Towards a Business Process-Driven Framework for Security Engineering with the UML

José L. Vivas[1], José A. Montenegro[2], and Javier López[2]

[1] Hewlett Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ, UK
jose-luis.vivas@hp.com
[2] Computer Science Department
E.T.S. Ingeniería Informática
Universidad de Málaga, Spain
{monte, jlm}@lcc.uma.es

Abstract. A challenging task in security engineering concerns the specification and integration of security with other requirements at the top level of requirements engineering. Empirical studies show that it is commonly at the business process level that customers and end users are able to express their security needs. In addition, systems are often developed by automating existing manual business processes. Since many security notions belongs conceptually to the world of business processes, it is natural to try to capture and express them in the context of business models in which moreover customers and end users feel most comfortable. In this paper, based on experience drawn from an ongoing work within the CASENET project [1], we propose a UML-based business process-driven framework for the development of security-critical systems.

1 Introduction

The view that the main challenges facing the development of secure systems are non-technological in nature is now widespread. Security involves people and organizations, relationships among people, and relationships between people and systems. Accordingly, the security of a computer system is almost always compromised by exploiting vulnerabilities in the environment into which they are inserted and the way they are employed, not by breaking dedicated mechanisms such as cryptographic protocols [3]. Nevertheless, work in this area has focused mostly on the technological part of the subject, such as cryptography, security protocols, firewalls, etc., resulting in what some authors have called the "technology trap" [9]. This discrepancy may be due to the fact that these technologies are much easier to understand and to secure [29]. In fact, results in the area of cryptography and cryptographic protocols are now extensive and well-established. Notwithstanding, there is little systematic support for software engineers to design secure systems [32], and the specification of security policies are typically not integrated with general software development. Security is usually added as an afterthought to system requirements and design [8].

However, the emphasis is now shifting away from technology to understanding the nature of the organization [9]. There is correspondingly also an increasing awareness that security concerns should inform every phase of software development, from requirements engineering to deployment [8]. Currently security concerns are specified in terms of highly specialized models that are not integrated with software engineering [4].

The most challenging task in security engineering seems to be at the top level of requirements engineering [26]. It is also at the most abstract level that customers and end users are able to express their security needs. Empirical studies show that some form of business process review generally occurs among engineers and stakeholders in order to develop a common understanding on the security needs of a system [34]. A formalization of this activity is thus called for.

We intend to develop a UML-based framework for representing security semantics in an integrated development environment. The focus is on the system level, including the environment or context into which the software system will eventually be inserted. We illustrate the approach by some examples extracted from a case study. The present paper is tentative and exploratory, and focuses on a discussion and identification of the problems rather than in providing solutions.

Two different lines of investigation are under development. The first one concerns a translation from UML into XMI intended to facilitate the use of security patterns to automate parts of the development process. The idea is to make extensive use of security patterns in order to integrate security requirements, expressed e.g. by means of UML extension mechanisms [19], into a functional description of the system given by UML diagrams. The procedure is performed with the help of a repository of previously worked solutions to security problems. The emergence of a new generation of tools and XML-based notations for business process management makes this approach a very promising one. Patterns [15] are intended to capture expertise in the form of worked solutions to recurring problems. Security patterns in particular [30] are intended to capture security expertise in a constructive way and to make this expertise available to developers who are not security experts. Although most patterns are supposed to be design patterns, as noted in [7] patterns can be used at higher levels of abstraction. A library or repository of security patterns can thus be created in a machine-readable notation such as XMI, thereby making available to the designer a wide variety of previously worked security solutions.

