Classifying security patterns

Eduardo B. Fernandez, Hironori Washizaki, and Nobukazu Yoshioka
Dept. of Computer Science and Engineering National Institute of Informatics
Florida Atlantic University Tokyo, Japan
Boca Raton, FL, USA

Abstract
Analysis and design patterns are well established as a convenient and reusable way to build high-quality object-oriented software. Patterns combine experience and good practices to develop basic models that can be used for new designs. Security patterns join the extensive knowledge accumulated about security with the structure provided by patterns to provide guidelines for secure system design and evaluation. In addition to their value for new system design, security patterns are useful to evaluate existing systems. They are also useful to compare security standards and to verify that products comply with some standard. Finally, we have found security patterns very valuable for teaching security concepts and mechanisms. A variety of security patterns has been developed for the construction of secure systems and catalogs of them are appearing. However, catalogs of patterns are not enough because the designer does not know when ot where to apply them. We discuss here several ways to classify patterns. We show a way to use these classifications through pattern diagrams where a designer can navigate in her pattern selection.

1. Introduction
A *pattern* is a packaged reusable solution to a recurrent problem. The appearance of *design patterns* [Gam94] has been one of the most important developments in software engineering. Design patterns embody the experience and knowledge of many designers and when properly catalogued, they provide a repository of solutions for useful problems. They have shown their value in many projects and have been adopted by many institutions. Design patterns have been extended to other aspects of software, first to architectural aspects of design [Bus96], then to the analysis stage [Fow97]. Analysis and design patterns are now well established as a convenient and reusable way to build high-quality object-oriented software. A pattern solves a specific problem in a given context and can be tailored to fit different situations. Analysis patterns can be used to build conceptual models, architectural patterns can build software architectures, design patterns can be used to make software more flexible and extensible.

Security has become an important concern in current systems. The main objectives of security are to protect the confidentiality, integrity, and availability of data. Data is a valuable resource and it has become the target of many attacks by people who hope to gain monetary advantages, make political statements, or just vandalize. All security countermeasures can be classified into five groups: Identification and Authentication, Access Control and Authorization, Logging, Cryptography, and Intrusion Detection. Security patterns describe mechanisms that fall into these categories (or combinations thereof) to stop or mitigate attacks as well as the abstract models that guide the design of these mechanisms. *Security patterns* join the extensive knowledge accumulated about security with the structure provided by patterns to provide guidelines for secure system construction and evaluation. Security has had a long trajectory, resulting in a variety of approaches to analyze security problems and to design security mechanisms. It is natural to try to codify this expertise in the form of patterns. A good number of security patterns have been described in the literature [Fer06a, Sch06, sec, Ste05].

However, it is not enough to have a catalog of security patterns, we also need a guidance for the designers about how to select appropriate patterns. Security should be applied in the whole life cycle of applications and at each stage patterns can be applied to provide specific security controls. A designer who is not a security expert would be lost trying to apply them into her design. Designers need guidance about how to select appropriate patterns. A step in this direction is a good classification of security patterns. We present here some

possible classifications that we have used, based on architectural concerns, architectural layers, or the relationship between patterns. We show the use of these classifications through a pattern diagram, where the designer can navigate in order to select appropriate patterns.

Section 2 discusses general aspects of pattern classification. Section 4 considers the use of security patterns for designing secure systems. Section 5 describes methodologies for secure systems design using patterns. We end with some conclusions.

2. The nature of security patterns?

We consider first in which of the basic pattern types we can classify security patterns. Four possibilities have been considered to define the nature of security patterns. A security pattern can be considered:

- An *architectural* pattern. They usually describe global system architecture concepts, e.g., do we need authentication between two distributed units? We consider this association to be the most convenient because security is a global property of a system.

- A *design* pattern. The fact that security can be considered an aspect of a software subsystem has made some groups consider them design patterns [ope]. However, design patterns are oriented towards code flexibility and do not consider global aspects, necessary for security.

- An *analysis* pattern. Security constraints should be defined at the highest possible level, i.e. at the conceptual model of the application. For example, we can define which users have which roles and what rights they need to perform their duties. This means that at least some security patterns are analysis patterns.

- A *special type* of pattern. We can add new sections or remove some sections from the standard template patterns but we don't see a compelling reason for an entirely new type of pattern.

While interesting, this classification is not very useful to designers, it is of value mostly to pattern writers. In order to use patterns in building systems we need operationally-oriented classifications. We discuss some of them in the following sections.

