Computers & Security (2005) **24**, 308–321

ELSEVIER

Computers & Security

www.elsevier.com/locate/cose

Secure information systems development – a survey and comparison

Rodolfo Villarroel[a], Eduardo Fernández-Medina[b,*], Mario Piattini[b]

[a]*Departamento de Computación e Informática, Universidad Católica del Maule, Chile*
[b]*Departamento de Informática, Universidad de Castilla-La Mancha, Paseo de la Universidad no 4, 13071 Ciudad Real, Spain*

Received 22 July 2004; revised 22 July 2004; accepted 6 September 2004

KEYWORDS
Security;
Confidentiality;
Security design;
Multidimensional modeling;
UML;
Secure information systems development;
Comparison framework

Abstract Nowadays, security solutions are mainly focused on providing security defences (such as firewalls, routers, configuration server, password and encryption) instead of solving one of the main reasons of security problems that refers to an appropriate information systems design. Fortunately, there have been developed new methodologies incorporating security into their development processes. This paper makes a comparison of eleven secure systems design methodologies. The analysed methodologies fulfil criteria partially and in this paper, we make it clear that security aspects cannot be completely specified by these methodologies since they have a series of limitations that we have to take into account. At the same time, each one of these methodologies comprises very important aspects concerning security that can be used as a basis for new methodologies or extensions that may be developed.

Introduction

Security is a ''horizontal'' aspect of software development that affects very closely each component of an application and, its integration into the software development process is not appropriately understood. According to ISO/IEC 15408-1 (1999), the concept of security refers to the capability of a software product to protect data and information in order to avoid that unauthorized individuals or systems are able to read and modify them and not to deny access to authorized staff. Castano et al. (1995) refer to computing security as the protection of information against unauthorized queries, inappropriate modifications or the lack of availability of a service in a given moment. Sometimes, databases and data warehouses also store information regarding private or personal aspects of individuals, such as identification data, medical data or even religious beliefs,

* Corresponding author. Tel. +34 926295300; fax: +34 926295354.
E-mail addresses: rvillarr@spock.ucm.cl (R. Villarroel), eduardo.fdezmedina@uclm.es (E. Fernández-Medina), mario.piattini@uclm.es (M. Piattini).

0167-4048/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2004.09.011

ideologies, or sexual tendencies. In this case, confidentiality is redefined as privacy. At present, ensuring appropriate information privacy is a pressing concern for many companies, since there is a privacy legislation such as the United States' HIPAA (Health Insurance Portability and Accountability Act) (Federal Trade Commission, 1996) that regulates the privacy of personal health care information, Gramm-Leach-Bliley Act (also known as the Financial Modernization Act), Sarbanes–Oxley Act, and the EUs Safe Harbour Law.

Confidentiality should be taken into account in every information systems (ISs) development, because of the fact that the very survival of the organization depends on the correct management, security and confidentiality of information (Dhillon, 2001). However, ISs security is considered in the industry once the system is developed. This approach is known as "Penetrate and Patch" (McGraw, 2002), and it has been proved to have bad results. It is less common that developers take this aspect into consideration in earlier stages such as analysis and design. Solutions are mainly focused on providing security defences (such as firewalls, routers, configuration server, password and encryption) instead of solving one of the main reasons of security problems that refers to an appropriate software design (Ghosh et al., 2002). In simple economic terms, to find and eliminate mistakes in a software system before it is finished is cheaper and more effective than to try to correct systems after having been finished (Brooks, 1995).

Several papers deal with the importance of security in the software development process. Ghosh et al. (2002) state that security must influence all aspects of design, implementation and software testing. Hall and Chapman (2002) put forward ideas about how to build correct systems that fulfil not only the normal requirements but also the security ones. These ideas are based on the use of several formal techniques of requirement representation and a strong correction analysis of each stage. As a result of technological changes, such as access to databases via web, development of electronic commerce, advances in data warehouses and even the use of data mining techniques (Thuraisingham et al., 1998), data security problems have increased. This fact justifies the use of methodologies incorporating security into the stages of ISs development.

Fortunately, we have identified eleven methodologies that incorporate security into their development process (Artelsmair et al., 2002; Fernández, 2004; Fernández-Medina and Piattini, 2003; Fernández-Medina et al., 2004; Georg et al., 2002; Jürjens, 2002; Liu et al., 2003; Marks et al., 1996; Priebe and Pernul, 2000; Siponen, 2002; Vivas et al., 2003). Each one of these methodologies comprises very important aspects concerning security that can be used as a basis for the improvement of the current methodologies. At the same time, these methodologies have a series of limitations that must be taken into account. This fact, unfortunately, indicates that there are no consolidated methodologies that integrate security into the development process yet.

The rest of the paper is organized as follows: In next section, we will shortly describe each one of the eleven proposals that incorporate security into the stages of systems development; then, we will show the comparison framework that we have used. Further, we will make the comparison, and finally, in the last section, we will explain our conclusions.

Proposal of methodologies incorporating security

The proposals that will be analysed in our comparison are listed below.

