The Secure Blackboard pattern [*]

Jorge L. Ortega-Arjona
Departamento de Matemáticas
Facultad de Ciencias, UNAM, México
jloa@fciencias.unam.mx

Eduardo B. Fernandez
Dept. of Computer Science and Eng.
Florida Atlantic University
Boca Raton, FL 33431, USA
ed@cse.fau.edu

ABSTRACT

This paper presents the Secure Blackboard pattern as a secure version of the original Blackboard pattern and Shared Resource pattern. The Blackboard pattern is an architectural pattern useful for problems for which no deterministic solution strategies are known. Several specialized subsystems or components "assemble" their knowledge to build a possibly partial or approximate solution, coordinated by a central controller. On the other hand, the Shared Resource pattern is a specialization of the Blackboard pattern, in which subsystems or components are allowed to perform simultaneous computations without a prescribed order on different, ordered data. The Secure Blackboard pattern includes ways to add security at the control component, providing secure handling of data, as well as controlling data transformation and movement.

Categories and Subject Descriptors

D.2.11 [**Software architectures: patterns**]: Design—*Security*; D.1.3 [**Concurrent Programming**]: Distributed programming—*Security*

General Terms

Security Patterns

Keywords

Software Pattern, Security Pattern, Software Architecture, Blackboard

[*]Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission. Preliminary versions of these papers were presented in a writers' workshop at the 15th Conference on Pattern Languages of Programs (PLoP). PLoP'08, October 18-20, Nashville, TN, USA. Copyright 2008 is held by the author(s). ACM 978-1-60558-151-4

1. INTRODUCTION

Many applications, such as problem solving, image processing, feedback control, or even some web applications, are developed based on a central information repository, composed of *(a)* a *blackboard*, which contains a centralized data structure, *(b)* several independent components, known as *knowledge sources*, which are capable of reading, processing, and updating the data elements of the blackboard, and *(c)* a *control*, which is in charge of monitoring the blackboard and synchronizing the access of the knowledge sources. This organization is used due to several reasons: every knowledge source performs specialized functions over the data, the global architecture or organization requires a centralized control so that state and consistency can be assured, or the whole system performs its functionality in a more efficient and flexible way. Operations on the data do not have a fixed precedence, that is, operations can be carried out in any order, and coordinated by the central controller.

The organization of this process has been well defined and converted into a pattern: the Blackboard pattern (Figure 1) [1] and its parallel counterpart, the Shared Resource pattern [8]. Both descriptions provided for these patterns take into consideration only functional properties, such as their potential for improving performance [8]. They have been proposed assuming that all components (blackboard, control, and knowledge sources) "implicitly trust" each other, and there is no means of any unwanted activity among them. However, many distributed applications (such as those mentioned earlier) require to take into consideration security, since data sources may handle sensitive or valuable data, such as personal or credit card information.

Security patterns are relatively new and starting to be accepted by industry because they are useful to guide the security design of systems by providing generic solutions that can prevent a variety of attacks [9]. These patterns describe how to take into consideration security during the analysis and design of systems. In this paper, we introduce the Secure Blackboard pattern as a secure version of the original Blackboard pattern and the Shared Resource pattern. This pattern includes ways to add security controls to the components, providing secure handling of data, as well as controlling data reading and updating. This pattern is part of an ongoing effort to catalog and provide a variety of security mechanisms for different architectural levels. By itself, this catalog has its own value (its patterns can be used in isolation). However, it is also part of a security systems