3. Classification based on architectural concerns

A first idea is that, since we consider security patterns to be architectural patterns, we should look at software architecture classifications. [Avg05] classifies architectural patterns using the type of concerns they address, e.g. Layered Structure, Data Flow, Adaptation, User Interaction, Distribution. This means we should classify security patterns according to their concerns, e.g. secure layers, secure adaptation, or similar [Fer06f]. A variation of this idea is to use the type of security concerns as classification, e.g., patterns for access control, cryptography, file control, identity, firewalling, etc. For example, authentication in distributed systems is considered in: **Authenticator**, **Remote Authenticator /Authorizer**, and **Credential** (see [Fer06a] for references). Chapters 7 and 8 of [Sch06] are organized this way. Another type of concern is the general structuring of a system into core (host), perimeter, and external [Haf06].

Patterns can be defined at several levels of abstraction. This is true for analysis patterns [Fer00] and also true for security patterns. The highest level is typically a principle or a very fundamental concept, e.g. the concept of Reference Monitor, which indicates that every access must be intercepted and checked. Another example shows that firewalls, database authorization systems, and operating system access control systems are special cases of access control systems. Figure 1 shows a generalization hierarchy showing that a Firewall pattern is a concrete version of a Reference Monitor. There are four basic types of firewalls, which filter at different

architectural levels: the Application (User level) firewall, the Proxy Firewall (system application), the Stateful firewall, and the Packet Filter Firewall. An XML Firewall is a specialized type of Application Firewall. One can combine Stateful firewalls with Proxy or Packet Filter firewalls to produce even more specialized types of firewalls such as Stateful Proxy firewall, which combines aspects of both Proxy and Stateful firewalls [Sch06]. Another study shows a similar dependence based on relationships between patterns (see Section 5). From an MDA point of view these levels correspond to the CIM and PIM, since they are platform independent.

