applications at different times. Role-based access control [12] techniques for easy administration of users and permissions are applied to create and enforce access control policies for different configurations of the active space. Through an example scenario, dynamic protection domains are built for the assignments of permissions to roles and roles to users based on context information. A security architecture is designed for a research project GAIA operation system [15]. The architecture provides dynamism and flexibility to manage the security concerns in such a computing system with physical spaces of hundreds of computational devices extending the user's view of the computational environment beyond the physical limitations of a traditional distributed system. A protocol is presented in [7] which preserves the privacy of users and keeps their communication anonymous. In this protocol, a "MIST" is created that can protect user's

ID from the system and other users and users are able to enjoy seamless interaction with services and other entities within the ubiquitous computing environment.

An application of payment prototype in ubiquitous computing is analysed in [13]. Computational models of trust are proposed for use in pervaive environments for deciding whether or not customers are allowed to pay with an e-purse. Authors have built a scheme (and its prototype) that mitigates this kind of loss of privacy without forbidding the use of trust for smoothing payment by giving the opportunity to the user to divide trust (i.e. transactions) according to context.

There are several security issues that need to be addressed in the application of ubiquitous computing. In particular, the following requirements are critical to developing a secure and flexible access control architecture.

1. Access based on contextual information.

   Existing access approaches are developed in the context of traditional distributed and multi-user systems which specify the allowed accesses for users or subjects to resources or objects. The users and objects in traditional systems are static, but in ubiquitous environments both subjects and objects may enter and leave the space dynamically. There is no mechanism for expressing other factors that may influence authorization.

2. Access for different users and devices. Ubiquitous computing may also be used for non-technical applications. Users in the pervasive environment may not have any special training and prefer the easy use of the devices in the environments.

3. Access control in a collaborative environment. Ubiquitous computing is often used for collaborative services, where many users work together to complete a task [17]. An access control model in ubiquitous environments has to provide a secure collaboration structure for the users sharing of permissions in the collaborative task between dynamic users.

4. Decentralized administration. A physical space in ubiquitous computing is usually part of a larger system, and the access control policy may be organized by the administrators of the larger systems and the space.

The second requirement is about human computer interface which is beyond this paper. We focus the remaining three requirements. This paper presents authorization models which adopt usage control to manage access to the space and secure architectures to perform the authorization models. Traditional access control has analyzed authorization decisions on a subject's access to target resources. "Obligations" are requirements that have to be followed by the subject for allowing accessing resources. "Conditions" are subject and object independent requirements that have to be passed. In ubiquitous environments, both users and objects are highly dynamic, obligations and conditions of new hosts are decision factors for the access management.

Traditional authorization decisions are generally made at the time of requests but do not consider ongoing controls for long access or for revocation. The objects including devices in pervasive computing space are used by various users. There are

complex relationships among users, objects, and arbitrary authorizations between users and objects. An object can be a device, a file, dynamically generated documents, and so on. Because of the complex ubiquitous environment, users are required to obey obligations and satisfy conditions and ongoing control with different security policies. Usage control [9] has been recognized as the next generation access control to be efficient in security administration. Authorizations, obligations and conditions are used to build a secure architecture. The ongoing control provides dynamic access verification for contextual information.

The remainder of this paper is organized as follows: Section 2 presents usage control and ubiquitous computing model including the relationships between space objects and users, and space objects and applications. Three decision factors *Authorization, Obligation, Conditions* and Continuity properties *pre* and *ongoing* are introduced in this section. Section 3 develops authorization models with usage control for ubiquitous computing. It includes *pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations*,
pre-Conditions and *ongoing-Conditions*. Section 4 discusses how to build secure architectures for collaborations in distributed administration by using reference monitors in details. Section 5 compares our work with the previous work on pervasive computing security. The difference between this work from others is presented. Section 6 concludes the paper and outlines our future work.

2 Related technologies

2.1 Usage control

Usage control is used for the access control in the pervasive environment. There are eight components: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions in usage control model [9] (see Figure 1). Subjects and objects are familiar concepts from the traditional access control, and are used in their familiar sense in this paper. A right represents access of a subject to an object, such as read or write. The existence of the right is determined when the access is attempted by the subject. The usage decision functions indicated in Figure 1 make this determination based on subject attributes, object attributes, authorizations, obligations and conditions at the time of usage requests.

Subject and object attributes can be used during the access decision process. Examples of subject attributes are identities, group names, roles, memberships, credits, etc. Examples of object attributes are security labels, ownerships, classes, access control lists, etc. In an on-line shop a price could be an object attribute, for instance, the book Harry Potter is priced at $20 for a read right and priced at $1000 price for a resell right.