- MOMT: Multilevel Object Modeling Technique, by Marks et al. (1996)
- Business process-driven framework for security engineering, by Vivas et al. (2003)
- UMLSec: Secure Systems Development Methodology using UML, by Jürgens (2002)
- Secure Database Design Methodology, by Fernández-Medina and Piattini (2003)
- Security and Privacy Requirements Analysis Methodology within a Social Setting, by Liu et al. (2003)
- A Paradigm for Adding Security into IS Development Methods, by Siponen (2002)
- CoSMo: An Approach Towards *Co*nceptual *S*ecurity *Mo*deling, by Artelsmair et al. (2002)
- Using Aspects to Design a Secure system, by Georg et al. (2002)
- A Methodology for Secure Software Design, by Fernández (2004)
- ADAPTed UML: A Pragmatic Approach to Conceptual Modeling of OLAP Security, by Priebe and Pernul (2001)
- A UML Extension for Secure Multidimensional Conceptual Modeling, by Fernández-Medina et al. (2004)

We have chosen these eleven methodologies because the majority of them try to solve the problem of security (mainly confidentiality) from the earliest stages of the ISs development,

emphasize security modeling aspects and use modeling languages that make it the easier security design process.

MOMT: Multilevel Object Modeling Technique

Marks et al. (1996) define MOMT (Multilevel Object Modeling Technique) as a methodology to develop secure databases by extending OMT in order to be able to design multilevel databases providing the elements with a security level and establishing interaction rules among the elements of the model. Although MOMT is mainly composed of three stages, authors only describe the essential points of its analysis stage. These stages are described below.

- Analysis stage: it allows us to analyse the requirements to detect potential system vulnerabilities. This stage consists of three models whose aim is to collect system information from several perspectives: multilevel object model (to represent static features), multilevel dynamic model (to represent dynamic features) and multilevel functional model (to represent system transformation features).
- System design stage: it allows us to design multilevel databases. To do so, it defines, at a high level, systems structure and multilevel databases.
- Object design stage: it allows us to design the modules of the automated system in a more detailed way.

Business process-driven framework for security engineering

Vivas et al. (2003) propose a business process-driven system development method where technology decisions are guided by the business model. Expressing security requirements at the business model level is motivated by the fact that applications like e-commerce transactions are conceptually similar to traditional non-automated business transactions. Notions such as non-repudiation, confidentiality, integrity, access control and authentication have played a role in business transactions long before the appearance of automated systems.

This framework is based on the UML and integrates security requirements into a business process model of the system. The UML is extended in order to express security notions. With the aim of facilitating its adoption by system developers, the framework intends to integrate security requirements into standard system development methodologies which, currently, are often UML-based and use case-driven.

Use cases and the corresponding scenarios are used as the basic tools to build threat models and elicit security requirements. The latter are originally stated at the high level of abstraction within a functional representation of the system, thus yielding a security-enriched specification. Thereafter, a machine readable XMI-representation of the system is produced and security requirements are integrated into the functional description by means of the pattern-based analysis and design process yielding a new specification of the system which the security requirements have been integrated into. The resulting representation is translated into a formal notation for testing, validation and verification. This procedure is iterated as many times as required. The result is used as an input to the following stages of system development.

UMLSec: Secure Systems Development Methodology using UML

Jürgens (2002) states a methodology to specify requirements regarding confidentiality and integrity in analysis models based on UML. Multilevel security and mandatory access control are the security models highlighted in this proposal. This approach considers a UML extension to develop secure systems. In order to analyse security of a subsystem specification, the behaviour of the potential attacker is modeled; hence, specific types of attackers, that can attack different parts of the system in a specific way, are modeled. This proposal uses the majority of UML diagrams to model security aspects, mainly those referred to confidentiality and integrity, for example, state chart diagrams model, the dynamic behaviour of objects, and sequence diagrams are used to model protocols. Deployment diagrams are also used to model links among components across servers. Besides, this methodology incorporates the translation of UMLSec models defined for the introduction of patterns into the design process.

Secure Databases Design Methodology

Fernández-Medina and Piattini (2003) propose a methodology to design multilevel databases by integrating security into each one of the stages of the databases life cycle.

This methodology includes the following aspects:

- A specification language of multilevel security constraints about the conceptual and logical models.
- A technique to the early gathering of multilevel security requirements.
- A technique to represent multilevel databases conceptual models.
- A logical model to specify the different multilevel relationships, the metainformation of databases and constraints.
- A methodology based upon the Unified Process (Jacobson et al., 1999) with different stages that allow us to design multilevel databases.
- A CASE tool that helps us to automate multilevel databases analysis and design process.

Security and Privacy Requirements Analysis Methodology within a Social Setting

In Liu et al. (2003), it is stated that a methodological framework to deal with security and privacy requirements based on i*, which is an agent-oriented requirements modeling language.

This framework is formed by a set of analysis techniques as follows:

- Attacker analysis: it helps us to identify system potential attackers and their malicious intents.
- Dependency vulnerability analysis: it helps us to detect vulnerabilities in terms of organizational relationships among stakeholders.
- Countermeasure analysis: the necessary factors for a successful attack are the attacker motivation, the system vulnerabilities and the attackers' capabilities to carry out the attack.
- Access control analysis: it establishes a link between security requirements models and security implementation models. To do so, it uses i* models to polish a proposed solution and to generate a system design.

The concepts provided by i* language enable us to analyse security aspects within their social settings, giving place to a systematic way to find vulnerabilities and threats.

A Paradigm for Adding Security into IS Development Methods

Rather than presenting another security approach with its own novel security features Siponen (2002), proposes a new paradigm for secure IS which will help developers to use and modify their existing methods as needed. The meta-method level of abstraction offers us a perspective of secure IS development that is in a constant state of emergence and change. Furthermore, developers recognize regularities or patterns in the way that problem settings arise and methods emerge.