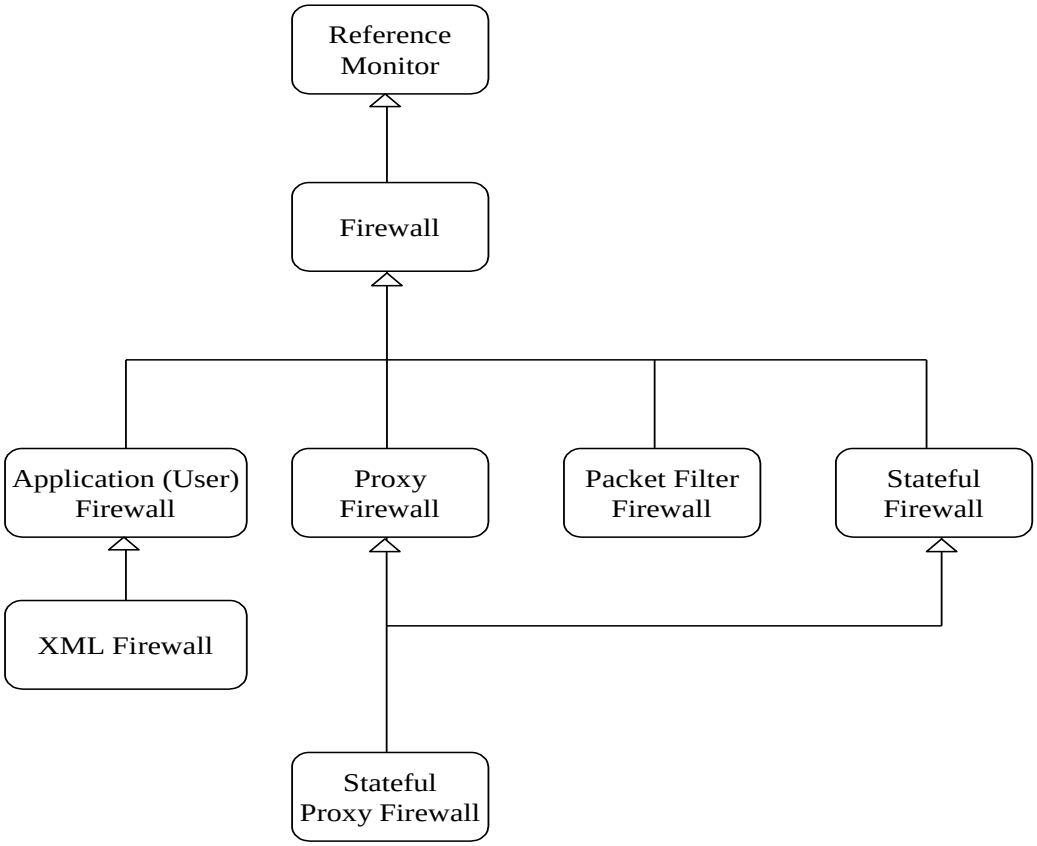

Figure 1. Firewall patterns generalization hierarchy

4. Classification based on architectural layers

We can think of a computer system as a hierarchy of layers, where the application layer uses the services of the database and operating system layers, which in turn, execute on a hardware layer. These layers provide another dimension for classification.

Two basic principles of security are:

- Security constraints should be defined at the highest layer, where their semantics are clear, and propagated to the lower levels, which enforce them.
- All the layers of the architecture must be secure.

A classification of patterns which we used in [Fer06b] is guided by these principles. We can define patterns at all levels. This allows a designer to make sure that all levels are secured, and also makes

easier propagating down the high-level constraints. At the highest level we have patterns that describe the use of security models to define access control to the application objects: **Authorization** (Access Matrix), **Role-Based Access Control** (RBAC), **Reference Monitor, Multilevel Security, Attribute-Based Access Control (ABAC**). A recent paper defines Session-Based versions of those patterns [Fer06d]. At the operating system level we have patterns such as **Secure Process**, **Controlled Virtual Address Space**, and others (see also Figure 4). Patterns for web services security include: **Application Firewall,** XML **Firewall, XACML Authorization, XACML Access Control Evaluation, and WSPL.** References for all these are found in [Fer06a]. Figure 2 shows the level distribution of these patterns. Figure 3 combines the concepts of Sections 3 and 4 showing how the same concern, e.g. Authentication, may appear in multiple levels.