Authorizations, obligations and conditions are decision factors used by decision functions to determine whether a subject should be allowed to access an object. Authorizations are based on subject and object attributes and the specific right. Authorization is usually required prior to the access, but in addition it is possible

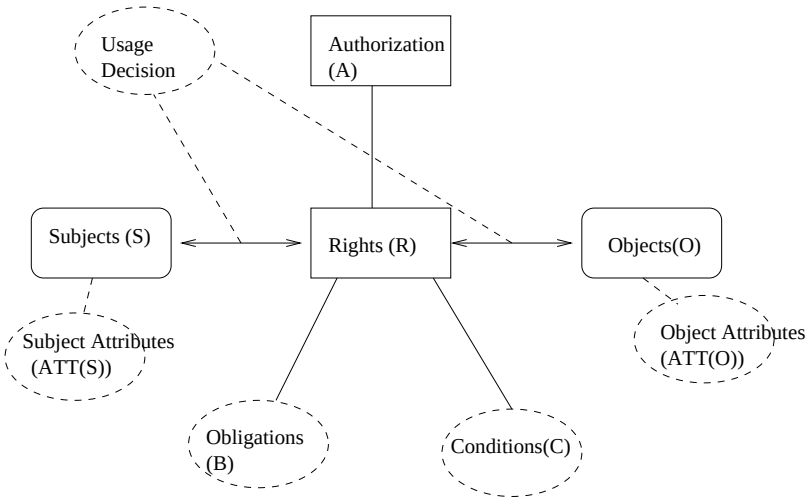

Figure 1: Components of Model

to require ongoing authorization during the access, e.g., a certificate revocation list (CRL) may be periodically checked while the access is in progress. An access is immediately revoked if the relevant certificate appears on the CRL. Authorizations may require updates on subject and object attributes. These updates can be either 'pre', 'ongoing', or 'post' that are called continuity properties shown in Figure 2.

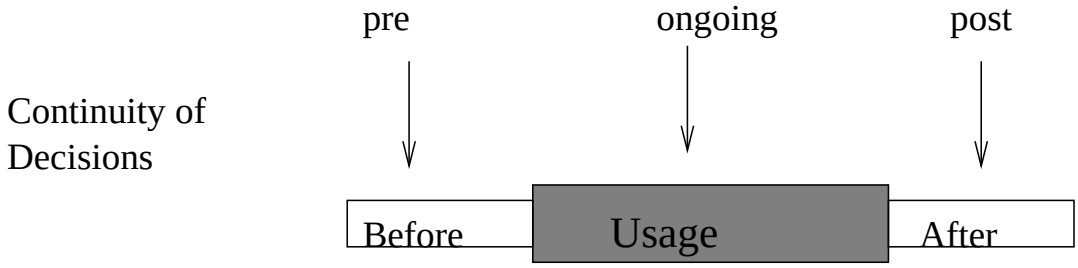

Figure 2: Continuity Properties

For example, pre-paid mobile phone requires update of the subjects (users) clearance prior to access and also requires periodic updates of the remaining credits while usage is in progress, with possible termination in case of overuse. Fixed phone payment system requires updates after the usage has ended to calculate current usage time. Obligations are requirements that a subject must perform before (pre) or during (ongoing) access. An example of a pre-obligation is the requirement that a user must provide some contact and personal information before accessing IEEE digital library. The requirement that a user has to keep certain advertising windows open while he is accessing some service, is an example of an ongoing obligation. Subject and object attributes can be used to decide what kind of obligations are required for access approval.

Conditions are decision factors that depend on environmental and system-oriented requirements. For example, IEEE member can access full papers in the IEEE digital

library. Conditions can also include the security status of the system, such as low level, normal, high alert, etc.

As discussed above, continuity is another decision factor as shown in Figure 2. In traditional access control, authorization is assumed to be done before access is allowed (pre). However, the dynamic changes of contextual information in ubiquitous computing environments, it is quite reasonable to extend this for continuous enforcement by evaluating usage requirements throughout usages (ongoing).

2.2 Ubiquitous computing model

The basic concept in usage control is the access right to an object, which is called usage. Users are assigned usage when they enter a special space of ubiquitous computing environments, and access policies for services in the space are generated by assigning access rights to users. The objects in pervasive environments are services, devices, dynamic generated objects. The access rights, for example, to a device are the set of operations that can be performed by users and to a service are the set of operations that can be applied by users.