The author uses the following analytical process for discovering the patterns of security design elements. In the first place, he looked across IS software development and IS security development methodologies in order to find common core concepts (subjects and objects). Secondly, he surfaced patterns in existing secure IS methods giving place to four additional concepts (security constraints, security classifications, abuse subjects and abuse scenarios, and security policy). Finally, he consulted a panel of practitioners for comments about these patterns. This process led to a pattern formed by six elements. Additional elements can be certainly added to the meta-notation on an ad hoc basis as required. To summarize, the meta-notation includes six dimensions: security subjects, security objects, security constraints, security classifications, abuse scenarios, and security policy. Security subjects denote the different security relevant entities, i.e., entities that have a relevant security connection to the assets of the organization (security objects). The term *security objects* refers to the assets of the organization that are relevant in terms of information security. These assets (security objects) may range from physical things such as paper to electronic entities such as files. Security constraints may include write access, read access, etc., and they may be discovered by analysing the security requirements (confidentiality, integrity, availability and non-repudiation) for each security object. Security classification stems from the need to classify security objects and subjects according to their information security sensitivity. Abuse subjects form a special class of security subjects and refer to those subjects that may carry out a security violation. Abuse subjects and abuse scenarios may be needed for two kinds of situations. Firstly, they may come in handy when there is a need to explore and identify what potential threat scenarios exist. Secondly, this class is relevant for testing purposes since it helps us to check that the system and software under design can cope with unwanted scenarios or attacks by unauthorized people or processes.

CoSMo: An Approach Towards *Co*nceptual Security *Mo*deling

In Artelsmair et al. (2002), the need to integrate security considerations into the software modeling

process is identified. Conceptual modeling should have to encompass security requirements and high-level security mechanisms. Authors work on the development of a conceptual security modeling method that they refer to as CoSMo (*Co*nceptual *S*ecurity *Mo*deling).

Prior to an overview of which security mechanisms can enforce which requirements, fundamental issues of security policies are elaborated. A security policy consists of a set of laws, rules and practices that regulate how an organization manages, protects and distributes sensitive information. Each security requirement can be enforced by one or more security mechanisms, resulting in a requirements/mechanisms matrix. The security requirements and mechanisms are generically defined, since they are used for security modeling at the conceptual level.

First of all, authors show how security considerations can be integrated into the conceptual modeling process. Next, they systematically enumerate frequently encountered security requirements and clearly indicate which mechanisms are used to enforce them.

Generally, a claimed security requirement is not part of the use case diagram but is part of the description of the use case. In CoSMo, it will be possible to model this security requirement at the conceptual level, even in a use case diagram.

Using Aspects to Design a Secure System

In Georg et al. (2002), authors focus on the use of aspects for modeling and weaving in security concerns. They propose an aspect-oriented design (AOD) technique for designing a secure system. An aspect-oriented design consists of a primary model and one or more aspects that capture design concerns that crosscut the design units of the primary model. Incorporating these aspects into the primary model is called weaving. Weaving results in a model in which the design units impacted by the concern represented by the aspects are accordingly modified. In this work, aspects are treated as design patterns.

An aspect is defined in terms of structures of roles called Role Models. Role Models are used because they allow us to express aspects as patterns. Aspect properties are defined in terms of roles that can be played by model elements representing application-specific concepts. Model elements are UML constructs. The way in which multiple aspects are woven into a primary model is determined by weaving strategies. A weaving strategy identifies security aspects taking into consideration the kinds of attacks that are possible in a system and the mechanisms that can be used to detect, prevent, and recover from such attacks. A design aspect (e.g., a security concern) can be modeled from a variety of perspectives. Authors focus on two aspect views: static and interaction views. A static view of an aspect defines the structural properties of the aspect. The interaction view specifies the interaction patterns associated with the aspect. In this proposal, authors use specialized forms of two Role Model types to model aspects from these views: Static Role Models (SRMs) and Interaction Role Models (IRMs). SRMs define patterns of UML static structural models (e.g. Class Diagram patterns) while IRMs define UML interaction diagram patterns (e.g. Collaboration Diagram pattern). An aspect definition typically consists of a single SRM and one or more IRMs.

A Methodology for Secure Software Design

The main idea of the methodology proposed in Fernández (2004) is that security principles should be applied at every development stage and that each stage can be tested for compliance with those principles. The secure software lifecycle is as follows.

- Requirements stage: from the use cases, we can determine the rights needed by each actor and thus, apply a need-to-know policy. Since actors may correspond to roles, this is now a Role-Based Access Control (RBAC) model. The set of all use cases defines all the uses of the system and from all the use cases; we can determine all the rights for each role. We can then consider possible attacks in the context of these use cases.
- Analysis stage: we can build a conceptual model where repeated applications of the authorization patterns realize the rights determined by use cases. Analysis patterns can be built with predefined authorizations according to the roles in their use cases. This fact makes the job of defining rights even easier.
- Design stage: interfaces can be secured by applying again the authorization pattern. The user interfaces should correspond to use cases. Secure interfaces enforce authorizations when users interact with the system. Deployment diagrams can define secure configurations to be used by security administrators. A multilayer architecture is now needed to enforce the security constraints defined at the application level. At each level, patterns are used to represent appropriate security mechanisms.

- Implementation stage: this stage requires reflecting in the code the security constraints defined for the application. Since these constraints are expressed as clauses and associations, they can be implemented as functional classes.