The second line of investigation concerns a mapping of UML diagrams into a formal notation for purposes of consistency checking, validation, verification, and simulation. We believe that formal analysis is both feasible and useful already at this stage of system development. As a first approach we have selected ConGolog [16], a concurrent logic programming language based on the situation calculus [22]. A formalism based on first-order logic is suitable for the specification of a system at a high level of abstraction, as well as for reasoning, testing, validation and verification. ConGolog is a notation in which most of the features associated with UML diagrams, both static (state diagrams, objects and classes, relationships in general) and dynamic (activity diagrams, sequence diagrams, processes) can be encoded. We can express in ConGolog such notions as business processes, use cases, sequence diagrams, cryptographic protocols, messages and goals. ConGolog is also suitable for expressing agent-related concepts such as actors, actions and roles, as suggested in [21]. This work presents a framework within which also the notions of knowledge and knowledge-producing action are formalized. Such concepts are more relevant that object-oriented ones at the initial stages of system development, even in a UML-based framework. Moreover, security requirements should be considered before the elaboration of design-level class diagrams, and this should be done in a more abstract setting than an object-oriented one.

As shown in [17], cryptographic protocols may also be defined and analyzed in the situation calculus with the help of the notions of knowledge, knowledge-production actions and sensing defined in [28, 27].

With regard to business processes, it has been shown that they can be specified, synthesized, simulated, and tested for feasibility and consistency using ConGolog [20]. The basic concepts of this approach are *action, process, role, actor* and *goal*. The notions of actors and roles connect the process to the organizational model. Actions within a process are carried out in the context of an organizational role by actors. An action is defined by a set of preconditions and effects. A process is a first-order term of the language, a complex action which may take part in any kind of relation with e.g. goals, roles, resources, and other processes. A business process is defined as a network of actions performed in the context of organizational roles in pursuit of a set of goals. A process consists of a list of goals and a list of role definitions, and a role consists of a list of goals and a list of procedures for achieving these goals.

The expressiveness of the UML notation for specifying security properties has recently been the subject of hot debate among researchers. There is agreement that robust tools and techniques for enabling syntactic and semantic analysis of UML diagrams are essential. There is also widespread concern in relation to what appears to be shortcomings of UML, e.g. lack of formal semantics and of expressiveness for modelling security properties. However, UML is considered to be attractive to a broader community with less critical security requirements. Many researchers are currently developing methods to bridge the gap between UML and formal specification languages and analysis tools. Together with an extensive tool support, UML has become a de facto industry standard, and the development of techniques for enabling UML users to apply formal analysis tools also talk in favor of UML. Moreover, the simplicity of UML modelling notations facilitates the capture of abstractions for a variety of domains, including security-critical systems. Most importantly, UML offers an opportunity for researchers to apply formal methods on a top-down basis, thus increasing the probability that these methods be adopted in an industrial setting.

The rest of this paper is organized as follows. Sect. 2 discusses the relation between process modelling and security. Sect. 3 gives a brief account of the role that XMI is intended to perform in our framework. Sect. 4 is dedicated to the presentation of a case study, the PASEN application, and Sect. 5 to the conclusions.

2 Security semantics in use cases and business processes

We propose in this study 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 the applications that concern us, e.g. 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. In the context of computer automation and communication systems these notions may acquire a new dimension, but scarcely a new meaning. It is thus natural to try to capture these security requirements within the business model, where they conceptually belong. Moreover, nowadays we see the automation of business activities and processes as such, not only support for part of those activities or for their information aspects. As a result, the security notions associated with business processes are themselves becoming the focal point of system security. This approach is further underpinned by the recent interest in letting business models not only provide the starting point for a system model, but also become an integral part of it.

With the aim of facilitating its adoption by system developers, our framework intends to integrate security requirements into standard system development methodologies, which currently is often UML-based and use case-driven. However, since we are advocating a business modelling approach, and since security requirements are regarded as non-functional whereas use cases are originally intended for capturing functional requirements, two questions arise here. The first one concerns the relation between use cases and business processes, and the second that between security and functional requirements.

With relation to the first question, we note that use cases are associated with the notion of scenario and sequence diagrams, whereas business process are based on workflow models and activity diagrams . However, at the most abstract levels of system specification these two kinds of diagrams are closely related, and one may be derived from the other in a variety of ways. In fact, business processes have been defined in ConGolog [20] basically in terms of agents, roles, and actions. A recent work [5] show how the two techniques can be integrated. Therefore, in practice no basic difference seems to exist between the two concepts. Sequence diagrams focus on the *interactional behavior* of agents, and disregard the activities that as a rule must be performed between interactions. Activity diagrams, by contrast, concentrate on the activities to be performed and the temporal and causal relations among them, and often disregard interactions.