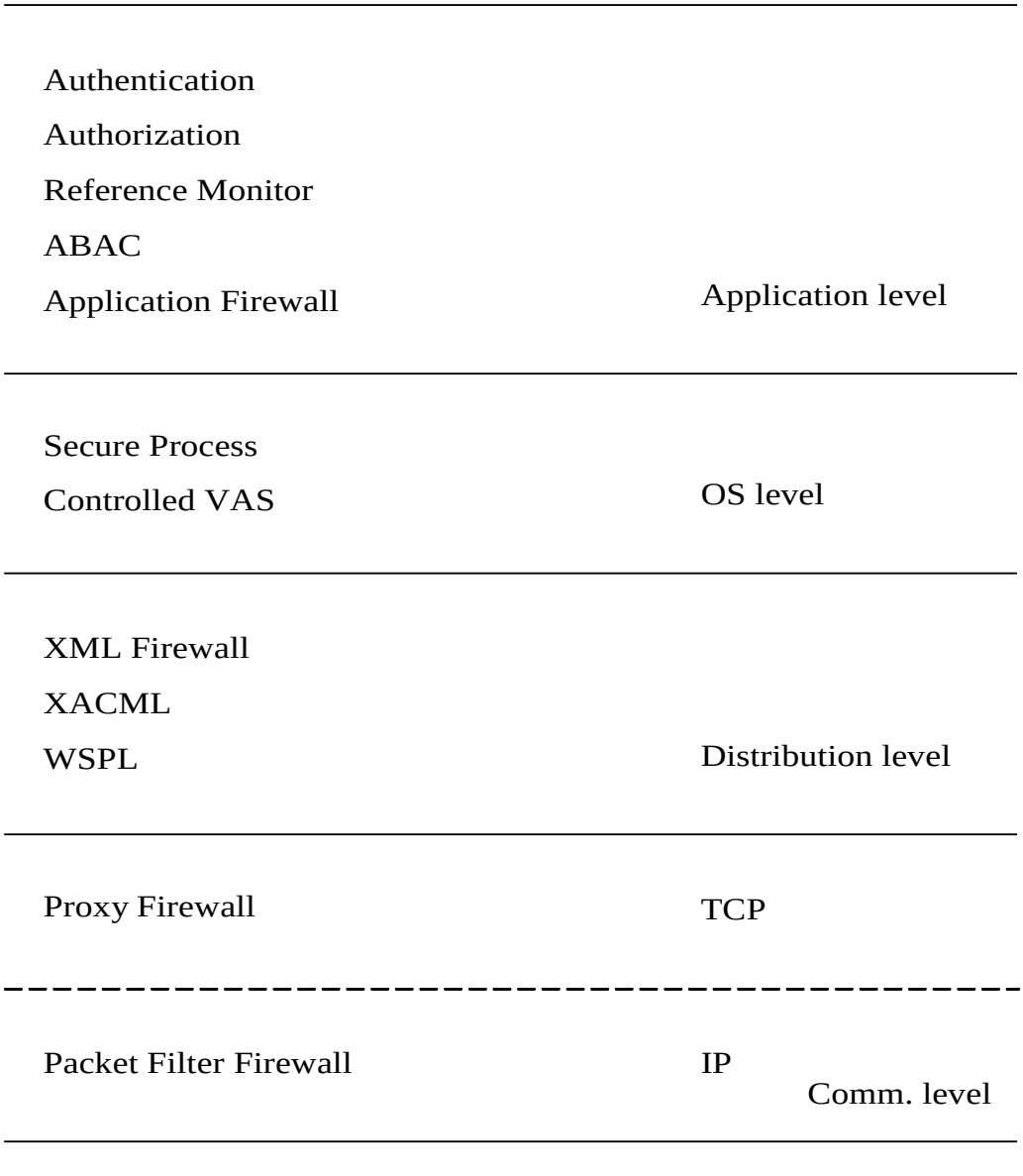

Figure 2. Pattern distribution in levels

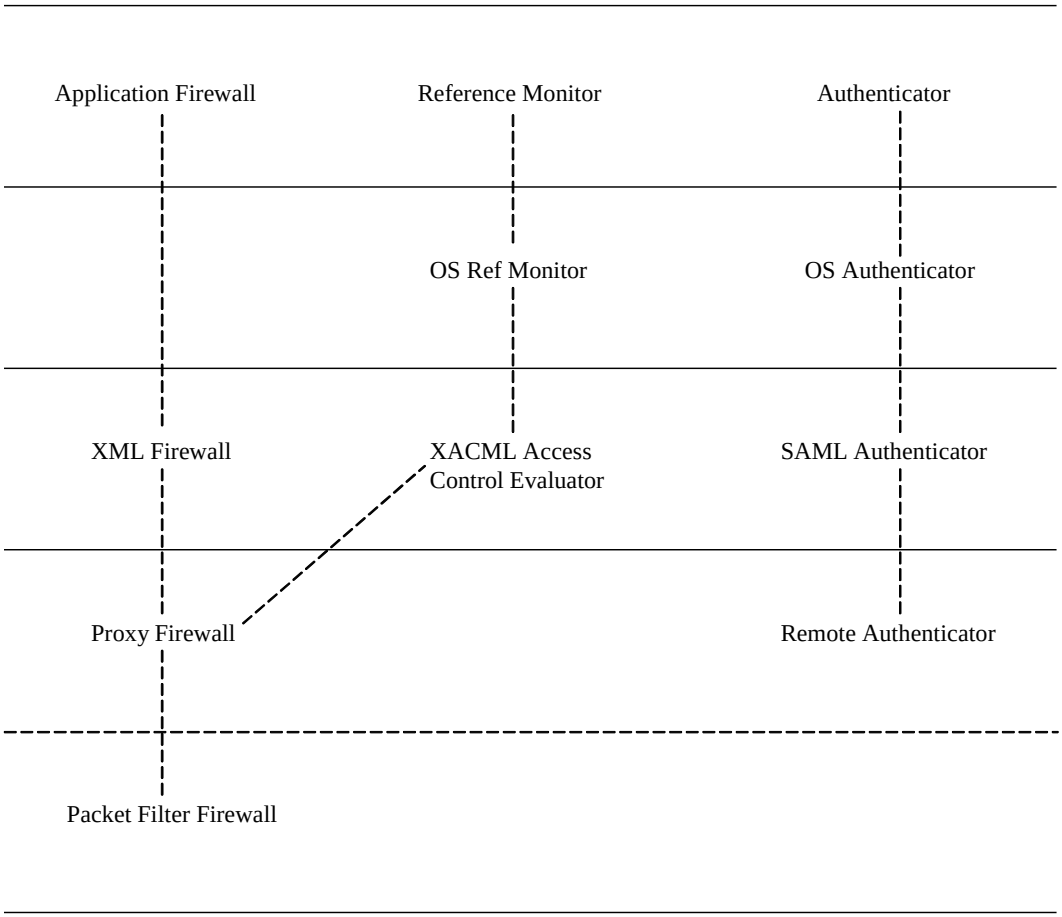

Figure 3. Types of patterns and levels

5. Relationships between patterns

Kubo et al. [Kub, Kub05] have applied a special metric to design patterns. This metric allows a classification in hierarchies with increasing level of concreteness. We apply now this metric to security patterns.