At the end of each stage, we can perform audits to verify that institution policies are being followed. If necessary, the security constraints can be made more precise by using Object Constraint Language (OCL) instead of textual constraints. Using layers, we can define patterns at all levels that altogether implement a secure or reliable system. The main idea of the Layers pattern is the decomposition of a system into hierarchical layers of abstraction, where the higher levels use the services of the lower levels.

ADAPTed UML: A Pragmatic Approach to Conceptual Modeling of OLAP Security

A methodology and language for conceptual modeling of OLAP security is presented in Priebe and Pernul (2001), by creating a UML-based notation named ''ADAPTed UML'' (which uses ADAPT symbols as stereotypes). Authors chose ADAPT because it is mentioned at different ORACLE Express user group conferences, and this notation is one of the few exceptions which does not use an intuitive ER like notation in conceptual modeling. The conceptual (meta) model that has been developed for conceptual OLAP models introduces the elements cube, measure, dimension level, and dimension attribute.

As part of the logical design, the system-independent conceptual model is transformed into a logical (system-dependent) ''implementation model''. Restrictions of the chosen system (i.e. DBMS or OLAP software) have to be taken into account. Authors use a logical OLAP model that is based on the Microsoft Analysis Services (shipped with SQL Server 2000).

The security model for OLAP is based on the assumption of a central (administrator based) security policy. The access restrictions are defined as authorization constraints making the identification of security objects and subjects necessary. At this point, they assume the notion of (non-overlapping, non-hierarchical) roles as security subjects. Therefore, in addition to the above-mentioned elements (cubes, dimensions, etc.) the element role is introduced. authorization constraints can be either positive (explicit grants) or negative (explicit denials). The security model is based on an open world policy (i.e. access to data is allowed unless it is explicitly denied) with negative authorization constraints. This fact corresponds to the open nature of OLAP systems.

Authors state a multidimensional security constraint language (MDSCL) that is based on MDX representation of the logical OLAP model used by Microsoft. In order to express the (negative) authorization constraints, they propose a set of HIDE statements. For example: HIDE CUBE, HIDE MEASURE, HIDE SLICE, HIDE LEVEL, HIDE LEVEL WHERE, HIDE MEASURE WHERE.

A UML Extension for Secure Multidimensional Conceptual Modeling

An extension of the UML that allows us to represent the main security information of data and their constraints in the multidimensional (MD) modeling at the conceptual level is presented in Fernández-Medina et al. (2004). The proposed extension is a UML profile that allows us to consider the main MD modeling properties and it is based on the UML (designers can avoid learning a new specific notation or language). The authors of this methodology consider the multilevel security model, but they focus on taking into consideration aspects regarding read operations because this is the most common operation for final user application. This model allows us to classify both information and user into security classes, and enforce the mandatory access control. By using this approach, we make it possible to implement secure MD models with any of the DBMS that are able to implement multilevel databases, such as Oracle Label Security (Levinger, 2002) and DB2 Universal Database (UDB) (IBM, 2004). An extension to the UML begins with a brief description and then lists and describes all of the stereotypes, tagged values, and constraints of the extension. In addition to these elements, an extension contains a set of well-formedness rules. These rules are used to determine whether a model is semantically consistent with itself. According to this fact, authors define a UML extension for secure conceptual MD modeling following a schema composed of these elements: *description* (a little description of the extension in natural language), *prerequisite extensions* (it indicates whether the current extension needs the existence of previous extensions), *stereotypes/tagged values* (the definition of the stereotypes and/or tagged values), *well-formedness rules* (the static semantics of the

metaclasses are defined both in natural language and as a set of invariants expressed by means of OCL expressions), and *comments* (any additional comment, decision or example, usually written in natural language). For the definition of the stereotypes, authors follow a structure composed of a name, the base metaclass, the description, the tagged values and a list of constraints defined by means of OCL. For the definition of tagged values, the type of tagged values, the multiplicity, the description, and the default value are expressed.

Comparison framework

We have used the comparison framework proposed by Khwaja and Urban (2002). We have chosen this framework since it establishes a clear differentiation between the concepts of specification and specification techniques. There are other comparison frameworks but this is one of the most recent and it solves the problem that many authors intermingle the concepts of specification and specification technique. The criteria used for one of these concepts should not be applied to the other, since it can influence the establishment/adaptation and suitable use of a methodology that considers information security aspects. For instance, a specification can be complete and consistent regardless of the way used to represent it, the process used in its construction, the degree/extent of tools and automation used or whether it is formal or informal. However, it is significant to indicate that a technique can be used to produce consistent or complete specifications. The criteria should be separated but there should exist a mapping between them, which means that the specification technique features help us to achieve certain features in a specification.

In the context of software engineering, specification is a description of externally known features, a complete behaviour, in other words, input/output, description of several systems interfaces, etc. The concept of specification is, thus, a precise sentence of the requirements that a system must satisfy. A software specification technique is a method to achieve the desired purpose or product.

The fulfilment of a technical criterion should carry out the fulfilment of the specification criteria related to that technical criterion as well. For example, if the technical criterion is the formality level, then, a high level of formality in a specification technique can help us to achieve a precise, unambiguous, consistent, complete definition and verifiable specifications.

The specification criteria, with their terms and phrases that describe the same criterion, are the following: understandable (a system specification must be a cognitive model, comprehensibility), appropriate (separate functionality from implementation), unambiguous (precision, lack of ambiguity), complete, consistent, correct, verifiable (analysability), validateable (testability), modifiable (maintainability, adaptability), traceable, minimal (economy of expression).