We turn now to the second question. Use cases are often intended to capture the behavioral requirements of software systems, and to drive the development process. A use case is an interaction between a user of a system and the system itself. Use cases are intended to capture the *functional* requirements of a system. By contrast, *non-functional requirements* are defined as requirements that specify "system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensiveness, and reliability," i.e. the "ilities" [18]. Most non-functional requirements are largely orthogonal to the functional ones. Thus, a system might be made more or less reliable, resilient, flexible or extensible, without affecting the system's basic functionality. However, this is not the case with security requirements. Although typically defined as non-functional, and often treated together with non-functional requirements in system development methods [24, 6], in our opinion this categorization obscures more that it reveals. Security requirements are a very special kind of non-functional requirements, and are more closely related to functional than non-functional ones. We prefer to say instead that security requirements are *anti-functional*, rather than non-functional, thus highlighting the special relationship between security and functional requirements. Security requirements commonly have a visible impact on the functionality of a system, and should thus be analyzed together with the functional requirements.

The latter is corroborated by the fact that in practice many kinds of security requirements are studied in the context of dynamic and behavioral aspects of a system, e.g. role-based access control [4]. Even threat analysis is currently being performed in the context of use cases and scenarios, cf. *misuse cases* [31] and *abuse cases* [23]. In fact, uses cases are being used for similar purposes even outside of its intended area of application, cf. their use to identify hazards in aerospace as *failure cases* [2]. Hence, far from being an impediment to the integration of security requirements, use cases offer in fact an opportunity for their elicitation and specification, a felicitous fact that should not be neglected.

The Eriksson-Penker extensions for business modelling with UML [12] offers a suitable starting point for our approach. According to the authors, modelling business surroundings involves answering the following questions: (i) how do different actors interact; (ii) what activities are part of their work; (iii) what are the ultimate goals of their work; (iv) what other people, systems or resources are involved that do not show up as actors to this specific system; and (v) what rules governs their activities and structures. The answers to these questions are important for the security aspects of a system.

We should mention here one comprehensive approach aiming at modelling the security requirements of business processes and transactions and based on the notation ALMO$T [25]. ALMO$T was intended to enable a firm to formulate market transactions security at a high abstraction level. The basic approach is to consider the security requirements as immanent to business transactions and dependent on the circumstances of the applications. A three-layered architecture is proposed, with graphical concepts to specify security requirements at the upper layer, a repository of already modelled solutions of basic security elements at the middle, and a repository with hard- and software building blocks at the lowest layer. Four perspectives are given to produce a comprehensive view of a business process: informational, functional, behavioral, and organizational, to which a fifth new perspective is added, the business process perspective. Security requirements specified in the business process model must also be represented and analyzed in the other models. The approach we take here is similar, with a shift of emphasis towards requirements engineering, UML, use case-driven development, security patterns, and formal analysis. Each one of the perspectives mentioned above can be defined in terms of different types of UML diagrams, and the business process model in terms of the process diagrams of the Eriksson-Penker model [12]. The UML notation offers an integrated framework within which most notions associated with the ALMO$T approach can be expressed.

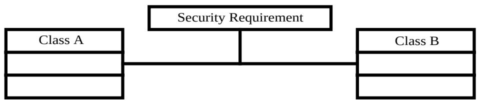
3 Representing UML in XMI

XMI (XML Metadata Interchange) [10] is a notation that is closely related to UML. It is used to give a textual XML representation to UML constructs, allowing automatic translation from a UML specification to other notations, for example to a formal language for the sake of model verification.

With regard to the security requirements, they should be introduced in a specification using the expansion mechanism of UML. These mechanisms are represented as labels in the XMI representation of a UML specification. Figure 1 shows a possible specification of a security requirement in UML using tags, and the corresponding XMI representation. The most important elements in the processed task are the labels *xmi.id*, which associate classes and security requirements. Thus, the XMI parser needs to process these labels to interpret the document and extract the corresponding security information.