//you write something here

6. Other classifications

Other classifications include:

- [Mun06] classifies security patterns using web services transactions as references. A transaction has a start (begin) section, a processing (continue) section, and completion section. The idea is that specific security mechanisms are needed at each stage. However, not all activities in an application are transactions, so this

approach does not provide support for the whole life cycle. It can complement, however, a more complete methodology.

- [Haf06] proposed combining a table listing of patterns with a tree-structured hierarchical classification. The tree has three branches corresponding to Core security, Perimeter security, and External security. Each branch has entries for types of patterns corresponding to threats taken from Microsoft's STRIDE classification, e.g. spoofing, tampering, repudiation, etc. The problem is that not all patterns fit these categories. Also, STRIDE confuses objectives, e,g, confidentiality, with ways of attack, e.g. elevation of privilege. [Haf06] also uses a Microsoft classification for global separation, which is not very precise or useful for architectural aspects.

Can we classify patterns according to the type of threats they address? The problem with this approach is having a complete set of threats. Also, a given pattern may contro several threats or be just a part of a mechanism to control some type of threats. We have propose the concept of attack patterns as a way to relate attacks to patterns [Fer07]. In this view, a specific generic attack is related to several patterns that could prevent it from happening.

7. Pattern diagrams

Figure 2 shows a pattern diagram that relates some of our operating system security patterns (the ones with double lines are described on [Fer06c], the others in [Sch06], a general discussion is given in [Fer05b]). A pattern diagram uses these classifications to help the designer navigate in the design space. For example, an operating system designer can start from a Secure Process and use a Controlled Process Creator to create new processes in a secure way (controlling their initial rights). These processes can then execute in a Controlled Virtual Address Space (with controlled rights). The general structure of the virtual address space is defined through a Virtual Address Space Structure Selection.

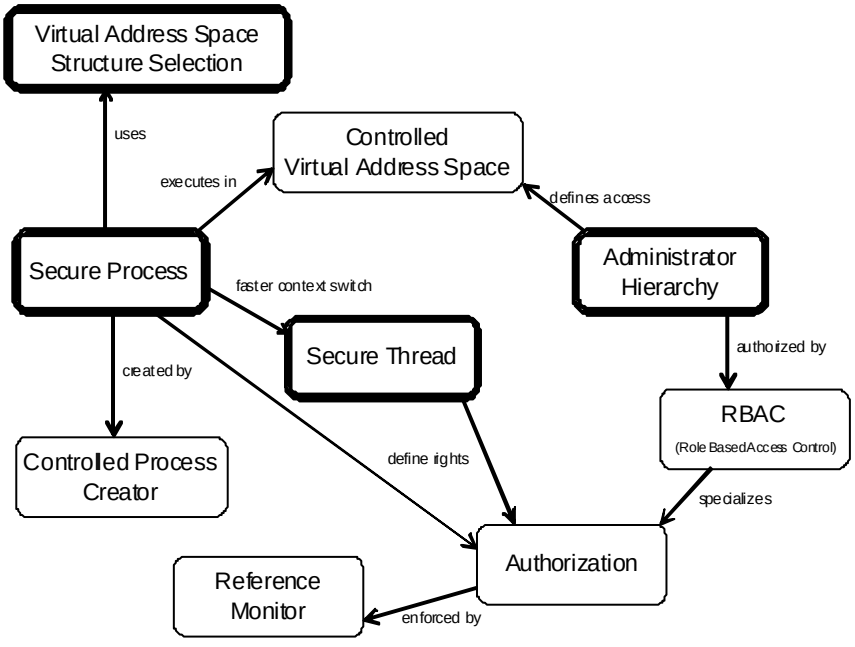

Figure 2. Pattern diagram for some operating system patterns

8. Conclusions

Patterns can be classified according to many viewpoints. A good classification can make their selection easier and more precise. We have shown some possibilities. Pattern diagrams, by summarizing all the relevant patterns at a given stage or for a given concern, can guide designers in the selection of appropriate patterns.

