Toward Configurable Access Control for Healthcare Information Systems

Kung Chen[a] and Da-Wei Wang[b]

[a]Department of Computer Science, National Chengchi University
[b]Institute of Information Science, Academia Sinica
[a]chenk@cs.nccu.edu.tw; [b]wdw@iis.sinica.edu.tw

Abstract

This paper examines the necessity and challenges of providing configurable access control for HIS. We discuss both technical issues of system development and non-technical issues specific to Taiwan. We present a framework based on granularity and implementation technology to compare different implementation approaches to access control. The shortcomings in current software practices are identified and aspect-oriented programming is introduced as a promising alternative. We also highlight the directions for moving forward and advocate forming an information technology consortium to pool the resources for developing an advanced solution.

Keywords: access control, aspect-oriented programming, healthcare information systems, electronic health record

1. Introduction

There is little doubt that healthcare information systems (HIS) will move toward a fully integrated electronic health record (HER). However, as we move closer to a paperless environment and Internet-based applications, we must realize that the risks to privacy and security incurred by using electronic systems are also increased. This is a very complicated issue, involving both technical and social aspects. In short, it is a balancing act between cost effectiveness and societal expectation. The healthcare industry wants to migrate to EHR with minimum cost, but the public demands a system that can be trusted without much concern about the price tag. The Health Insurance Accountability and Portability ACT (HIPAA) [3] is the outcome of such a balancing act.

Since access control is an essential part of a secure and privacy enhanced EHR system, we will focus on it in this paper. Right now, the granularity of the privacy protection unit is a debatable topic, and we foresee that society will probably move toward finer grain and the debate will continue. Therefore, it would be more cost effective to build a system which can be easily configured or at least adapted to comply with future security and privacy regulations as well as patient expectations. However, current software development practices are not good at providing the desired flexibility in a modular manner. In contrast, we find that the emerging practice called aspect-oriented programming (AOP) is a very promising technology for developing configurable access control. So we would like to introduce AOP to the HIS community.

In addition to software technology, we must also consider the practical issues in our society to make the move more feasible. Here we present our analysis of the situation in Taiwan and highlight the future directions on enhancing data privacy protection for HIS. Besides, we appeal to the community to support the idea of forming a healthcare information technology consortium to pool the resources for developing an advanced solution.

The remainder of the paper is organized as follows. Section 2 analyzes access control requirements for HIS and presents a framework to discuss the trade-offs between various implementation approaches. AOP is introduced in Section 3 as a software technology for building adaptable and configurable access control mechanisms. Section 4 describes the situation in Taiwan. Section 5 highlights future directions and advocates forming a healthcare IT consortium.

2. Access Control and HIS

2.1 Challenges of Access Control

It is not easy to develop a comprehensive yet flexible mechanism for access control in HIS with EHR. There are at least two major difficulties. First, like other security requirements, access control is a system-wide concern that permeates through all the major modules of a system. Hence it is very often to see the code for implementing access control scattered over the whole system and tangled with other functional code. This is not only error-prone but also makes it difficult to verify its correctness and perform the needed maintenance.

Second, access control rules in healthcare domain are inherently fine-grained and dynamic. It is common for information system developers to partition users into different categories, e.g. by roles in an organization, and define access privileges in terms of the application functions that a particular category of users has a right to access, e.g., an administrative clerk is limited to administrative functions and excluded from transaction functions. For HIS, however, an additional level of access control must be defined at the *data field* level. Users may be allowed to access a specific function, but some data elements may be excluded from view. Furthermore, access may be limited to specific patient records or specific elements within a patient record. For example, while a physician can view some fields of a patient's EHR, only the patient's *attending* physician can see the whole the record and modify it. On the other hand, we will have to bypass the constraint in an emergency. In addition, changes in legislation or changes in the interpretation of legislation can lead to major revision of the access control rules. All these lead to the needs of frequent changes for access control rules in healthcare domain.

An ideal approach to overcome these difficulties is to provide a framework where the access control logic is separated from the core of application and specified declaratively in a configuration file without actual coding, while still being able to satisfy fine-grained access control requirements and incurring a low runtime overhead. However, based on our observation, there is a significant gap between this goal and how access control is achieved in the existing HIS. Here we present a framework for analyzing the gap and, in the following sections, highlight the directions toward realizing this goal.

2.2 Comparing Access Control Approaches

Our framework investigates access control for HIS from two different dimensions. One is granularity which concerns user requirements; the other is implementation technology adopted by system developers.