The specification technique criteria and their meanings are as follows:

- Expressive adequacy: technique supports conciseness of representation. The expressive capability of a technique may enhance specification comprehensibility, appropriateness, and minimality.
- Constructibility: it refers to the ease of construction of a specification using the technique.
- Scope of specifications: scope deals with both functional and performance specifications. Indeed, a specification for a system should consist of functional as well as non-functional requirements specifications.
- Level of formality: high level of formality in a specification technique may help us to define precise, unambiguous, consistent, complete, and verifiable specifications.
- Formal foundation: high formal foundation in a specification technique may help us to define precise, unambiguous, consistent, complete, and verifiable specifications.
- Extent of applicability: it deals with the range of domains that can be specified by a technique.
- Easy to use: it deals with the ease that a technique may be used with, without much knowledge or special training.
- Help support: it deals with aspects such as the procedures, guidelines, standards, and case studies available for a technique. This criterion may help us to use a technique and construct specifications within the technique.
- Integrated environment and tool support: it deals with the tools available in an integrated fashion for a technique. This criterion may help us to use a technique, construct specifications within a technique, and make an automatic analysis of specifications.
- Specification organizational support: technique supports good organizational principles to control complexity. Good specification organization helps us to control complexity and enhance understandability.

- Support for maintainability: the technique should facilitate specification modifications. Maintainable specifications are easily modifiable and traceable.
- Executable: a specification must be operational. An operational model of specifications may help us to increase understandability, reduce ambiguity, improve consistency, ensure completeness and correctness, and make specifications more verifiable and validateable.
- Tolerance for incompleteness: the system specification must be tolerant of incompleteness and augmentable. Execution of incomplete specifications may help testability at several stages of the specification development.
- Multiple views: proper use of a technique should enhance understandability for non-computer specialists. Multiple views of a specification may enhance its understandability.
- Notational simplicity and flexibility: technique supports conceptual clarity to the client. It may improve specification understandability.
- Internal verification support: a technique should provide means for specification consistency checks. Automatic internal verification supported by a technique may improve the reduction of ambiguity, ensure completeness, improve consistency, and hence, make specifications more verifiable.
- External validation support: it may ensure correctness of a specification by validating it against requirements and/or implementation. This criterion may also improve validation by generating test cases and using the same test for specification, as well as the implementation validation.
- Support for other development stages: automatic design and implementation generation from specification may improve traceability across the development stages.
- Support for documentation generation: automatic documentation generation from specifications may help us to increase understandability of specifications.

Comparison

Table 1 allows us to relate specification and specification technique criteria. We can see, for instance, that the fulfilment of a technical criterion must generate the fulfilment of all specification criteria related to that criterion. The fulfilment of a specification criterion (for example, unambiguous) can partially help to achieve the fulfilment of several technical criteria such as the level of formality, formal foundation, maintainability and internal verification support. The degree of fulfilment will be ''X'' for *Yes*, '' '' for *No* and ''(x)'' for *Partial*. As each specification technical criterion can be associated to one or more specification criteria, the answer of each methodology will be related to the technical fulfilment with respect to a specification criterion. For example, we can look up if there is an ''expressive adequacy'' that allows an ''understandable'' specification. To know if this criterion is completely fulfilled, the specification must be ''understandable'', ''appropriate'' and ''minimal''.

All the above-proposed ideas are very interesting and provide us with important contributions to solve the security problem in a methodological way. We can conclude that, at a general level, all of them fulfil the criteria associated to formal aspects; they are serious proposals, very well based and supported by a modeling language. The deficiency is observed in the automated support that each one of these methodologies needs; specifically, it can be mentioned as the lack of an automatic instrument of internal verification. Moreover, each proposal has several weaknesses.

The multilevel databases design methodology called MOMT only explains the analysis stage; it does not propose valid solutions for current situations in which used technologies and security needs have changed, as we can see in the specification criterion ''appropriate''.

The proposal made by Vivas et al. is an ongoing work intended to establish a use case-driven software development framework based on the UML, as well as to integrate security requirements into a business process model of the system. The proposal is tentative and exploratory, but it is focused on a discussion and identification of the problems rather than on providing solutions. This fact can be seen in the partial fulfilment of most of the criteria.

The proposal made by Liu et al. is mainly associated to the security requirement analysis process from a top-down or bottom-up perspective. Moreover, the used techniques allow us to check the model and can be applied in several stages of the requirements process. The weakness of this methodology is the fact that it mentions neither the database processing nor the remaining stages of the information systems development. In addition to this, it does not consider tools that support the kind of reasoning regarding security. This fact can be visualized in the nonfulfilment of most of the necessary supports, except for the external validation support, where goal-reasoning techniques such as qualitative goal labeling algorithms and quantitative techniques can be used to identify the best design solutions.