We introduce the concept of a space object, with associated space rights, which are permissions to resources within the space. Access control policies for the space are described by space objects and rights, and this simplifies the task of the space administrator. The main future of ubiquitous computing is to create a user centric and application oriented computing environment. Therefore, we consider users and applications in this pervasive environments. Users who want to access the space have their accounts created by system administrators and are assigned a usage of access space objects based on their rights and responsibilities within the space. When a user with a usage enters to a space, the user is automatically assigned a space object, which is restricted to a set of rights that make sense within the space. Figure 3 shows how users are assigned to space usages.

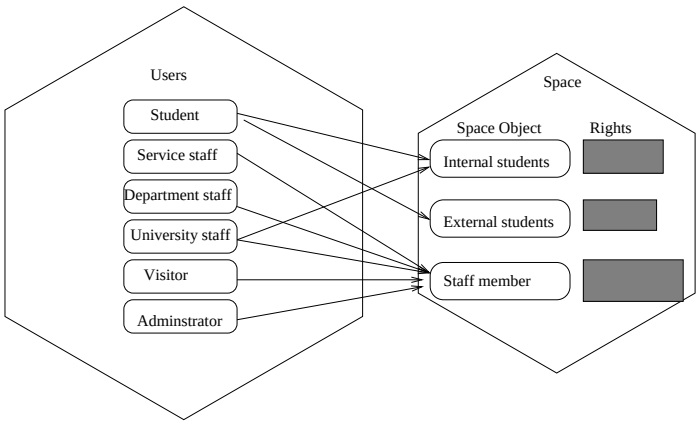

Figure 3: User - space usage assignments

For instance, a space administrator could create a usage of interview-room that is allowed to setup the space as an interview room, and decide that only users of human

resource offices could be assigned this usage. An officer in human resource may have many other permissions in the entire system, but when the office is assigned into the usage of interview-room, it is only allowed rights related to that task.

A user's rights in a space are a subset of his rights within the entire system, it means that rights assigned to a space object are always a part of rights to the corresponding system.

Application usages are used to specify access control policies for applications. These application usages are mapped into space usages by the space administrator. For example, as shown in Figure 4, an application for an interview may have two application usages: committee members and candidates. These two usages are decided with the functions of different participants in the application. Candidates should access to a presentation device and committee members must have read access to the slides presented. The specific rights that a candidate requires depends on the devices in the space that are running the application. Application usages and users are assigned by the space administrator to appropriate space objects and their access rights, and access control within a space is only enforced in terms of these space usages.

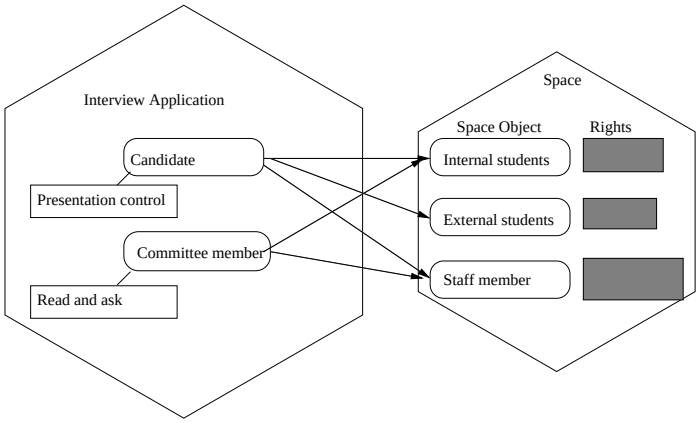

Figure 4: Application - space usage assignments

The protection domains in our model are defined by authorization approaches in the space. When the space switches modes, new objects may be dynamically created, based on the context of the space, and rights automatically assigned to them.

3 Authorization models

We now discuss authorization models for space objects adopting usage control in this section. Based on three decision factors: authorizations, obligations, and conditions, we develop a family of core models for usage control. By core models, we mean that they focus on the enforcement process and do not include administrative issues. We assume there exists a usage request on space object. Decision-making can be done either before (pre) or during (ongoing) exercise of the requested right. Decision-making after the usage has no influence on the decision of current usage.

Based on these criteria, we have 6 possible cases as a core model for usage control in the space: pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions. Depending on the access requirements on pervasive computing environments in real world, it is possible to utilize more than one case.

For simplicity we consider only the pure cases consisting of *Authorizations, Obligations* or *Conditions* alone with *pre* or *ongoing* decisions only. We focus on developing comprehensive usage control models for the space objects. Next we present usage control models (UCM) with different pure cases.

A. UCM_{preA}**:pre-Authorizations Model**

In an UCM_{preA} model, the decision process is performed before access is allowed. The UCM_{preA} model has the following components:

1. $S, Obj, App, ObjR, AppR, ATT(S), ATT(Obj), ATT(App)$ and usage decision Boolean functions $preA, preA_1$ on Obj for S and App, respectively.