For access control purpose, the interaction between a user and an information system can be modeled as a sequence of three-tuples: <user, function, data>, indicating a user's request to execute the function on a specific data object. The access control rules determines which tuples are allowed and which must be denied based on the attributes of the three elements in the tuple[1]. Along the lines of thought we divide the granularity of a system's access control into three levels: user-level, function-level, and data-level. User-level granularity allows a user to access anything if he is a legitimate user, e.g., pass the password check. Function-level granularity restricts an authenticated user to only functions permitted by his access privileges regardless of the data contents being accessed. Data-level granularity is the most fine-grained one that also takes the contents of the data to be accessed into consideration when making the decision. As demonstrated above, HIS by nature have to deal with access control at data-level.

As to the implementation technologies for access control,

[1] For example, by treating roles as an attribute of a user, we can easily achieve role-based access control (RBAC) [11] in this model.

we also divide it into three different levels: *hard-wired*, *adaptable*, and *configurable*. Hard-wired implementation means that the code for enforcing access control is scattered through the system and mixed with other functional code. This is quite common in existing systems since security related code is often an afterthought in current practice. In contrast, both adaptable and configurable implementations require that the access control code is properly modularized and can be adapted to new requirements with little efforts. The key difference between them is that a configurable implementation enables us to set the access control rules in a non-programming language, such as XML, and afterwards revises only the rules to get a different access control setting [13]; while in a highly adaptable implementation, we still need to look into the source code to make the necessary changes.

By combining these two dimensions, we get nine different configurations of granularity and implementation technology for access control. These configurations span a wide spectrum that covers existing HIS, security services of modern application platforms, emerging application development technology, and the ideal approach. Figure 1 shows the relative positions of these approaches based on this framework. Most existing HIS fall in the squares of hard-wired implementation while some offer data-level granularity. Few of them have a certain degree of adaptability or even configurability geared to function-level using technology such as J2EE.

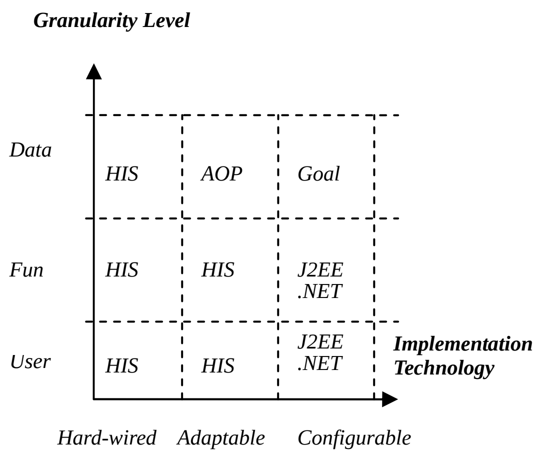

Figure 1: Access Control Classification

J2EE and .NET are two modern platforms that provide configurable security services such as authentication and access control. For example, using the Java Authentication and Authorization Services (JAAS) provided by the J2EE compliant JBoss application server [5], we can specify access permissions in an XML configuration file deployed with the application and get them enforced at runtime by the server. However, these permissions are associated with the methods of objects in the application and thus provide only function-level granularity. To get data-level granularity, we still need to code the control programmatically; no configurable tools are available.

The new application development practice, called aspect-oriented programming (AOP), shows much potential to aid the development of highly adaptable systems with data-level granularity. In addition, we anticipate that the configurable access control for data-level granularity can also be achieved through aspect-oriented technologies and other tools. We will discuss this in the next section.

3. AOP and Access Control

As described earlier, access control is a security concern that cross-cuts all the modules in an application and is therefore difficult to modularize using current programming methods. Indeed, security requirements are best addressed in four different facets: *what, how, where and when* [15], yet implementations using the state-of-the-art OO techniques are successful only to the degree of factoring out the what and the how parts. For example, we may use an authorization engine (how) to enforce role-based access control (what). However, code regarding where in the application access control should be enforced or when this check can be bypassed is still deeply embedded in the application's core. This is because we still need to make calls to the required security libraries within the application to enforce access control.

AOP is a new programming paradigm to support separation of concerns in software development [6]. It addresses the where and when issues of a crosscutting concern through a new kind of modules, called *aspect*, and new ways of module composition. In AOP, a program consists of many functional modules, e.g. classes in OOP, and some aspects that captures concerns that cross-cuts the functional modules, e.g. security. The complete program is derived by some novel ways of composing functional modules and aspects. This is called *weaving* in AOP[2]. Weaving results in a program where the functional modules impacted by the concern represented by the aspect are modified accordingly. Figure 2 illustrates the weaving process. In aspect-oriented languages such as AspectJ [7] and AspectC++ [12], the weaver tool is tightly integrated into the compiler and performs the weaving during compilation[3].

