The Secure Blackboard pattern

Jorge L. Ortega-Arjona[1] and Eduardo B. Fernandez [2]
[1] Departamento de Matemáticas, Facultad de Ciencias, UNAM. México.
[2] Dept. of Computer Science and Eng. Florida Atlantic University, Boca Raton, FL 33431, USA

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.

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 other independent components, known as *knowledge sources*, which are capable of reading, processing, and updating the data elements of the blackboard, and *(c)* the *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 needs to perform 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, coordinated by the central controller.

The organization of this process has been well defined and converted into patterns: the Blackboard pattern (Figure 1) [Bus96], and its parallel counterpart, the Shared Resource pattern [Ort03]. The descriptions provided for these patterns take into consideration only functional properties, such as their potential for improving performance [Ort03]. These patterns have been proposed considering that all components (blackboard, control, and knowledge sources) "implicitly trust" each other, and there is no concern about unwanted activity among them. However, many distributed applications (such as those mentioned earlier) require taking into consideration security, since data sources may handle sensitive or valuable data, such as personal or business 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 [Sch06]. 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 and Shared Resource patterns. This pattern includes ways to add security controls at the control components, providing secure handling of data, i.e., 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 development methodology we have proposed [Fer06a].

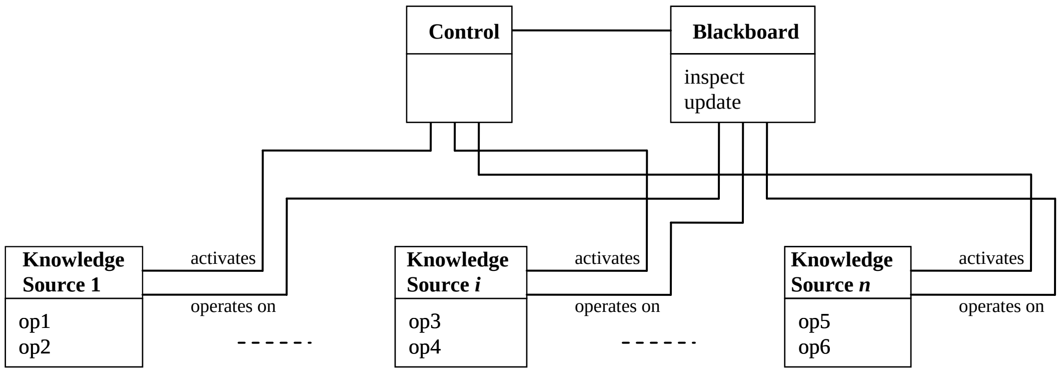

Figure 1. UML Object Diagram of the Blackboard pattern

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 according to their predefined 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 [Fer07]. The conduction of a case requires 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. A blackboard is used to conduct a case, where immediate results of court appearances and case strategy are kept for analysis and updating by lawyers. The data handled is very sensitive and access to it needs to be controlled. If we are not careful we might end up with invalid data or data will leak to our opponent, which will hurt our chances of winning the case.

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

The essence of the Blackboard pattern is that every time data is processed, this happens 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 so we provide a good level of security to the system?

This problem requires considering the following forces:

- The sequence of activities or operations over data is usually unpredictable. Also, the number of knowledge sources might be hard to predict. Knowledge sources can be added or removed dynamically.
- Blackboard data can only be read or modified by authorized knowledge sources. The system needs to assign privileges according to the function of the knowledge source.
- It might be necessary to verify that the source of the data is authentic. Otherwise, we might receive false information or our information could be leaked outside our 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 way to access blackboard data from a variety of knowledge sources in a secure way 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.

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 [Ste05]. The **Reference Monitor** associated with the control indicates the enforcement of authorization [Sch06]. **Knowledge Source**s can be not only software components, but also humans. Nevertheless, either automated or human **Knowledge Source**s require that their access is authenticated by the **Authenticator** [Sch06] in the control to verify their origin. The sources belong to **Roles**, according to their functions, and their rights depend on these roles.