   Where $S, Obj, App, ObjR, AppR$ represent Subject, Space object, Application in a pervasive computing environment and Rights required on the object and on the application (e.g. read, write) respectively. $ATT(S), ATT(Obj), ATT(App)$ represent attributes of subjects, objects and applications in the environments respectively. $preA$ and $preA_1$ are predicates about authorization functions. Fox example, when users enter the space, $preA$, or $preA_1$ is a function on users' account and password (subject attributes) or service tickets , objects and rights (read, write or resell) that are used to check whether users can access the objects with the rights or not,

2. $allowed(S, Obj, ObjR) \Rightarrow preA(ATT(S), ATT(Obj), ObjR)$,

   Where $A \Rightarrow B$ means B is a necessary condition for A. This predicate indicates that if subject S is allowed to access object Obj with right $ObjR$ then the indicated condition $preA$ must be true.

3. $allowed(App, Obj, ObjR, AppR) \Rightarrow preA_1(ATT(App), ATT(Obj), ObjR, AppR)$.

   The $allowed(App, Obj, ObjR, AppR)$ predicate indicates that if application App is allowed to map a space object Obj with right $ObjR$ then the decision function $preA_1$ is true.

The UCM_{preA} model provides an authorization method on whether a subject can access an object or not, and whether an application can map to an object or not. The $allowed(S, Obj, ObjR)$ predicate shows that subject S can access some part of information in the space. At this stage, private information in the ubiquitous computing environments is restricted.

B. UCM_{onA}**:ongoing-Authorizations Model**

An UCM_{onA} model is used to check ongoing authorizations during access processes. The UCM_{onA} model has the following components:

1. $S, Obj, App, ObjR, AppR, ATT(S), ATT(Obj), ATT(App)$ as before, and ongoing usage decision functions onA on Obj (Space object) and onA_1 on both Obj and App,

   onA is used to check whether S can continue to access the object or not, and onA_1 is to check whether App can continue to map the object or not.

2. $allowed(S, Obj, ObjR) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on Obj.

3. $allowed(App, Obj, ObjR, AppR) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on Obj.

4. $stopped(S, Obj, ObjR) \Leftarrow \neg onA(ATT(S), ATT(Obj), ObjR)$,

   The access of subject S to Obj is terminated if the ongoing authorization onA is failed.

5. $stopped(App, Obj, ObjR, AppR) \Leftarrow \neg onA_1(ATT(App), ATT(Obj), ObjR, AppR)$.

   The map of App to Obj is terminated if the ongoing authorization onA_1 is failed.

UCM_{onA} introduces the *onA, onA*$_1$ predicate instead of *preA, preA*$_1$. $allowed(S, Obj, ObjR)$ and $allowed(App, Obj, ObjR, AppR)$ are required to be *true*, otherwise ongoing authorization should not be initiated. Ongoing authorization is active throughout the usage of the requested right, and the onA and onA_1 predicates are repeatedly checked for continuation access. These checks are performed periodically based on time or event. The model does not specify exactly how this should be done. When attributes are changed and requirements are no longer satisfied, *stopped* procedures are performed. We use $stopped(S, Obj, ObjR)$ and $stopped(App, Obj, ObjR, AppR)$ to indicate that rights $ObjR$ of subject S on object Obj and $AppR$ of application App are revoked and the ongoing access terminated. For example, suppose only one candidate can access the presentation device in an interview room simultaneously. If another person requests access and passed the pre-authorization, the user with the earlier time access is terminated. While this is a case of ongoing authorizations, it is important that the certificate should be evaluated in a *pre* decision.

C. UCM_{preB}:pre-Obligations Model

UCM_{preB} introduces pre-obligations that have to be fulfilled before access is permitted. Examples of pre-obligations are requiring a student to register by filling forms before accessing an education website, requiring the student to click the ACCEPT box on a license agreement to run some software such as "$Slide - show$" and "$Sample - demo$", etc. The pre-obligation action may perform on a different object (e.g., register, license agreement) than the object that the user is trying to access (e.g., Slide-show). It means that the pre-obligation action may be done by some other subject. Hence obligation subjects, objects, and actions are added in the following UCM_{preB} model.

For simplicity, the following model is described for subject and object only. It can directly be used for any part of restricted information in the space.