Table 1 Comparison using evaluation criteria for software specifications and specification techniques

Technique criterion	Specification criteria	Methodologies										
		Marks et al. (1996)	Vivas et al. (2003)	Jürjens (2002)	Fernández-Medina and Piattini (2003)	Liu et al. (2003)	Siponen (2002)	Artelsmair et al. (2002)	Georg et al. (2002)	Fernández (2004)	Priebe and Pernul (2001)	Fernández-Medina et al. (2004)
Expressive adequacy	Understandable	X	X	X	X	X	X	X	X	X	X	X
	Appropriate	(x)	X	(x)	X	(x)	X	X	X	X	X	X
	Minimal	X	X	X	X	(x)	(x)	X	X	X	X	X
Constructibility	—	X	X	X	X	(x)	(x)	(x)	(x)	(x)	X	X
Scope of specifications	Complete	(x)	X	(x)	(x)	(x)		(x)	(x)	(x)		X
Level of formality	Unambiguous	X	(x)	X	X	X	(x)	(x)	X	(x)	X	X
	Consistent	X	(x)	X	X	X	(x)	(x)	X	(x)	X	X
	Complete	X	(x)	X	(x)	X	(x)	(x)	X	(x)	X	X
	Verifiable	X	(x)	X	X	X	(x)	(x)	X	(x)	X	X
	Validateable	X	(x)	X	X	X	(x)	(x)	X	(x)	X	X
Formal foundation	Unambiguous	X	(x)	X	X	X	X	X	X	X	X	X
	Consistent	X	X	X	X	X	X	X	X	X	X	X
	Complete	X	(x)	X	(x)	X	X	X	X	X	X	X
	Verifiable	X	X	X	X	X	X	X	X	X	X	X
	Validateable	X	X	X	X	X	X	X	X	X	X	X
Extent of applicability	—	(x)	X	(x)	X	(x)	X	X	X	X	X	X
Easy to use	—	X	X	X	X	(x)	X	(x)	(x)	(x)	X	X
Help support	—		(x)	(x)	(x)				(x)		(x)	(x)
Integrated environment and tool support	—		(x)	(x)	(x)							
Specification organization support	Understandable	X	(x)	X	X	X	(x)		(x)			(x)
	Modifiable	X	X	X	X	X	(x)		(x)			(x)
Support for maintainability	Modifiable	X	(x)	X	X	X						
	Traceable	(x)	(x)	X	(x)	X						
Executable	Understandable	(x)	(x)		X	(x)	(x)				(x)	
	Unambiguous	(x)	(x)		X	X	(x)				(x)	
	Consistent	(x)	(x)		X	X	(x)				(x)	
	Complete	(x)	(x)		(x)	(x)	(x)				(x)	
	Correct	(x)	(x)		X	X	(x)				(x)	
	Verifiable	(x)	(x)		X	X	(x)				(x)	(x)
	Validateable	(x)	(x)		X	(x)	(x)				(x)	(x)

Tolerance for incompleteness	Verifiable	X	X	X	(x)	X	X		(x)			
	Validateable	X	X	X	(x)	X	X		(x)			
Multiple views	Understandable	X	(x)	X	X	X		(x)	(x)	(x)	X	
Flexibility and notational simplicity	Understandable	X	X	X	X	(x)	(x)	(x)	(x)	X	X	X
Internal verification support	Unambiguous											
	Complete											
	Consistent											
	Verifiable											
External validation support	Correct	X	(x)	(x)	(x)	(x)						
	Validateable	(x)	(x)	(x)	(x)	(x)						
Support for other development stages	Traceable			(x)	(x)							
Support for documentation generation	Understandable	(x)		(x)	(x)			(x)	(x)			

UMLSec security proposal takes into account security requirements related to confidentiality and integrity aspects. However, it does not comprise aspects associated to databases security design. As this methodology tries to make a broader study, it does not consider secure databases design according to conceptual, logical and physical aspects, which is essential in systems security.

The proposal stated by Fernández-Medina and Piattini only studies use cases diagrams, class diagrams and OCL (Object Constraint Language) to model security. This proposal offers us an extension of involved languages and techniques to comprise confidentiality aspects in databases. Nevertheless, this methodology is not adequate for developing secure ISs.

The proposal made by Siponen is a meta-notation (six dimensions) which ensures that security issues are properly and easily addressed in IS secure design. According to the author, this proposal satisfies the requirements of autonomy better than any existing approach for designing secure IS/software. Practitioners can continue to use their favoured IS secure design methods as a basis for the development and management of secure IS. Our critic is associated to a deficiency of the graphical models and languages used to support this proposal or to provide it with greater formality. The proposal will be successful depending on the ability of the user who must incorporate elements of security into the methodologies that are being used in a given moment because this proposal, in spite of being very interesting, is dealt with in a very general way. Furthermore, doubts can be casted upon the applicability of the idea of the meta-notation in the different IS development methods.

The proposal called CoSMo will provide us with a method for conceptual modeling of secure systems on the basis of the UML, but it has been developed partially; it does not exist either as a definition of a formal notation based on the UML or as an integration of it into existing toolkits.

The proposal made by Georg et al. differs from the others in the fact that the focus is not on extending modeling notations such as the UML but on how security concerns can be first encapsulated and then woven into primary design models. By supporting the separation of concerns during the design stage, the complexity of designing large systems can be better managed. This work can also take advantage of UML security extensions: the weaving process can be designed to produce extended forms of the UML that reflect the properties expressed in the aspects in a more direct way. Authors have not developed

a prototype tool that will support flexible weaving, by providing users with a language for describing reusable weaving strategies and weaving procedures.

The proposal made by Fernández states that the combination of multilayer architectures with patterns provides us with a framework to develop a systematic and reusable approach to build systems that satisfy specific non-functional requirements, but it is necessary to work on this subject to add more patterns at each level and to collect and unify these patterns. Also, our critic is associated to a deficiency of the graphical models and languages used to support this proposal or to provide it with greater formality.