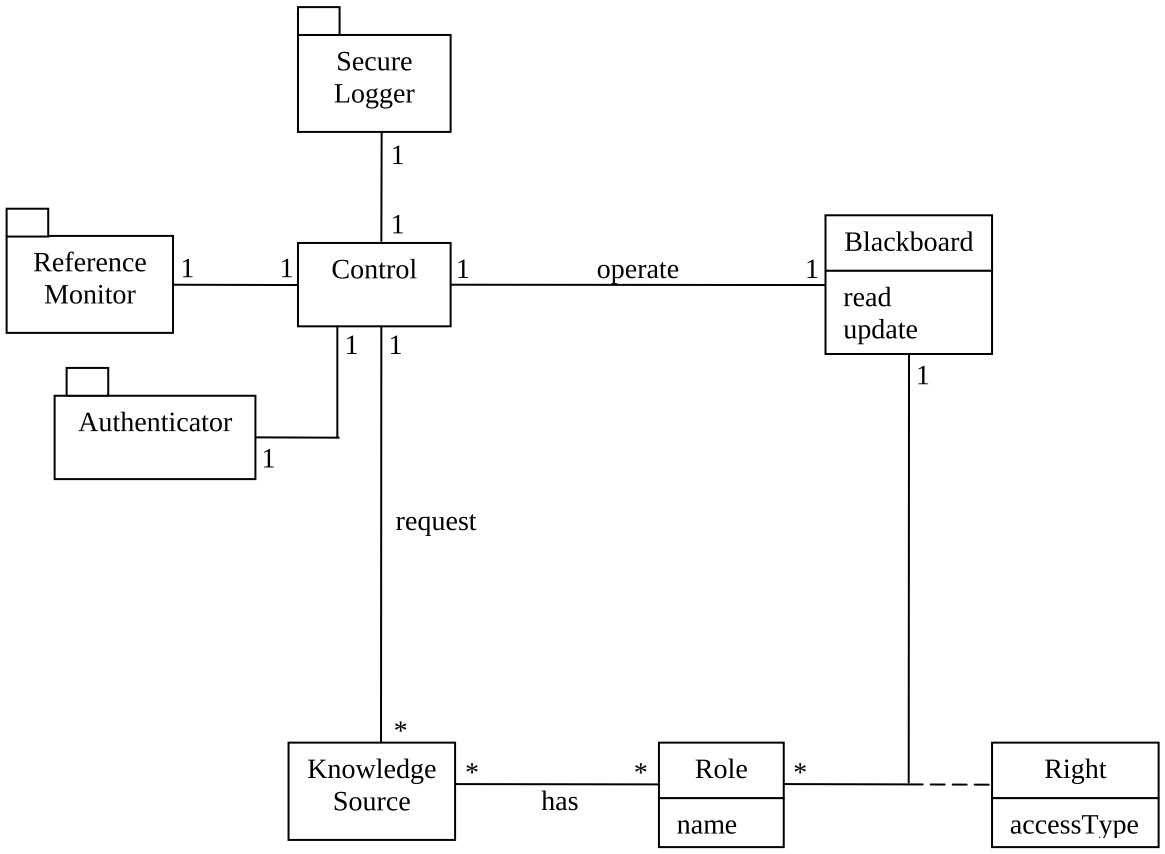

Figure 2. UML Class Diagram for the Secure Blackboard pattern

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.

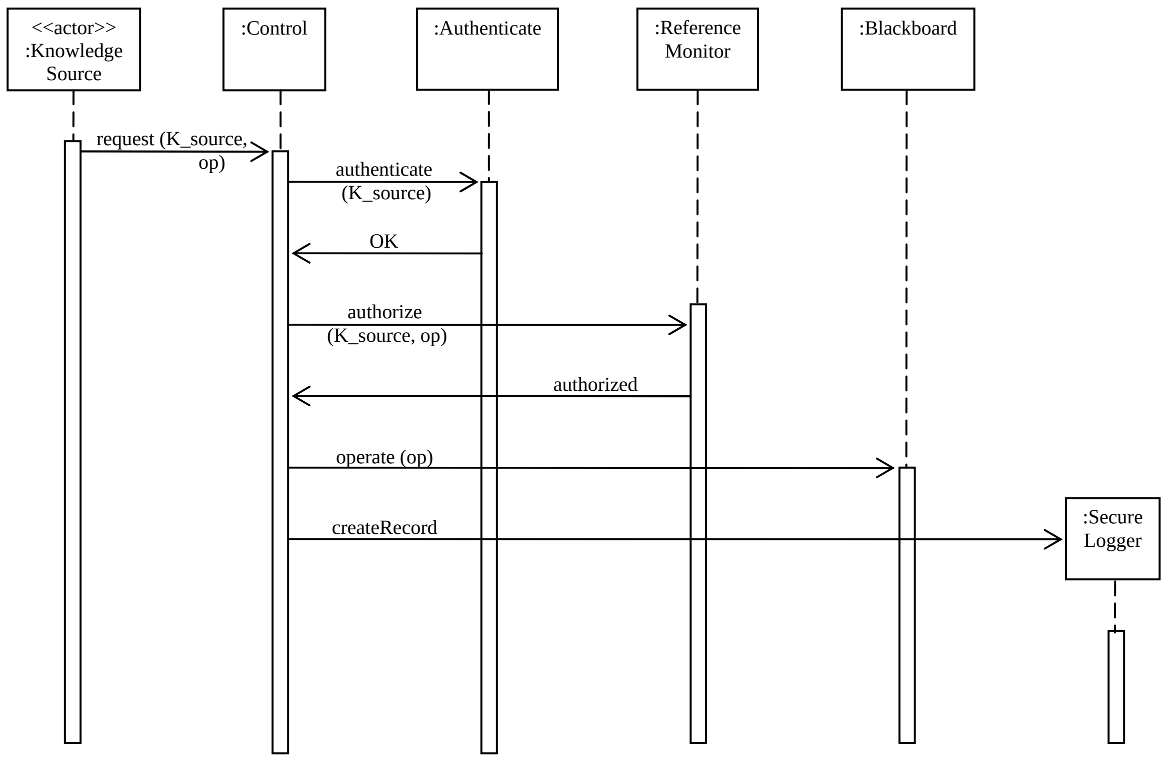

Figure 3. UML Sequence Diagram for use case apply an operation on the Blackboard

2.5 Implementation

[Bus96] 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 [Gol06], 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 unauthorized access.