The following points specify the functions that XMI is intended to perform in our framework:

1. Verify the consistency of the specifications associated with distinct UML views and abstraction levels. The possibility to describe a system in UML at several views and levels of abstraction allows different designers to work separately in the specification of the security requirements. Later, when the distinct views have been designed, a mechanism is needed to verify the consistency of the requirements across these views. For example, if we define a security requirement in a package, this requirement must show latter when we define the class that implements this package. XMI permits translation of a UML model to a checker, a non-UML tool, that will perform this task. This checker searches the labels that represent the security requirements in the XMI file and verifies the relation among them.
2. Use of formal method. Currently the less abstract views in a UML specification can be used to produce automatically the source code in various languages as Java and C++ for later implementation. Following the same lines, we propose to translate the model to formal languages for verification purposes. There is a semi-formal notation associated with the UML standard, the OCL (Object Constraint Language) [11], but this is a limited



```
<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.2" xmlns:UML="org.omg.xmi.namespace.UML" >
    <XMI.header>
        <XMI.documentation>
            <XMI.exporter>Netbeans XMI Writer</XMI.exporter>
            <XMI.exporterVersion>1.0</XMI.exporterVersion>
        </XMI.documentation>
    </XMI.header>
    <XMI.content>
                <UML:Class xmi.id="a4" name="Class A" visibility="public" ...... >
                     ...................
                </UML:Class>
                <UML:Class xmi.id="a6" name="Class B" visibility="public" ...... >
                     ...................
                </UML:Class>
                <UML:Association xmi.id="a8" isSpecification="false" .....>
                     ...................
                        <UML:TaggedValue xmi.id="a10"  dataValue="Security Requirements">
                            <UML:TaggedValue.type>
                                <UML:TagDefinition xmi.idref="a11"/>
                            </UML:TaggedValue.type>
                        </UML:TaggedValue>
                     ...................
                </UML:Association>
                     ...................

        <UML:TagDefinition xmi.id="a11" name="CASENET" isSpecification="false" tagType="CASENET">
                     ...................
        </UML:TagDefinition>
    </XMI.content>
</XMI>
```

Fig. 1. Security Requirement in UML and its XMI translation (simplified)

language for certain tasks. The model translation to XMI allows one to establish a standard interface to translate a UML specification to any formal language.

3. Apply previously worked solutions to security requirements specified in a UML model. The textual specification of UML designs, XMI, facilitates the database management of security design patterns. Currently there is a tendency to specify the design patterns in XML, and we are working in the specification of a XML schema [13] to represent security patterns in XML. This representation results in a XML document extended with labels for expressing security notions, in accordance with the scheme proposed in [14]. The representation of patterns in XML is intended to enable the interaction of the model and the repository of patterns, and to support the application of techniques such as composition and inheritance of patterns.

4 A case study: PASEN

PASEN is an e-government application being developed by SADIEL, Spain, and intended to enable the teachers, administrators and student tutors within an educational center to manage their communication needs via a web portal. The services to be offered to teachers, administrative staff, parents and students, include basic services such as pre-admission, student's registration, and grant management. One the main objectives of the final system will be to facilitate the tracking of the status of submitted requests, e.g. grant requests. The PASEN core activities that need to be secured include student registration, grant management, pre-admission requests, homework, student absences, and examinations. Each user requires a profile to access to the PASEN portal. The main different profiles are tutors, teachers and several categories of administrators.

We concentrate here on the grant management service. This activity consists of a part dealing with grant application and another with the grant follow-up. The tutor may apply for a grant by filling in and submitting a grant form. Once submitted, he receives a reference number that can be used to obtain the information from the grant follow-up, which is provided by the administrator of the educational center. A sequence diagram for this service is shown in Figure 2. The steps in the sequence diagram are explained in Figure 3. As we may observe, this diagram does not specify any security requirements, only the functional ones. It corresponds to a high level use case or a core business process, and must be decomposed several times during the software development process. This type of representation is one we may expect from system developers, and offers a good setting for studying and eliciting the required security properties of the system.