The proposal made by Priebe and Pernul is interesting but they have limited their model to very simple security subjects (non-hierarchical, non-overlapping roles). Role-based access control models usually provide us with the possibility of role hierarchies. A role hierarchy is interpreted in such a way that authorizations or constraints of a superior role are inherited by subordinate roles. Additionally, due to the difficulty of keeping the constraint set consistent, they have limited their security model to negative authorizations. In principle, a combination of negative and positive authorizations would be possible (i.e. HIDE and SHOW statements). Nevertheless, a combination with role hierarchies would make a conflict resolution strategy even more difficult.

The proposal made by Fernández-Medina et al., in spite of being very interesting and solid in terms of conceptual modeling, has deficiencies associated to the remaining stages of the development process. In addition, it does not have, until the moment, an automatic support that allows us to work with it, for example, the implementation of a CASE tool based on UML incorporated into the multidimensional modeling.

Table 2, shows a synthesis of the contributions, in security terms, made by each one of the analysed methodologies.

It is very difficult to develop a methodology that fulfils all criteria and comprises all security constraints in terms of confidentiality, integrity and availability. If that methodology was developed, its complexity would avoid its success. Therefore, the solution would be a more complete approach in which techniques and models defined by the most accepted model standards were used. And, if these techniques and models could not be directly applied, they must be extended by integrating the necessary security aspects that, at present, are not covered by the analysed methodologies.

Conclusions

The criticality of IS, and especially databases and data warehouses for modern business, together with the new requirements of laws and governments, make it necessary the development of more sophisticated approaches to ensure data security. Traditionally, information security deals with different research topics, like access control techniques, cryptographic methods, etc. Although all these topics are very important, we think that it is fundamental to use a methodological approach, where security (at different levels) is taken into consideration at all stages of the systems development process. There are several interesting methodological proposals, but some of them propose different notations for modeling security aspects of IS, databases and data warehouses development. Hence, we propose that a standardised methodological approach that allows us to build IS taking into account security aspects from the earliest stages of development until the end of the process should be developed. This methodological approach should be an extension of existing modeling methodologies and standards because, otherwise, organizations that are really interested in DW security would have to make a big effort to adapt to the new technology.

The most widely spread modeling standard is UML. Therefore, it would be interesting to obtain a consensus to standardise the different methodologies to specify a security profile for UML. In this way, it would be possible to provide UML with security features to be able to develop modeling including, on the one hand, the UML syntax and power and on the other hand, the new security features, ready to be used, when the application includes the security requirements needed by these features. UML has been widely accepted as the standard object-oriented modeling language to model several aspects of software systems. Hence, any approach using UML will minimize the effort of developers to learn new notations or methodologies for each subsystem to be modeled. UML is an extending language since it provides mechanisms (stereotypes, tagged values and constraints) in specific domains, if necessary, such as web applications, business modeling, software development processes and so on. We consider it appropriate to use a design methodology that uses a UML profile for security aspects to be added. Moreover, we think that it is essential to use an OCL (Object Constraint Language) based language to be able to specify security restrictions precisely together with other UML diagrams and

Table 2 Summary of the contributions, in security terms, made by each one of the analysed methodologies

	Modeling/development standard	Technologies	Access control type	Constraints specification	Case tool support
MOMT	OMT	Databases	MAC	NO	NO
Vivas	UML	Information systems (only requirements, business process-driven)	—	NO	YES (ConGolog, a concurrent logic programming based on the situation calculus)
UMLSec	UML patterns	Information systems	MAC (multilevel)		NO, but work towards this goal is being undertaken by giving translations from UML into CSP which allow us to use the model checker FDR2 to check security properties
Fernández-Medina and Piattini	UML, unified process	Databases	MAC, DAC, RBAC, Constraint-based	OSCL (OCL based)	YES (Rational Rose add-in)
Liu and Yu	Agent-oriented requirement modeling language (i*)	Information systems (only requirements)	RBAC	A lightweight object modeling notation alloy	YES (alloy)
Siponen	—	Information systems, meta-methodology	—	NO	NO
CoSMo	UML	Information systems (only requirements)	—	—	NO
George et al.	UML, aspect-oriented patterns	Information systems (only design)	RBAC	They do not show the meta-model constraints in OCL, the constraints are expressed in template form	NO
Fernández	UML patterns	Information systems	Access matrix RBAC	He refers to OCL as a good solution	NO
ADAPTed UML	ADAPT UML	OLAP	RBAC	MDSCL (MDX based)	NO
Fernández-Medina et al.	UML	Data warehouses	MAC, DAC, RBAC, constraint-based	OSCL (OCL based)	NO

the development of a CASE tool (integrated, for example, into Rational Rose) to support the systems design process in a secure way for the later validation of the proposal by applying it to real situations. The biggest cost of software system is the maintenance and this is a consequence of imprecise, incomplete and arbitrary documentation. With a UML profile that allows us to model IS security requirements; a more robust specification will be achieved.

Acknowledgements

This research is part of the CALIPO (TIC2003-07804-C05-03) and RETISTIC (TIC2002-12487-E) projects, supported by the Dirección General de Investigación of the Ministerio de Ciencia y Tecnología, and the network VII-J.RITOS2 financed by CYTED.