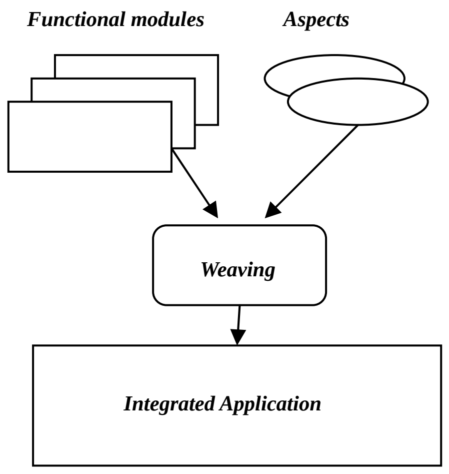

Figure 2: Aspect Weaving in AOP

To facilitate the weaving process, a set of program *joinpoints* are introduced to specify where an aspect may cross-cut the other functional modules in an application. Typical joinpoints in an OO-based aspect-oriented programming language are such as method invocation and field access. A set of joinpoints related by a specific concern are collected into a *pointcut.* Code units called *advices* in an aspect are tagged with a pointcut and determine how the application should behave in those crosscutting points. A single piece of advice can be woven into multiple modules of an application through a pointcut and thus implement a crosscutting concern. In addition, when activated, an advice can obtain comprehensive information about the application at the joinpoints to take the required action.

From the description above, it is clear that AOP lays a very good foundation for developing highly adaptable information systems with fine-grained access control. The basic idea is as follows. Put the code for enforcing access control rules in the advices of security aspects, and use pointcuts to specify where and when in the application access control check must be performed, usually the points around executing the functions to be protected. At runtime, as the activated advice is equipped with complete information about the attempted access and the application state, it will be able to make the proper decisions for even data-level access control. Now as the code for enforcing the access control rules is centralized and properly encapsulated, it is obvious that adjusting the access control rules will require very little efforts to implement.

Adaptability is definitely a good feature to expect from any information systems. Yet, given the dynamic nature of access control rules in HIS, it is highly desirable to take a step forward and make HIS configurable for access control. Based on the aspect-oriented technology and recent research results on access control policies [2, 10], we believe that it is feasible to develop a general framework that can support configurable access control policies for HIS at the data-level. In fact, we are experimenting with an aspect-oriented application architecture that support access control policies in the form of <*user_role, function, data, constraint*>, where the constraint is a first-order formula over the attributes of user, function and data, and some environment

[2] Another approach is based on the notion of interceptor. Recently announced JBoss AOP [4] is a typical example in this category.
[3] Since version 1.1, AspectJ also supports byte-code weaving.

parameters such as time and location of the access. The goal is to explore the acceptable balance between policy expressiveness and implementation complexity.

4. Situation in Taiwan

In Taiwan, electronic health information security and privacy protection regulation can be traced back to 1995, when the Legislator Yuan enacted "Computer-Processed Personal Data Protection Law" (電腦處理個人資料保護法). Based on it, the Department of Health (DOH) issued an order asking all the hospitals, which use computer to process personal information, to submit a document regarding how the personal information would be protected. However, due to various regulations hospitals are still required to keep paper-based patient records. Recently, following the legislation of the Electronic Signature Act (電子簽章法), DOH has revised the Medical Care Act (醫療法) to pave the way for a paperless environment.

Besides, anticipating the trend of privacy concern, DOH has also sponsored quite a few pilot studies to investigate the sensible regulatory framework and possible implementations. However, considering the market scale and IT-maturity of Taiwan's healthcare industry, we suspect that enacting new legislation alone would produce the thrust we observed from HIPAA.

First, according to our prior study [1], most hospitals are willing to allocate only a small fraction of their information technology (IT) budget on improving information security: around one million NTD for medical centers and area hospitals and a quarter of million NTD for local hospitals. They tend to take a reactive position to wait for regulations to be finalized. As for the local HIS vendors, from our observation, most of them are struggling for stay profitable, and are weak in IT innovation. Hence, not seeing any market opportunities, it is very unlikely that they would invest more resources to master the new technologies and improve their systems' security on their own initiative.

Therefore, the initial thrust must come from the security regulations for hospitals that want to go paperless. But it could be a difficult problem for policy makers. On one hand it is very risky to have a less strict regulation, but on the other hand a very demanding regulation might slow down or even stop hospitals moving toward more efficient health information system. Worse yet, as Taiwan is a rapid developing society, the norm of the mainstream value moves very fast. It is conceivable that the regulation will be revised from time to time to reflect the societal norm at that time. It will be a disaster if every regulatory change will cause a major revision of information systems. Nevertheless, we would like to argue that with the help of new software technology and the creation of a mutually beneficial consortium, the future could be brighter.