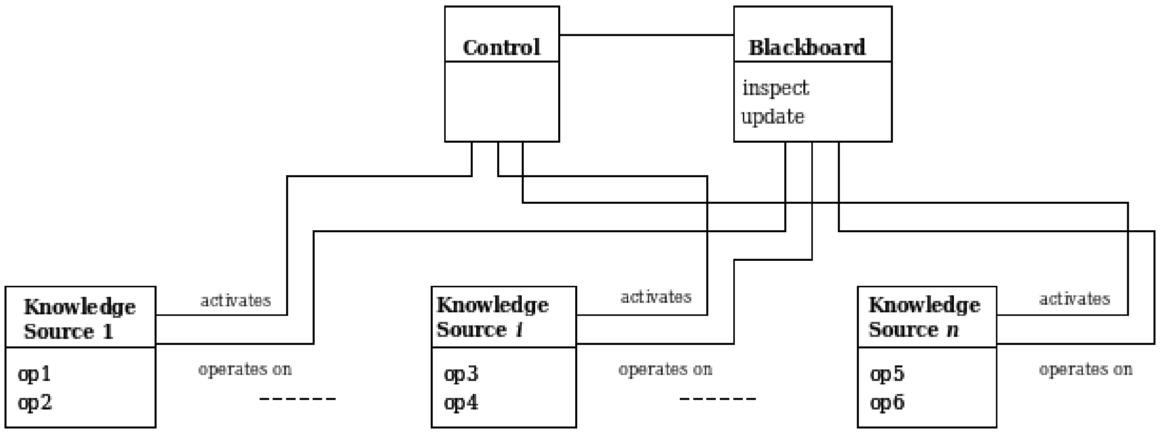

Figure 1: UML Object Diagram of the Blackboard pattern

development methodology we have proposed [5].

2. THE SECURE BLACKBOARD PATTERN

The Secure Blackboard pattern provides secure handling of data when its blackboard is accessed by the knowledge sources. Each knowledge source reads data from the blackboard, applies some processing or data transformation, and updates the blackboard. In order to prevent violations of integrity and confidentiality, the rights to reading and updating data should be controlled and their actions should be logged. The sources also must be authenticated before being allowed to access the blackboard.

2.1 Example

Suppose we are developing an application for a law firm [3]. The conduction of a case require inputs from many data sources: lawyers, witnesses, defendants, etc. Court appearances are scheduled according to court and lawyer availability. All this makes the sequence of actions rather unpredictable. The variety of inputs requires coordination. The data handles is very sensitive and needs to be controlled. If we are not careful we might end up with invalid data which will hurt our chances of winning the case. There can be also data misuses such as a clerk selling the information of a case to the opponent party.

2.2 Context

A Blackboard system is used to receive and modify information about a problem in progress from several data sources. The execution platform for this kind of system is normally distributed, with knowledge sources possibly remote. The data is exchanged between blackboard and knowledge sources in a client/server fashion.

2.3 Problem

How to let the knowledge sources access the blackboard while keeping an acceptable level of security for the whole system? A software system based on the Blackboard pattern is suited to process data as a shared resource among the knowledge sources [1, 8]. This has several reasons: every component may perform specialized functions over the data, the global architecture or organization is the simplest and easiest to develop, requiring only devise knowledge sources that read and update the blackboard, and this approach makes the whole system more efficient and flexible.

The essence of the Blackboard pattern is that every time data is processed, its is done so outside the data structure of the blackboard: that is, within the knowledge sources where different functions are applied on it (see Figure 1). In the previous example, the personnel of the law firm are the only ones who should have access to the blackboard. . Notice that in this kind of system, we have the flexibility to take any order of steps of the process or change the processing steps. Nevertheless, how do we control the actions to be performed in the blackboard?

This problem requires considering the following forces:

- The blackboard itself is shared data and how it is used is very important for security. We may end up with bad information or leaking information.
- Data should only be created by authorized knowledge sources, and its use (reading or modification) should also be controlled. The system needs to assign privileges according to the knowledge source and the functions of the users of these sources.
- It might be necessary to verify the source of the data is authentic. Otherwise we might receive false information or our information could be leaked outside the system.
- Due to regulatory constraints, work changes, or efficiency, we need to be able to reconfigure the number of knowledge sources or their order of operation. This reconfiguration must be controlled.
- For billing and security purposes, logging the actions at each updating may be necessary.
- The security controls should be transparent to the users of the system or they would not use them.

2.4 Solution

The Secure Blackboard pattern provides a secure way to access blackboard data from a variety of knowledge sources, by adding to the control component some basic security mechanisms (as instances of security patterns), providing authentication (Authenticator), authorization (Role-Based Access Control, RBAC), and logging Secure Logger) in each access operation.

2.4.1 Structure

Figure 2 shows an UML Class Diagram of the Secure Blackboard pattern, in which security pattern instances have been added to the components of the Blackboard pattern. **Secure Logger** indicates an instance of the Secure Logger pattern [10]. The **Reference Monitor** associated with the control indicates the enforcement of authorization [9]. **Knowledge Sources** can be not only software components, but also humans. Nevertheless, either automated or human **Knowledge Sources** require that their access is authenticated by the **Authenticator** [9] in the control to verify their origin. The sources belong to **Roles**, according to their functions, and their rights depend on these roles.

2.4.2 Dynamics

Figure 3 shows a UML Sequence Diagram in which a **Knowledge Source** (with a specific role) requests an operation on the **Blackboard**. The **Control** receives the request and invokes the Autheticator to validate that it proceeds from a legitimate source. After source validation, the **Reference Monitor** checks if its role is allowed this operation and, if true it performs the operation on the **Blackboard**. A **Secure Logger** record is created after the operation is performed.