The UCM_{preB} model has the following components:

1. $S, Obj, ObjR, ATT(S), ATT(Obj)$ as before,
2. OBS, OBO, and OBA represent obligation subjects, objects, and actions, respectively; decision function $preObfilled : OBS \times OBO \times OBA \rightarrow \{true, false\}$,

   As mentioned above, subject S and access object may be different from OBS and OBO, hence, we separately describe them. The function $preObfilled(s, xd, r)$ is used to check if obligations are obeyed or not.
3. $allowed(S, Obj, ObjR) \Rightarrow preObfilled(S, Obj, ObjR)$.

   The $preObfilled(S, Obj, ObjR)$ function must be true if S is allowed to access Obj with right $ObjR$.

The model indicates that obligations have to be fulfilled before S can access Obj. Note that each obligation has to be true if there are more than two obligations.

D. UCM_{onB}: ongoing-Obligations Model

Obligations are required to fulfill in UCM_{onB} models while rights are exercised. Ongoing-obligations may have to be fulfilled periodically or continuously. For example, a user may have to open a security policy window at least every 30 minutes. Alternatively, a user may have to leave the window active all the time with inconvenience. The model concerns obligations that have to be fulfilled.

The UCM_{onB} model has the following components:

1. $S, Obj, ObjR, ATT(S), ATT(Obj)$ as before,
2. OBS, OBO, and OBA represent obligation subjects, objects, and actions, respectively; an ongoing decision function $onObfilled : OBS \times OBO \times OBA \rightarrow \{true, false\}$,

   The ongoing function $preObfilled(S, Obj, ObjR)$ is used to check if obligations are continually obeyed or not.
3. $allowed(S, Obj, ObjR) = true$,

   A prerequisite for UCM_{onB}. It means S is accessing Obj.
4. $stopped(S, Obj, ObjR) \Leftarrow \neg onObfilled(S, Obj, ObjR)$.

Where $stopped(S, Obj, ObjR)$ indicates that the access of S on Obj with $ObjR$ is revoked if the ongoing obligations are failed.

E. $UCM_(preC)$: pre-Conditions Model

As described earlier, conditions are not directly related to subjects and objects since they define environmental and system restrictions. We focus on this model for subject and space object. The model can also be used for restricted device in the space.

The UCM_{preC} model has the following components:

1. $S, Obj, ObjR, ATT(S), ATT(Obj)$ as before,
2. $preCON$(a set of pre-conditions), verify conditions function $preConSatisfied : preCON \rightarrow \{true, false\}$,

   The function $preConSatisfied$ is used to check whether the pre-conditions are satisfied or not.
3. $preC(S, Obj, ObjR) = \bigwedge_{preCon_i \in preCON} preConSatisfied(preCon_i)$.

   All pre-conditions have to be checked if there are more than two conditions.
4. $allowed(S, Obj, ObjR) \Rightarrow preC(S, Obj, ObjR)$.

The third component indicates situations with more than two conditions and the $allowed(S, Obj, ObjR)$ expresses that all conditions have to be satisfied before access is approved.

F. UCM_{onC}: ongoing-Conditions Model

UCM_{onC} model requires conditions to be satisfied while rights are in active use. For example, *realOne player* does not work when Windows XP system works on safety module.

The UCM_{onC} model has the following components:

1. $S, Obj, ObjR, ATT(S), ATT(Obj)$ as before,
2. $onCON$(a set of ongoing conditions), verify ongoing conditions function $onConSatisfied : onCON \rightarrow \{true, false\}$,

   The function $onConSatisfied$ is used to check whether ongoing conditions are satisfied or not.
3. $onC(S, Obj, ObjR) = \bigwedge_{onCon_i \in onCON} onConSatisfied(onCon_i)$,

   All ongoing conditions are required to check.
4. $allowed(S, Obj, ObjR) = true$,

   A prerequisite for UCM_{onC}.
5. $stopped(S, Obj, ObjR) \Leftarrow \neg onC(S, Obj, ObjR)$.

$stopped(S, Obj, ObjR)$ indicates that the access of S on Obj is stopped if ongoing conditions are not satisfied. In practice, the above six models may need to be combined for an access control. The following algorithm is based on these models

and introduces how to manage an ubiquitous computing access control when a user (subject) applies to access an object (Obj) with right $ObjR$.