Patterns under development include patterns for identity management and for wireless standards. Future work will include completing our methodology and the development of further patterns. We are also working on the use of patterns combined with Model-Driven Development to produce secure systems.

References

[Avg05] P. Avgeriou and U. Zdun, "Architectural patterns revisited—A pattern language", *Procs. EuroPLoP 2005*, 1-39.

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal., *Pattern- oriented software architecture*, Wiley 1996.

[Fer05a] E.B.Fernandez and M. M. Larrondo-Petrie, "Teaching a course on data and network security using UML and patterns", *Procs. of the Educators Symposium of MoDELS/UML 2005*, Montego Bay, Jamaica, October 2-7, 2005.

[Fer05b] E.B.Fernandez and D. L. la Red Martinez, "Using patterns to develop, evaluate, and teach secure operating systems", *Proceedings of the Congreso Internacional de Auditoría y Seguridad de la Información (**CIASI** 2005)*, Madrid, Spain, 125-130.

[Fer06a] E.B. Fernandez, "Security patterns", *Procs. of the Eigth International Symposium on System and Information Security - SSI´2006,Keynote talk,* Sao Jose dos Campos, Brazil, November 08-10, 2006.

[Fer06b] 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.

[Fer06c] E.B.Fernandez, T. Sorgente, and M.M. Larrondo-Petrie, "Even more patterns for secure operating systems', *Procs. of the Pattern Languages of Programming Conference (PLoP 2006).*

[Fer06d] E.B.Fernandez and G. Pernul, "Patterns for session-based access control", *Procs. of the Pattern Languages of Programming Conference (PLoP 2006).*

[Fer06e] E.B.Fernandez and N. Delessy, ""Using patterns to understand and compare web services security products and standards", *Proceedings of the IEEE Int. Conference on Web Applications and Services (ICIW'06*), Guadeloupe, February 2006.

[Fer06f] 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, J.C. Pelaez, and M.M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool", *Procs. of the Third Annual IFIP WG 11.9 Int. Conf. on Digital Forensics,* Orlando, FL, Jan. 29-31, 2007.

[Fow97] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

[Gam94] E. Gamma, R. Helm,R. Johnson, and J. Vlissides, *Design patterns –Elements of reusable object-oriented software*, Addison-Wesley 1994.

[Haf06] M. Hafiz and R. Johnson, "Security patterns and their classification schemes", https://netfiles.uiuc.edu/mhafiz/www/

[Hat07] D. Hatebur, M. Heisel, and H. Schmidt, "A security engineering process based on patterns", accepted for the *1st Int. Workshop on Secure Systems Methodologies Using Patterns (SPattern'07),* Regensburg, Germany, September 03-07, 2007.

[Jue02] J. Juerjens, "UMLsec: Extending UML for secure systems development", *Proc. of the 5th Int. Conf. on UML, UML 2002*, LNCS, Vol. 2460, Springer, 2002, 412-425.

[Jue04] J. Juerjens, *Secure systems development with UML*, Springer-Verlag, 2004.

[Kub] A. Kubo, H. Washizaki, and Y. Fukuzawa, "A metric for measuring abstraction level of design patterns", //where?

[Kub05] A. Kubo, H. Washizaki, A. Takasu, and Y. Fukazawa, "Extracting relations among embedded software design patterns", *Trans. of the Society for Design and Process Science*, Sept. 2005, vol.9, No 3, 39-52.

[Mañ04] A. Maña, D. Ray, F. Sanchez, and M.I.Yague, "Integrando la ingenieria de seguridad en un proceso de ingenieria software", *Reunion Española sobre Criptologia y Seguridad de Informacion (RECSI 2004*), Madrid, 2004.

[Mun06] J. Muñoz Arteaga, R. Mendoza González, F. J. Álvarez, M. Vargas Martín "A classification of security patterns for the transactions between a requester, an intermediary and a web-service". Communication, Network, and Information Security 2006, 132-137

[ope] The Open Group, *Security Design Patterns Technical Guide,* http://www.opengroup.org/security/gsp.htm

[Ray04] Indrakshi Ray, R.B. France, N. Li, and G. Georg, "An Aspect-Based approach to modeling Access Control Concerns", *Journal of Information and Software Technology*, Vol. 46, No. 9, July 2004, 575-587.

[Sch06] M. Schumacher, E.B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns*, J. Wiley & Sons, 2006.

[sec] The Security Patterns page, maintained by M. Schumacher, http://www.securitypatterns.org

[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.