2.5 Implementation

[1] list several general implementation aspects. From a security point of view we need to consider:

- The authentication system should be appropriate to the value of the information handled.
- Instead of RBAC we could use an access matrix or even a multilevel access control model [7], depending on the environment.
- Since the repository and its control are centralized applying the proposed security functions is relatively simple.

2.6 Example Resolved

The law firm now uses a Secure Blackboard structure to conduct its cases. The case blackboard receives changes for the case documents which get stored in specific classes. The case blackboard can be protected from illegal access. We can also verify that new information is authentic.

2.7 Known Uses

- The software system used by many news agencies (such as AP, AFP, or Reuters) has a structure like the Secure Blackboard pattern. All information retrieved by reporters and correspondents (articles, editorials, notes, photographs, an so on) is gathered into a single blackboard, which at the same time is read by many other news and media enterprises (newspapers, television, radio, etc.), who distribute the information. Nevertheless, all the information written to or read from the database should be secure. This means, nobody should be allowed to modify the database unless she logs in and authenticates in order to have the right of writing. In a similar way, the blackboard can be read as long as the news and media enterprises are allowed to do so.
- The Automatic Teller Machines (ATM) of any bank or credit institution require having a similar organization as the Secure Blackboard pattern. The credit or savings information of all the bankąÇs or credit institutionąÇs customers flows every day from the ATMs to a central database, which requires to be protected from corruption, or be modified with the purpose of stealing. So, every time a customer performs an operation on an ATM, and before allowing any change to or consult for the information of the database, the ATM system requires to take appropriate security measures in order to prevent any corruption or misuse.
- A Wiki web is also an example of the use of the Secure Blackboard pattern. In such a case, knowledge sources are actually humans, whose role within the Wiki could be "reader", "editor", or "admin". The Wiki should actually function like a blackboard, whose secure use requires that users are always authenticated, and access is controlled according to their roles within the Wiki system.
- Two designs for applications that may use this pattern include a travel booking system [12], a law firm [3], and a Java-based knowledge processing and agent programming software framework [11].
- The Reflective Blackboard pattern [2] includes security services.

2.8 Consequences

The Secure Blackboard pattern shares the same benefits considered for the original Blackboard pattern [1], having the following additional advantages:

- We can define precise role rights, e.g. an expert can only add to the information, not change it.
- The access control mechanism enforces controlled access to the information.
- Authentication services can validate that the data sources are legitimate.
- We can reconfigure the data sources to comply with regulations or other reasons. This reconfiguration can be done only by authorized people.
- We can log accesses to the blackboard for future auditing.

Similarly, the Secure Blackboard pattern shares the same liabilities of the original Blackboard pattern [1], and also:

- Adding security capabilities affects the response time of the whole Blackboard system.