5. Moving Forward

Indeed, facing the growing concerns over privacy for personal health information, it is inevitable that we have to enhance personal privacy protection for HIS. Instead of taking this as requirements, we can take this as an opportunity to really boost our technical competence in HIS development. There are many forward-looking information standards recently developed in the healthcare arena, most notably the XML-based HL7 3.0, Reference Information Model (RIM) and Clinical Document Architecture (CDA). Basing future HIS on these solid foundations and the promising aspect-oriented technology, it is very likely that we can make a quantum leap and reach a leading position in HIS.

For example, conventional access control rules are established from an organization's concern. But, as we enter the era of EHR, it is a matter of time that access control will move toward *consent-based:* patients themselves will be able to define the policies that control third-party access to their personal health information [9]. This is another big challenge for HIS development. We believe that, with standards such as RIM and CDA, supporting consent-based access control policies will be a more tractable task. Yet these new technologies and new standards are highly sophisticated and can not be assimilated by a single healthcare organization in Taiwan. Therefore we strongly advocate forming a healthcare IT consortium to pool the resources from the industry, government and academia for developing an advanced solution.

Having argued intensively for the ideal side, we should also say a few words on practicality. Admittedly, there are still many legacy systems out there functioning well for daily operations and are unlikely to be replaced soon. We need to investigate technologies for reengineering security into these legacy systems. Here AOP is also worth further exploration. We have seen some positive preliminary results [8, 14] published recently on this subject using aspect-oriented technology. The proposed technical consortium should also take the development of security reengineering methodology as one of its mission.

5. References

[1] 蔡佳婷, 王大為, 郭旭崧 (2001)，醫療院所高階主管對醫療資訊安全與隱私看法的調查研究，國際醫學資訊研討會，台灣，桃園。

[2] R. Chandramouli (2001), “A Framework for Multiple Authorization Types in a Healthcare Application System,” 17th Annual Computer Security Applications Conference, New Orleans, Louisiana.

[3] HIPAA. http://www.cms.hhs.gov/hipaa.

[4] JBoss AOP. http://www.jboss.org/products/aop

[5] JBoss Security - J2EE Security Configuration and Architecture, Chapter. 8 of JBoss Administration and Development, 3rd Edition, http://docs.jboss.org/admin-devel/Chap8.html

[6] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin (1997), “Aspect-Oriented Programming,” in ECOOP '97, LNCS 1241, pp. 220-242.

[7] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W.G. Griswold (2001), “Getting Started with AspectJ”, Communication of ACM, vol. 44, no. 10, pp 59-65.

[8] R. C. Laney, J. van der Linden, P. Thomas (2003), “Evolving Legacy System Security Concerns Using Aspects,” Technical Report Number 2003/13, Department of Computing, The Open University, U.K http://computing-reports.open.ac.uk/index.php/content/download/82/322/file/2003_13.pdf

[9] J. Reid, I. Cheong, M. Henericksen, and J. Smith (2003), “A Novel Use of RBAC to Protect Privacy in Distributed Health Care Information System,” Proc. the 8th Australasian Conference on Information Security and Privacy, ACISP 2003, LNCS 2727, pp. 403-415, 2003.

[10] P. Samarati and S. Vimercati (2001), “Access control: policies, models and mechanisms,” Foundations of Security Analysis and Design, LNCS 2171, pp. 137–196. Springer Verlag.

[11] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman (1996), “Role-based access control models,” IEEE Computer, 29(2):38–47, 1996.

[12] O. Spinczyk, A. Gal, W. Schroder-Preikschat (2002), “AspectC++: An Aspect-Oriented Extension to C++,” Proceedings of the 40th Int’l Conf. on Technology of Object-Oriented Languages and Systems, Sydney, Australia, Feb. 18-21, 2002.

[13] T. Verhanneman, L. Jaco, B. De Win, F. Piessens, and W. Joosen (2003), “Adaptable Access Control Policies for Medical Information Systems,” Proc. of Distributed Applications and Interoperable Systems, 2003, Paris, France, LNCS 2893, pp. 133-140.

[14] I. S. Welch and R. J. Stroud (2003), “Re-engineering Security as a Crosscutting Concern,” The Computer Journal, Vol. 46, No. 5, Sept. 2003: pp. 578-589.

[15] B. De Win, F. Piessens, W. Joosen and T. Verhanneman (2002), “On the importance of the separation-of-concerns principle in secure software engineering,” ACSA Workshop on the Application of Engineering Principles to System Security Design, pp. 1-10, Boston, Massachusetts.