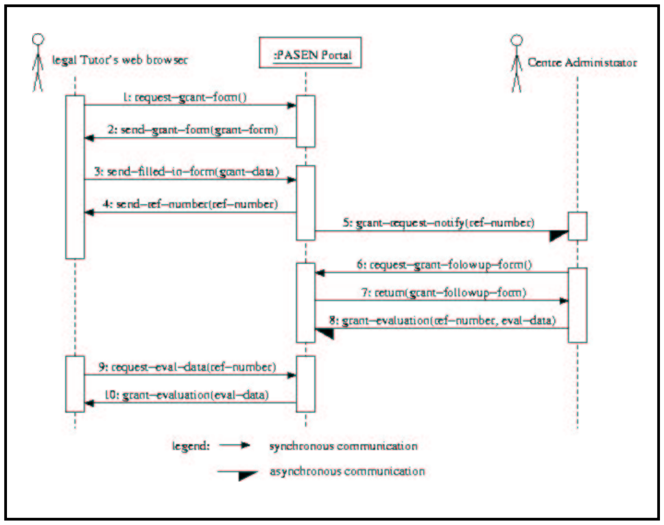

legal Tutor's web browser
:PASEN Portal
Centre Administrator
1: request–grant–form()
2: send–grant–form(grant–form)
3: send–filled–in–form(grant–data)
4: send–ref–number(ref–number)
5: grant–request–notify(ref–number)
6: request–grant–folowup–form()
7: return(grant–followup–form)
8: grant–evaluation(ref–number, eval–data)
9: request–eval–data(ref–number)
10: grant–evaluation(eval–data)
legend:
synchronous communication
asynchronous communication

Fig. 2. Sequence diagram for grant management

The security requirements associated with this service turn out to be extensive. Some are of a more general character, while others are related to the information exchanged during interactions: the grant application form, the grant data returned in the application (the filled application form), the reference number, and the evaluation data. In a general manner for the grant management, we need: (i) a security service allowing the notarization for each type of non-

1. The PASEN User (PU) using a web browser (WB) requests from the PASEN Portal (PP) a grant application form.
2. PP sends to WB a grant application form.
3. WB sends to PP the grant data, i.e., the filled out grant application form plus any other needed document.
4. PP sends to WB a reference number to be used as an identification of the submitted grant application.
5. PP sends the grant data to the Center Administrator (CA) in charge.
6. CA requests a grant follow-up form from PP.
7. PP sends to CA the grant follow-up form.
8. CA sends to PP result from the evaluation of the grant, i.e. the evaluation data.
9. PU via the WB requests grant information from PP.
10. PP sends to PU the grant evaluation.

Fig. 3. Steps of the sequence diagram

repudiation proof, and (ii) a security service that provides a unique temporal reference. More particular requirements are as follows.

The integrity of the application form is required. Therefore, we need a security service that allows verifying if an unauthorized modification of information (including changes, insertions, deletions, and duplications) has not occurred either maliciously or accidentally.

Concerning the grant data, both integrity and confidentiality are required. As a result we need a security service here that apart from guaranteeing data integrity, as above, also avoids any unauthorized access or any disclosure of information. It should also be possible to verify whether any form of unauthorized modification (including changes, insertions, deletions, and duplications) has occurred either maliciously or accidentally. Moreover, the different actors involved should be able to sign the grant file because this file is regarded as a specific contract. Once signed, it is required that the contract cannot to be modified. Finally, a security service is required that provides a proof of the grant evaluation submission. The reference number should also be protected. Integrity and confidentiality are both required, as well as proof of reception of the reference number.

Although the application is rather standard and simple, the security requirements, taken together, are very complex. The requirements are hard to meet if they are added as an afterthought to the developed system. The complexity increases if moreover flexible solutions are required, e.g. if several similar systems exhibiting different sets of security requirements should be developed. In addition, the requirements may change during the lifetime of the deployed system. The need for a precise method to develop this kind of security-critical system becomes evident here.