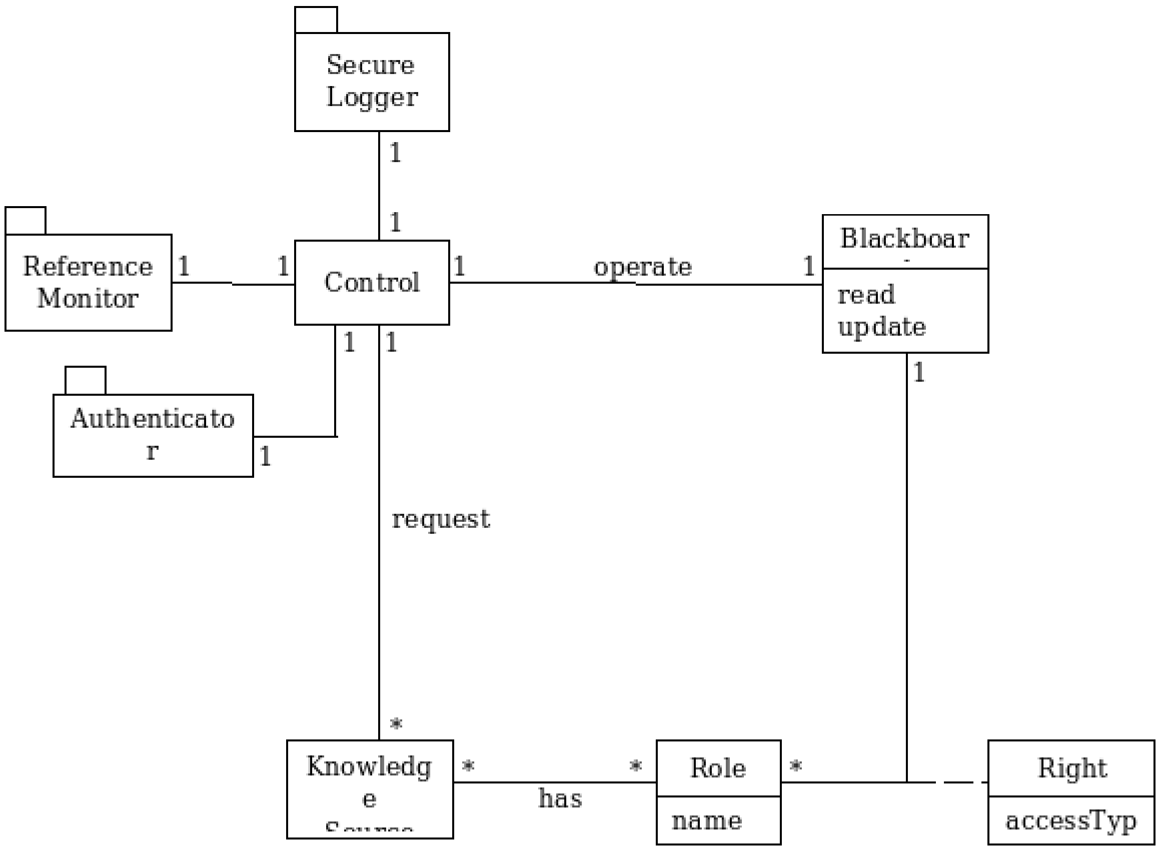

Figure 2: UML Object Diagram for the Secure Blackboard pattern

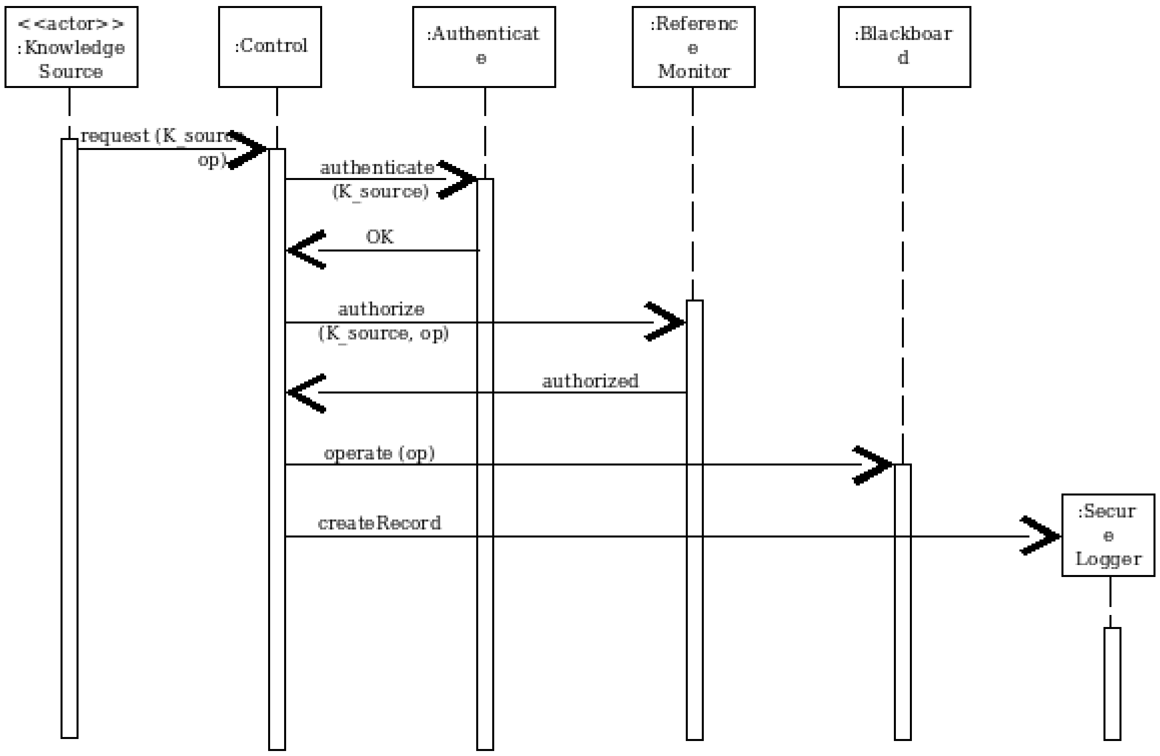

Figure 3: UML Sequence Diagram for use case to apply an operation on data

- Even though normally the implementation of the Blackboard pattern requires to develop the blackboard, the control, and the knowledge sources as simple, loosely connected components, when adding security capabilities a more complex implementation is required. Several software components, such as the Reference Monitor, the Authenticator, the Log, Role, and Right components should be taken into consideration in order to have a correct functionality.