Authorization Algorithm:
Input: Access request: (S, ObjR, Obj)
Output: result.xml
Method:
// Verify UCM_preA:
1) **if** $preA(ATT(S), ATT(Obj), ObjR) = false$
// The process in pre-Authorization is not successful
2) ACCESS denied;
3) **endif**

// Verify UCM_onA:
4) **if** $preA(ATT(S), ATT(Obj), ObjR) = false$
// The process in pre-Authorization is failed, do not need further verification.
5) Application denied;
6) **endif**
7) $onA(ATT(S), ATT(Obj), ObjR) = false$
// The process in ongoing-Authorization is not successful
8) ACCESS stopped;

// Verify UCM_preB:
9) **if** $preObfill(S, Obj, ObjR) = false$
// Obligations are not fulfilled and pre-Obligation is not passed.
10) ACCESS denied;
11) **endif**

//Verify UCM_onB:
12) **if** $allowed(S, Obj, ObjR) = false$
13) Stop verification.
14) **endif**
15) **if** $onObfill(S, Obj, ObjR) = false$
// Obligations are not continually fulfilled and on-Obligation is not passed.
16) ACCESS is stopped;
17) **endif**

// Verify UCM_preC:
18) **if** $preC(S, Obj, ObjR) = false$
// Conditions are not satisfied and pre-Condition verification is not passed.
19) ACCESS denied;
20) **endif**

//Verify UCM_onC:
21) **if** $allowed(S, Obj, ObjR) = false$
22) Stop verification.
23) **endif**
24) **if** $onC(S, Obj, ObjR) = false$

// Conditions are not continually satisfied and on-Condition is not passed.
25) ACCESS is stopped;
26) **endif**
27) ACCESS Obj is permitted;
28) Output result.xml;

Table 3: Authorization algorithm

We obtain an authorization method for users access objects in the space by checking users' (subjects') authorizations, obligations and conditions with continuity properties. The algorithm provides a solution of the first requirement. We analyse security architectures in the next section for the requirements 3 and 4 in which both client and server sides is required to be monitored.

4 Security architecture

In this section, we discuss architecture solutions for ubiquitous computing access control based on reference monitors. Reference monitors have been discussed extensively in access control community. Subjects can access objects only through the reference monitor since it provides control mechanisms on access space objects such as services and devices in the space.

4.1 Structure of Reference Monitor

ISO has published a standard for access control framework by using reference monitors [5]. Based on the standard, a reference monitor consists of Usage Decision Facility (UDF) and Usage Enforcement Facility (UEF) as shown in Figure 5. Each facility includes several functional modules.

UEF includes *Customization, Monitor and Update modules* and UDF includes *authorization, conditions and obligations decision modules.* When a subject sends an access request through *Customization module* to *Authorization module*, *Authorization module* verifies authorization process and checks whether the request is allowed or not. It may return yes or no or metadata information of the authorization result. This metadata information can be used for approved access on objects by *Customization module* in UEF. *Condition module* is used to make a decision for whether the conditional requirements are satisfied or not. *Obligation module* is applied to verify whether obligations have been performed or not before or during the requested usage. When any obligation is changed, it must be monitored by *monitor module* and the result has to be resolved by *Update module* in UEF. Applications of these modules rely on ubiquitous computing requirements.

4.2 Architectures

There are two kinds of reference monitors: Server-side (or Space-side) Reference Monitor (SRM), and Client-side Reference Monitor (CRM). Servers provide objects

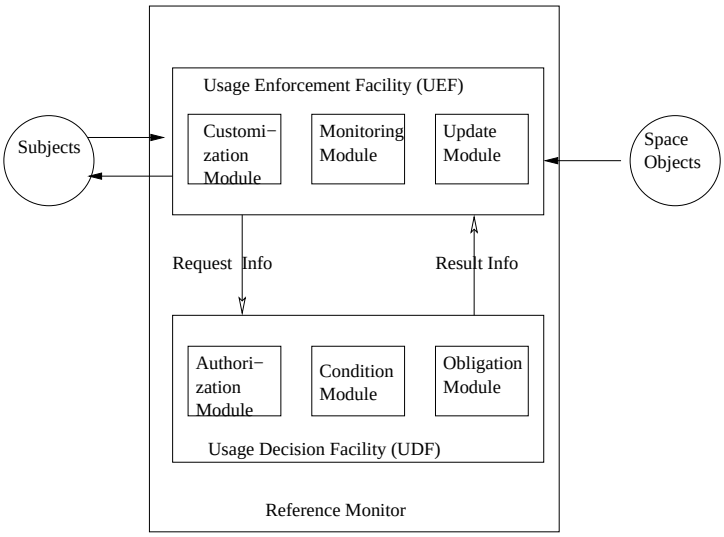

Figure 5: Space Access Reference Monitor

such as services and devices in the space and clients require access to the objects. Like a traditional reference monitor, an SRM works in server space environment and manages access to objects in the space. On the other hand, a CRM works in the client environment and controls access to the objects when it works as a server for other clients. For example, the client acts as a server when a file or a document is delegated or disseminated to other users. SRM and CRM can coexist within a system. For real implementations, both CRM and SRM should be used for better security. We analyse architectures according to reference monitors on space (server) side only (SRM-only), on client side only (CRM-only) and on both server and client sides (SRM & CRM).