References

Artelsmair C, Essmayr W, Lang P, Wagner R, Weippl E. CoSMo: an approach towards conceptual security modeling. In: Database and expert systems applications: 13th international conference (DEXA 2002). Air-en-Provence, France: Springer-Verlag; 2002.

Brooks F. The mythical man month: essays on software engineering. Addison-Wesley; 1995.

Castano S, Fugini M, Martella G, Samarati P. Database security. Addison-Wesley; 1995.

Dhillon G. Information security management: global challenges in the new millennium. Idea Group Publishing; 2001.

Federal Trade Commission (U.S.). Health Insurance Portability and Accountability Act (HIPPA); 1996.

Fernández EB. A methodology for secure software design. In: The 2004 international conference on software engineering research and practice (SERP'04). Las Vegas, Nevada, USA; 2004.

Fernández-Medina E, Piattini M. Designing secure database for OLS. In: Database and expert systems applications: 14th international conference (DEXA 2003). Prague, Czech Republic: Springer; 2003.

Fernández-Medina E, Trujillo J, Villarroel R, Piattini M. Extending the UML for designing secure data warehouses. In: 23rd international conference on conceptual modeling ER2004. Shanghai, China: Springer-Verlag; 2004.

Georg G, Ray I, France R. Using aspects to design a secure system. In: Eighth IEEE international conference on engineering of complex computer systems (ICECCS'02). Greenbelt, Maryland, USA; 2002.

Ghosh A, Howell C, Whittaker J. Building software securely from the ground up. IEEE Software 2002;19(1):14—6.

Hall A, Chapman R. Correctness by construction: developing a commercial secure system. IEEE Software 2002;19(1): 18—25.

IBM. Security: IBM to provide multilevel security on the zSeries mainframe; 2004.

ISO/IEC. ISO/IEC 15408-1. Information technology. Security techniques. Evaluation criteria for TI security. Part I: introduction and general model. Switzerland; 1999.

Jacobson I, Booch G, Rumbaugh J. The unified software development process. Addison Wesley; 1999.

Jürjens J. UMLsec: extending UML for secure systems development. In: Jézéquel J, Hussmann H, Cook S, editors. UML 2002 — the unified modeling language, model engineering, concepts and tools. Dresden, Germany: Springer; 2002:412—25.

Khwaja A, Urban J. A synthesis of evaluation criteria for software specifications and specification techniques. International Journal of Software Engineering and Knowledge Engineering 2002;12(5):581—99.

Levinger J. Oracle label security. Administrator's guide. Release 2 (9.2). Available from: <http://www.csis.gvsu.edu/GeneralInfo/Oracle/network.920/a96578.pdf>; 2002.

Liu L, Yu E, Mylopoulos J. Security and privacy requirements analysis within a social setting. In: 11th International requirements engineering conference. IEEE Computer Society; 2003.

Marks D, Sell P, Thuraisingham B. MOMT: a multi-level object modeling technique for designing secure database applications. Journal of Object-Oriented Programming 1996;9(4): 22—9.

McGraw G. Penetrate and patch is bad. IEEE Software 2002; 15.

Priebe T, Pernul G. Towards OLAP security design — survey and research issues. In: 3rd ACM international workshop on data warehousing and OLAP (DOLAP'00). Washington DC, USA; 2000.

Priebe T, Pernul G. A pragmatic approach to conceptual modeling of OLAP security. In: 20th International conference on conceptual modeling (ER 2001). Yokohama, Japan: Springer-Verlag; 2001.

Siponen M. Designing secure information systems and software (academic dissertation). In: Department of information processing science. OuLu, Finland: University of Oulo; 2002.

Thuraisingham B, Schlipper L, Samarati L, Lin TY, Jajodia S, Clifton C. Security issues in data warehousing and data mining: panel discussion. Database Security XI: status and prospects 1998;3—16.

Vivas JL, Montenegro J, López J. Towards a business process-driven framework for security engineering with the UML. In: Information security, 6th international conference (ISC 2003). Bristol, UK: Springer-Verlag; 2003.

Rodolfo Villarroel is MSc in Computer Science from the Universidad Técnica Federico Santa María (Chile), and a PhD student at the Escuela Superior de Informática of the Universidad de Castilla-La Mancha at Ciudad Real (Spain). He is Assistant Professor in the Computer Science Department of the Universidad Católica del Maule (Chile). His research activities are security in data warehouses and information systems, and software process improvement. He is author of several papers on data warehouses security and improvement of software configuration management process. He belongs to the Chilean Computer Science Society (SCCC) and the Software Process Improvement Network (SPIN-Chile).

Eduardo Fernández-Medina is PhD and MSc in Computer Science. He is Assistant Professor at the Escuela Superior de Informática of the Universidad de Castilla-La Mancha at Ciudad Real (Spain). His research activities are security in databases, data warehouses, web services and information systems, and also in security metrics. He is the co-editor of several books and chapter books on these subjects, and has several dozens of

papers in national and international conferences. He participates at the ALARCOS research group of the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. He belongs to various professional and research associations (ATI, AEC, AENOR, IFIP WG11.3 etc.).

Mario Piattini is MSc and PhD in Computer Science from the Politechnical University of Madrid. He is a certified information system auditor by ISACA (Information System Audit and Control Association). He is Associate Professor at the Escuela Superior de Informática of the Castilla-La Mancha University (Spain). He is author of several books and papers on databases, software engineering and information systems. He leads the ALARCOS research group of the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. His research interests are: advanced database design, database quality, software metrics, object-oriented metrics and software maintenance.