As we may observe, the security requirements of the PASEN application refer to several perspectives. These perspectives can be associated with different types of UML diagrams. Hence, the functional perspective corresponds basically to activity, use case, and sequence diagrams. The static perspective corresponds mainly to class diagrams, the dynamic perspective to state charts, the organizational perspective to lanes in activity diagrams, class diagrams and packages, and the business process perspective to process models. All these diagrams can be affected by the security requirements. Our approach is to begin by encoding the functional aspects of the system in the different diagrams, and then to extend these diagrams in a variety of ways in order to express the corresponding security requirements. These extensions should be easily understandable by domain experts, and should as far as possible be based on standard definitions of security concepts. As an example, a security enriched use case, using a scheme shown in [33], could turn out to be as shown Figure 4.

Use case: Grant Management

Functional Summary: A tutor requests a grant and an administrator of the educational center returns a grant evaluation.

Actor: tutor, administrator.

Security subject: tutor: authenticity.

Preconditions: Tutor authenticated.

Security objects:

- grant form: integrity
- grant data: integrity and confidentiality
- reference number: integrity, confidentiality, non-repudiation of reception by tutor
- evaluation data: integrity, confidentiality, authentication of origin, non-repudiation of reception and submission

Fig. 4. Security enriched use case

The corresponding business process diagram integrates several perspectives and includes an input object, an input event executed by a tutor, and input data. The output is a grant evaluation. Omitting many details, the purely functional version of the Grant Management process, i.e. without the security requirements, could be schematically represented as shown in Figure 5. The two subprocesses in the grant management process correspond to the Grant Application resp. the Grant Follow-up subprocesses.

The Grant Management process extended with security could be described as shown in Figure 6, where irrelevant details are left out. In this diagram we introduce a new element, stereotyped by <<security>>, which can be interpreted as a kind of security goal. In the lower rectangle of each element we include the parameters associated with the security goal, stated in an informal way.

The next step would be to integrate the security goals into the business process itself, which might result in extensive changes to the previous flow of events and activities, including the addition of new subprocesses. Non-repudiation, for instance, may require the inclusion of new actors, e.g. trusted third parties, and new types of information objects and activities may also be required. The re-

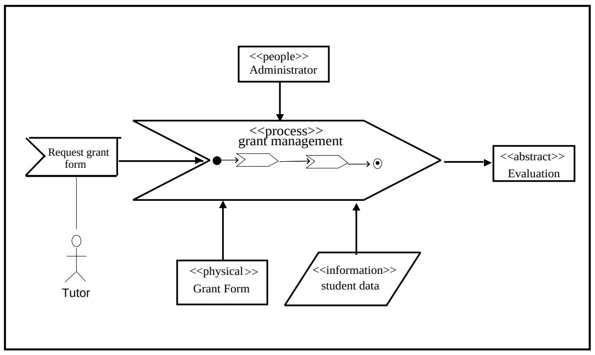
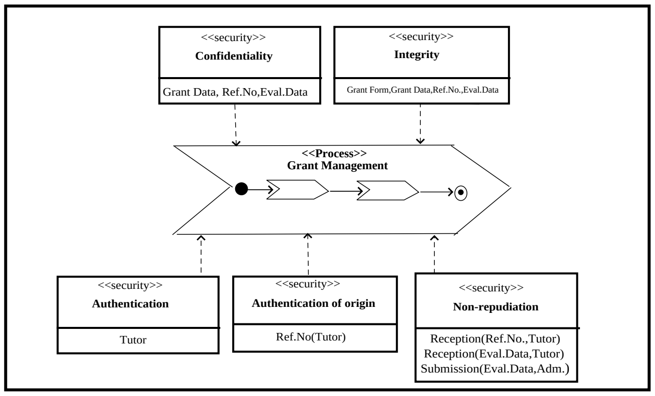


Fig. 5. Grant management business process

sulting business process should ideally be based on solutions contained in the repository, and be the result of a pattern-based analysis applied to the security-enhanced business process shown in Figure 6, or rather to its XMI representation. A similar procedure, with similar results, can be applied to the corresponding sequence diagram, thus yielding a new sequence diagram that might include e.g. communication between the original parties and a trusted third party, e.g. an e-notary. Other diagrams corresponding to distinct perspectives may also require changes in the same spirit.