SRM-Only Architecture

A system with SRM-only facilitates works on server side only to control subjects access objects. In this case an object may or may not be stored in client-side. If the object (e.g. a file) is allowed to reside in client-side, it means the saved client copy of the object is no longer valid and doesn't have to be controlled. It can be used and changed freely at client-side. For example, a cookie file can be saved at a client's local machine for his records and the server doesnt care how the copy will be used by the client since the system keeps original account information safe. However if the file or some parts of the document has to be protected and controlled centrally, the document must remain at server-side storage and is not allowed to be stored in client-side. This is the main topics of traditional access control and trust management system.

CRM-Only Architecture

No reference monitor exists on the server-side in a system with CRM-only environment. Rather, a reference monitor exists at the client system for controlling usage of delegated or disseminated objects. In this environment objects can be stored either centrally or locally. The usage of the objects saved at the client-side on behalf of a server is still under the control of CRM. Distributed ubiquitous computing environ-

ments are associated with certain usage rules and users may need to prove they have sufficient credentials to access the document.

SRM & CRM Architecture
With both SRM to CRM, this architecture can provide a comprehensive access control. SRM may be used for distribution related control while CRM can be used for object delegation or dissemination. For instance, in SRM, objects can be pre-customized for distribution. The pre-customized objects can be further controlled and customized by CRM. As a result, server can restrict or eliminate unnecessary exposure of objects that do not have to be distributed. If a user requests certain object document that includes some secret information, SRM can pre-customize the requested objects before distribution such that the distributed version of the objects doesn't include any secret information. If the document cannot be delegated or disseminated, the CRM at client side can do this work.

The SRM & CRM architecture provides a solution for restricting access to objects and protecting the objects from malicious delegation and dissemination in collaborative and distributed environments.

5 Comparisons

Related work has been done on Context-Aware Dynamic Access Control for Pervasive Applications [21], Role-based access control in ambient and remote space [19] and Access Control for Active Spaces [11].

Zhang and Parashar [21] proposed a context-based access control model for ubiquitous computing through dynamic role-based access control (DRBAC) model which is an extension of role-based access control (RBAC). DRBAC borrowed the definitions of users, roles, permissions and role hierarchy structure from RBAC, and dynamically adjusts two key requirements based on context information: (1) A users access privileges must change when the users context changes, and (2) A resource must adjust its access permission when its system information changes. The operation of the model was illustrated using a sample application scenario of smart building. Compared to traditional access control mechanisms, the DRBAC model provides improved security for pervasive applications. Their work is different from ours in two aspects. First, it focused on the updating of user-role assignments and permission-role assignments based on the context. Therefore, it only discussed the management in server side and without any management about how to control the object accessed by users. By contrast, our work provides a rich variety of options that can deal with objects in both server and user sides. Second, role-based access control was used in DRBAC which must be combined with feasible authentication mechanisms to secure pervasive applications in the real world. In our scheme, a security architecture has been designed to support the authorization processes such as pre-Authorizations, pre-Obligations and pre-Conditions as well as ongoing-Authorizations, ongoing-Obligations, and ongoing-Conditions.

Role-based access control in ambient and remote space was described by H. Wedde and M. Lischka in 2004 [19]. Their work was based on a distributed and layered

RBAC approach that allows at the same time for 1) more efficient evaluation,2) avoiding evaluation circles, due to the layered trust level and access rules, and 3) more flexibility and more efficient administration. This paper restricts itself to presenting a distributed and location- dependent RBAC approach which is multilayered. The main difference between our scheme and the work in [19] is that we focus on a systematic level for objects in pervasive computing by using usage control model and consider a solution for different kinds of authorizations, whereas the latter is a discussion of providing a secure infrastructure with efficient evaluation and negotiation to objects.