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 blackboard should be secure. This means that nobody should be allowed to modify or read the blackboard unless authentication and authorization are applied.

- 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 bank's or credit institution's customers flows every day from the ATMs to a central database, which requires to be protected from corruption. So, every time a customer performs an operation on an ATM, and before allowing any change to or consult the information of the database, the ATM system requires to take appropriate security measures in order to prevent any 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 [Tem], a law firm [Fer07], and a Java-based knowledge processing and agent programming software framework [Tar02].
- The Reflective Blackboard pattern [Sil02] includes security services..

2.8 Consequences

The Secure Blackboard pattern shares the same benefits considered for the original Blackboard pattern [Bus96], having the following security advantages:

- We can define precise role rights, e.g. an expert can only add to the information, not change it, a lawyer can decide on the next step, bring new witnesses, but cannot change depositions.
- The access control mechanism in the control enforces controlled access to the information.
- Authentication services can validate that the data sources are legitimate.
- We can log accesses to the blackboard for future auditing.

Similarly, the Secure Blackboard pattern shares the same liabilities of the original Blackboard pattern [Bus96], and also:

- Adding security capabilities affects the response time of the whole Blackboard system, although their effect should be small.
- 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 Logger, and Authorization components should be taken into consideration in order to have a correct functionality.
- The three security mechanisms incorporated in this pattern are not enough to control all possible security threats and must be complemented with additional mechanisms according to the needs of the application.

2.9 See Also

- The *Blackboard* pattern [Bus96] is the basis for this pattern.
- Assignment of knowledge sources can use the *Resource Assignment* pattern [Fer05].
- The rights structure can follow an RBAC pattern [Sch06].
- Authentication is performed by means of instances of the Authenticator pattern [Sch06].
- Logging can be done using a Secure Logger [Ste05].

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 [Fer06b]. Future work includes developing secure versions of the Adapter pattern [Bus96], 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.

Acknowledgements
Our shepherd, Peter Sommerlad, provides valuable comments that helped improve this paper.

References

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns, Volume 1*, West Sussex, England: John Wiley & Sons, 1996.

[Fer05] E.B.Fernandez, T. Sorgente, and M. VanHilst, "Constrained Resource Assignment Description Pattern". *Proceedings of the Nordic Conference on Pattern Languages of Programs, Viking PLoP 2005*, Otaniemi, Finland, 23-25 September 2005.

[Fer06a] E. B. Fernandez, M.M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A methodology to develop secure systems using patterns", Chapter 5 in "*Integrating* *security and software engineering: Advances and future vision"*, H. Mouratidis and P. Giorgini (Eds.), IDEA Press, 2006, 107-126.

[Fer06b] E. B. Fernandez and M. M. Larrondo-Petrie, "Developing secure architectures for middleware systems", *Procs. of CLEI 2006. (XXXII Conferencia Latinoamericana de Informática).*

[Fer07] E. B. Fernandez, D. L. laRed M., J. Forneron, V. E. Uribe, and G. Rodriguez G. "A secure analysis pattern for handling legal cases", accepted for the *6th Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP'2007).*

[Gol06] D. Gollmann, *Computer security (2nd Ed.)*, Wiley, 2006.

[Ort03] J.L. Ortega-Arjona, "The Shared Resource Pattern. An Activity Parallelism Architectural Pattern for Parallel Programming." *Procs.of the Conference on Pattern Languages of Programs (PLoP 2003)*

[Sch06] M. Schumacher, E.B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering"*, Wiley 2006.

[Sil02] O.R. da Silva, A.f. Garcia, and C.j.P. de Lucena, "The Reflective Blackboard architectural pattern", Rept. PUC-Rio Inf. MCC24/02, Sept. 2002.

[Ste05] 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.

[Tar02] P. Tarau, "Object oriented logic programming as an agent building infrastructure", Oct. 2002, http://logic.csci.unt.edu/tarau/research/slides/oolpAgents.ppt

[Tem] E. Tempero, Notes for SOFTENG 325: Software Architecture, Lecture 11, http://www.se.auckland.ac.nz

Appendix

Authenticator.[Sch06]. 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) [Sch06]. 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 [Ste05]. Defines how to capture the application-specific events and exceptions in a secure and reliable manner to support security auditing.