2.9 See Also

- The Blackboard pattern [1] is the basis for this pattern.
- Assignment of knowledge sources can use the Constrained Resource Assignment Description pattern [6].
- The rights structure can follow an RBAC pattern [9].
- Authentication is performed by means of instances of the Authenticator pattern [9].
- Logging can be done using a Secure Logger [10].
- The Control acts as a Reference Monitor [9].

3. CONCLUSIONS

The use of blackboards is frequent in software design. Following the principle that security must be applied in all stages of the software development, the designer should include security aspects in any application. As mentioned earlier, this pattern will become part of a catalog of security patterns. Combined with other similar patterns, it gives a designer a choice of possibilities when building the middleware of a complex system [4]. Future work includes developing secure versions of the Adapter pattern [1], a pattern frequently used together with this pattern. Implementing these patterns together in a real application would be another useful direction that would confirm the value of using patterns; specifically implementing the law office example using web resources.

4. ACKNOWLEDGMENTS

Our shepherd, Peter Sommerlad, provided valuable comments that helped improve this paper. The workshop participants at PLoP 2008 provided valuable comments.

5. REFERENCES

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley and Sons, West Sussex, England, 1996.

[2] O. R. da Silva, A. F. Garcia, and C. J. P. de Lucena. The reflective blackboard architectural pattern. In *Rept. PUC-Rio Inf. MCC24/02*, September 2002.

[3] E. B. Fernandez, D. L. laRed, J.Forneron, V. E. Uribe, and G. Rodriguez. A secure analysis pattern for handling legal cases. In *6th Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP'2007)*, 2007.

[4] E. B. Fernandez and M. M. Larrondo-Petrie. Developing secure architectures for middleware systems. In *Procs. of CLEI 2006. (XXXII Conferencia Latinoamericana de Informática)*, 2006.

[5] E. B. Fernandez, M. M. Larrondo-Petrie, T. Sorgente, and M. VanHilst. A methodology to develop secure systems using patterns. In *Integrating security and software engineering: Advances and future vision*, pages 107–126. H. Mouratidis and P. Giorgini (Eds.), IDEA Press, 2006.

[6] E. B. Fernandez, T. Sorgente, and M. VanHilst. Constrained resource assignment description pattern. In *Proceedings of the Nordic Conference on Pattern Languages of Programs, Viking PLoP 2005*, pages 23–25. Otaniemi, Finland, September 2005.

[7] D. Gollmann. *Computer security (2nd Ed.)*. John Wiley and Sons, West Sussex, England, 2006.

[8] J. L. Ortega-Arjona. The shared resource pattern. an activity parallelism architectural pattern for parallel programming. In *Procs.of the Conference on Pattern Languages of Programs (PLoP 2003)*, 2003.

[9] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns: Integrating security and systems engineering.* John Wiley and Sons, West Sussex, England, 2006.

[10] C. Steel, R. Nagappan, and R. Lai. *Core Security Patterns: Best Strategies for J2EE Web Services and Identity Management.* Prentice Hall, Upper Saddle River, New Jersey, 2005.

[11] P. Tarau. Object oriented logic programming as an agent building infrastructure. In *http://logic.csci.unt.edu/tarau/research/slides/oolpAgents.ppt*, October 2002.

[12] E. Tempero. Notes for softeng 325: Software architecture, lecture 11. In *http://www.se.auckland.ac.nz.*

APPENDIX

Authenticator [9]. How to verify that a subject is who it says it is? Use a single point of access to receive the interactions of a subject with the system and apply a protocol to verify the identity of the subject.

Role-Based Access Control (RBAC) [9]. How do we assign rights to people based on their functions or tasks? Assign people to roles and give rights to these roles so they can perform their tasks.

Secure Logger [10]. Defines how to capture the application-specific events and exceptions in a secure and reliable manner to support security auditing.

Constrained Resource Assignment Description Pattern [6]. Once resources have been allocated in some way to functional units, this pattern describes the assignment, including resource types, constraints, roles, and other descriptions.

Reference Monitor [9]. In a computational environment in which users or processes make requests for data or resources, this pattern enforces declared access restrictions when an active entity requests resources. It describes how to define an abstract process that intercepts all requests for resources and checks them for compliance with authorizations.