<<security>> Confidentiality
Grant Data, Ref.No,Eval.Data
<<security>> Integrity
Grant Form,Grant Data,Ref.No.,Eval.Data
<<Process>> Grant Management
<<security>> Authentication
Tutor
<<security>> Authentication of origin
Ref.No(Tutor)
<<security>> Non-repudiation
Reception(Ref.No.,Tutor)
Reception(Eval.Data,Tutor)
Submission(Eval.Data,Adm.)

Fig. 6. Security enriched grant management

A process corresponds to a procedure defined at some level of abstraction, and this procedure may change during the lifetime of a system. A subprocess needs not represent a purely internal activity, it may very well communicate with agents outside the organization, and may even be allocated to a party outside the organization. The advantage of having a uniform representation of the several views of the system and its parts is that inconsistencies arising from these complex phenomena might become visible at any stage of the software development life cycle.

An important question here is how the security goals associated with business processes relate to process decomposition. In Figure 6 we have a process, Grant Management, which consists of two subprocesses in sequence. Several security goals are associated with the parent process. The goal of authenticity can be assumed to apply to the two subprocesses in the same way as to the parent process. By contrast, non-repudiation is a kind of security goal that does not decompose into two identical goals, one for each subprocess. It may require a complete solution at the parent process level. There is no general solution to the process decomposition problem with regard to with security requirements. Only a case by case study of each security goal, supported possibly by a repository of previously worked solutions and a pattern-based analysis tool, seems to be feasible here. This is part of the XMI-based design procedure proposed in this study.

Once the business model is encoded into ConGolog, the main challenge is to express in this notation the security goals associated with the UML business process model, which in our example include non-repudiation, authentication, integrity, and confidentiality.

A domain expert can easily understand the security requirements expressed in the form exemplified in Figure 6, and might even have created them. It is however uncertain whether the developer would like to go further than this in the specification of the system requirements. The requirements might also have been partially generated from other views of the system, and consistency across different perspectives should be checked at this stage. The semantics of the expressions related to security should be as precise as possible, ideally based on some standard. The resulting solution should in any case be checked by the domain expert e.g. for validation, since it will probably have an impact on the overall functionality of the system and affect other requirements. Also, several forms of threat analysis can also be performed at this stage, for instance those based on use cases or scenarios as explained in [31, 23]. The final result is a specification of the system at a high level of abstraction including the security requirements, which becomes an input to the next stage of system development.

5 Conclusions

In this exploratory study we have presented an ongoing work intended to establish a use case-driven software development framework based on the UML, and to integrate security requirements into a business process model of the system.

The UML is extended in order to express security notions. Use cases and the corresponding scenarios are used as the basic tool to build threat models and elicit security requirements. The latter are originally stated at a 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 the security requirements are integrated into the functional description by means of a pattern-based analysis and design process, yielding a new specification of the system into which the security requirements have been integrated. The resulting representation is translated into a formal notation like ConGolog for testing, validation and verification. This procedure is iterated as many times as required. The result is used as input to the following stages of system development.

Acknowledgements

This work was funded by CASENET European Project IST-2001-32446. Special thanks to Sabine Delaitre and Guillermo León at Sadiel, Seville, Spain.

References

1. *http://www.casenet-eu.org.*
2. K. Allenby and T. Kelly. Deriving safety requirements using scenarios. In *Fifth IEEE International Symposium on Requirements Engineering (RE '01)*, pages 228–235. IEEE Computer Society Press, 2001.
3. Ross J. Anderson. Why cryptosystems fail. *Communications of the Association for Computing Machinery*, 37(11):32–40, November 1994.
4. Gerald Brose, Manuel Koch, and Klaus-Peter Löhr. Integrating security policy design into the software development process. Technical Report B-01-06, Institut für Informatik,Freie Universität Berlin, November 2001.
5. M. Chaudron, K. van Hee, and L. Somers. Use cases as workflows. *Lecture Notes in Computer Science*, 2678:88–103, 2003.
6. L. Chung. Dealing with security requirements during the development of information systems. *Lecture Notes in Computer Science*, 685:234–??, 1993.
7. P. Coad. Object-oriented patterns. *Communications of the ACM*, 35(9):153–159, September 1993.
8. Premkumar T. Devanbu and Stuart G. Stubblebine. Software engineering for security. In *Proceedings of the 22th International Conference on Software Engineering (ICSE-00)*, pages 227–240, NY, June 4–11 2000. ACM Press.
9. Gurpreet Dhillon. *Managing Information System Security.* Macmillan education, Limited, 1997.
10. OMG Document. *OMG XML Metadata Interchange (XMI)Specification, v1.2*, 2002.
11. OMG Document. *Initial Response to UML 2.0 OCL RFP ad/ 00-09-03 (UML 2.0 OCL)*, 2003.
12. Hans-Erik Eriksson and Magnus Penker. *Business Modelling with UML: Business Patterns at Work.* John Wiley & Sons, 2000.
13. David C. Fallside. *XML Schema, W3C Recommendation*, 2001.

14. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstraction and reuse of object-oriented design. *Lecture Notes in Computer Science*, 707:406–431, 1993.
15. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, Reading, Mass., 1995.
16. Giuseppe De Giacomo, Yves Lésperance, and Hector J. Levesque. ConGolog, A concurrent programming language based on situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.
17. J. Hernández and J. Pinto. Especificación formal de protocolos criptográficos en cálculo de situaciones. *Novatica*, 143:57–63, 2000.
18. Ivar Jacobson, James Rumbaugh, and Grady Booch. *The Unified Software Development Process*. Object Technology Series. Addison-Wesley, Reading/MA, 1999.
19. Jan Jürjens. Towards development of secure systems using UMLsec. *Lecture Notes in Computer Science*, 2029, 2001.
20. M. Koubarakis and D. Plexousakis. A formal framework for business process modeling and design. *Information Systems*, 27(5):299–319, 2002.
21. Yves Lespérance, Hector J. Levesque, and Raymond Reiter. A situation calculus approach to modeling and programming agents. In Michael J. Wooldridge and A. Rao, editors, *Foundations of Rational Agency*, pages 275–299. Kluwer Academic Pub;ishers, Dordrecht, 1999.
22. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.
23. J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *15th Annual Computer Security Applications Conference, (ACSAC '99)*, 1999.
24. John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, June 1992. Special Issue on Knowledge Representation and Reasoning in Software Engineering.
25. Alexander W. Röhm, Gaby Herrmann, and Günther Pernul. A language for modelling secure business transactions. In *IEEE Annual Computer Security Application Conference (ACSAC'99)*, Phoenix, USA, December 1999.
26. John Rushby. Security requirements specifications: How and what? In *Symposium on Requirements Engineering for Information Security (SREIS)*, Indianapolis, IN, March 2001.
27. R. Scherl, H. Levesque, and Y. Lesprance. The situation calculus with sensing and indexical knowledge. In Moshe Koppel and Eli Shamir, editors, *Proceedings of BISFAI'95: The Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence*, pages 86–95, Israel, 1995. Ramat Gan and Jerusalem.
28. Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In Richard Fikes and Wendy Lehnert, editors, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 698–695, Menlo Park, California, 1993. American Association for Artificial Intelligence, AAAI Press.
29. Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley and Sons, Inc., New York, NY, USA, 2000.
30. M. Scumacher and U. Roedig. Security engineering with patterns. In *Pattern Languages of Programs 2001*, Monticello, IL, 2001.
31. Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements by misuse cases. In *Proc. TOOLS Pacific 2000*, pages 174–183, November 2000.
32. Mikko Siponen. Designing secure information systems and software. Academic dissertation, University of Oulo, 2002.

33. Mikko Siponen and Richard Baskerville. A new paradigm for adding security into is development methods. In Jan H.P. Eloff, Les Labuschagne, Rossouw von Solms, and Gurpreet Dhillon, editors, *Advances in Information Security Management & Small Systems Security*, pages 99–111. Kluwer Academic Publishers, 2001.
34. R. Vaugh, R. Henning, and K. Fox. An empiricakl study of industrial security engineering practices. *Journal of Systems and Software*, November 2001.