G. Sampemane, P. Naldurg and R. Campbell presented an access control system that automates the creation and enforcement of access control policies for different configurations of an Active Space [11]. The Active Space is a physical space augmented with heterogeneous computing and communication devices along with supporting software infrastructure. The system explicitly recognizes different modes of cooperation between groups of users, and the dependence between physical and virtual aspects of security in Active Spaces. The access model in the system provides support for both discretionary and mandatory access control policies, and uses role-based access control techniques for easy administration of users and permissions. However, our work substantially differs from that access control system. Differences arise in the following three aspects. First, their system is based on role based access control (RBAC) and hence it focuses on permissions-role assignment, objects hierarchies and constrains. By comparison, based on usage control, we have analyzed the characteristics of various access authorizations and presented detailed models for different kinds of authorizations. Second, the approach addresses the problems of Active Spaces in the server side. It does not discuss a secure architecture for objects . By contrast, we have discussed a security architecture for access control by considering both server and client sides. Finally, their approach is based on RBAC, and hence does not mention how to update users' permissions on objects when their conditions or obligations have changed. It is an important state for objects in an Active Space since users always alter their conditions or obligations. By contrast, users in our scheme have to pass pre-Authorizations and ongoing-Authorizations as well as pre-Obligations, pre-Conditions and ongoing-Obligations and ongoing-Conditions. It means our approach is much more powerful in dynamic environment.

6 Conclusions and future work

This paper has discussed access models and architectures for pervasive computing by using usage control. We have analysed not only decision factors in usage control such as authorizations, obligations and conditions, but also the continuity. Different kinds of models are built for ubiquitous computing. To protect objects from malicious delegation and dissemination in collaborative and distributed management, we have analysed reference monitors on both server and client sides and obtained several secure architecture solutions. The work in this paper has significantly extended previous work in several aspects, for example, the ongoing continuity for authorizations, obligations and conditions. These methods can be used to control objects in a

dynamic environment since they provide a robust access control for ubiquitous computing environments and can protect sensitive messages from dissemination. It also begins a new application with usage control.

The future work includes develop algorithms based on the models and architectures proposed in this paper and application of the algorithms in real implementation.

References

[1] Bertino E.,Castano S., Ferrari E. and Mesiti M. Controlled access and dissemination of xml documents. In *Proceedings of the second international workshop on Web information and data management*, pages 22–27. ACM Press, 1999.

[2] Damiani E., Sabrina D., Paraboschi S. and Samarati P. Fine grained access control for soap e-services. In *Proceedings of the tenth international conference on World Wide Web*, pages 504–513. ACM Press, 2001.

[3] Edwards W., Newman M. and Sedivy J. Building the ubiquitous computing user experience. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 501–502, New York, NY, USA, 2001. ACM Press.

[4] Goldschlag D., Reed M., and Syverson P. Onion routing for anonymous and private Internet connections. *Communications of the ACM*, 24(2):39–41, 1999.

[5] ISO. Security frameworks for open systems: Access control framework. Iso/iec 10181-3, 1996.

[6] Jajodia S., Samarati P., Subrahmanian V. and Bertino E. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 474–485. ACM Press, 1997.

[7] JAl-Muhtadi J., Campbell R., Kapadia A., Mickunas M.and Yi S. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 74, Washington, DC, USA, 2002. IEEE Computer Society.

[8] Li Q. and Atluri V. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM workshop on XML security*, pages 94–103. ACM Press, 2003.

[9] Park J. and Sandhu R. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64. ACM Press, 2002.

[10] Sabrina D. An authorization model for temporal xml documents. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 1088–1093. ACM Press, 2002.

[11] Sampemane G., Naldurg P. and Campbell R. Access control for active spaces. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, page 343, Washington, DC, USA, 2002. IEEE Computer Society.

[12] Sandhu R. Role activation hierarchies. In *Third ACM Workshop on RoleBased Access Control*, pages 33–40. ACM Press, October 1998.

[13] Seigneur J. and Jensen C. Trust enhanced ubiquitous payment without too much privacy loss. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1593–1599, New York, NY, USA, 2004. ACM Press.

[14] Todd B. Auditing firewalls: A practical guide. 2004.

[15] Viswanathan P., Gill B. and Campbell R. Security architecture in gaia. Technical report, Champaign, IL, USA, 2001.

[16] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for permission-role assignments using relational algebra operations. In *Proceedings of the 14th Australian Database Conference ADC2003*, Adelaide, Australia, 2003.

[17] Wang H., Li J., Addie R. Dekeyser S. and Watson R. A framework for role-based group delegation in distributed environment. In *Proceedings of the 29th Australasian Computer Science Conference*. Australian Computer Society, 2006.

[18] Wang H., Zhang Y., Cao J. and Varadharajan V. . Achieving secure and flexible m-services through tickets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A, Special issue on M-Services*, pages 697–708, 2003.

[19] Wedde H. and Lischka M. Role-based access control in ambient and remote space. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 21–30, New York, NY, USA, 2004. ACM Press.

[20] Weiser, M. Hot topics-ubiquitous computing. *Computer*, 26(10):71–72, 1993.

[21] Zhang G. and Parashar M. Context-aware dynamic access control for pervasive applications. In *CND'04: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 21–30. Society for Modeling and Simulation International, 2004.