Secure Information Systems Engineering: Experiences and Lessons Learned from two Health Care Projects

H. Mouratidis[1], A. Sunyaev[2], J. Jurjens[3]

[1]School of Computing and Technology, University of East London, England
haris@uel.ac.uk
[2]Institut fur Informatik, Technische Universitat Munchen, Germany
sunyaev@in.tum.de
[3]Computing Department, The Open University, Great Britain
j.jurjens@open.ac.uk

Abstract. In CAiSE 2006, we had presented a framework to support development of secure information systems. The framework was based on the integration of two security-aware approaches, the Secure Tropos methodology, which provides an approach for security requirements elicitation, and the UMLsec approach, which allows one to include the security requirements into design models and offers tools for security analysis. In this paper we reflect on the usage of this framework and we report our experiences of applying it to two different industrial case studies from the health care domain. However, due to lack of space we only describe in this paper one of the case studies. Our findings demonstrate that the support of the framework for the consideration of security issues from the early stages and throughout the development process can result in a substantial improvement in the security of the analysed systems.

1 Introduction

Current information systems contain a large number of important and sensitive information that needs to be protected. Therefore, the need to secure these systems is recognised by academics and practitioners alike. This is reflected in the current literature where it is now widely accepted [13] [5] that security should be embedded into the overall information systems development and not added as an afterthought. As a result, a number of researchers are working towards the development of modelling languages and methodologies that can support the consideration of security as part of the information systems development process, and various approaches coming from different schools of thought have been reported in the literature(see for example [13]). Along these lines, a number of model-based security engineering approaches have been proposed [8][1][2]. In such approaches, a model of the system is initially constructed and a corresponding implementation is derived from that model either automatically or manually. An important limitation of these approaches is the lack of consideration of the earlier stages of the development process, such as early requirements analysis. To overcome this issue, in previous work, which was presented in CAiSE 2006 [14], we integrated the UMLsec approach [8] with the

secure Tropos methodology [11]. The resulting framework allows the construction of an initial security requirements model that is constantly refined until a well defined model of the system has been constructed. In particular, the framework defines a set of guidelines and transformation steps to enable developers to "translate", in a structured manner, the initial high level security requirements models, defined in Secure Tropos, to a well defined design model defined in UMLsec. Our framework is different from other works [16][1][2][6][10] trying to integrate security considerations into the development lifecycle. Existing work is mainly focused either on the technical or the social aspect of considering security. Moreover, approaches are usually applicable only to certain development stages. In contrast our approach considers security as a two dimensional problem, where the technical dimension depends on the social dimension. Moreover, our approach is applicable to stages from the early requirements to implementation. The next two sections describe the application of the framework to the two industrial case studies.

In this paper we report on the application of our framework to an industrial case study for the development of a Telematics system at a German hospital. We then reflect on the applicability of this framework and our experiences from its applications to two industrial case studies from the health care domain, the described German Hospital Telematics case study and the Single Assessment Process of the English National Health Service (NHS) case study. The paper is structured as follows. Section 2 provides a summary of the main elements of the framework to assist readers not familiar with it. Section 3 discusses the case study and it demonstrates how our framework was applied and how the security of the Telematics system was improved. Section 4 reflects on the application of the framework. Our reflection is mainly sub-divided into three main areas: *Framework Development*, *Lessons Learned*, and *Improvements*. Section 5 concludes the paper.

2 Secure Tropos meets UMLsec: A model-based security aware framework

As mentioned above, the framework, under discussion in this paper, has been presented in CAiSE 2006 [14]. Therefore, the aim of this section is not to repeat the details of the framework but rather to summarise it, in order to enable the readers of the paper to understand the following sections. The security-aware process of the framework includes four main stages: *Security Analysis of System Environment*, *Security Analysis of System*, *Secure System Design*, and *Secure Components Definition*. In each of these stages a number of models are defined that are then refined in the later stages. In particular, the main aim of the first stage is to understand the social dimension of security by considering the social issues of the system's environment, which might affect its security. In doing so, the environment in which the system will be operational is analysed with respect to security. In particular, in line with the Secure Tropos methodology, the stakeholders of the system along with their strategic goals are analysed in terms of actors who have strategic goals and dependencies for achieving some of those goals. Then the security needs of those actors are analysed in terms of security-related constraints that are imposed to those

actors. Such analysis results into the Secure Tropos security-enhanced model. Then, for each of the actors depicted on the Secure Tropos security-enhanced model, security goals and entities are identified, in order to satisfy the imposed security constraints. This information is modelled with the aid of the Secure Tropos security-enhanced goal model. During the second stage, the technical dimension of security is analysed by employing modelling and reasoning activities similar to the ones used in the previous stage, but now the focus is on the system rather than its environment. The output of this stage is refined Secure Tropos security-enhanced actor and goal models. During the third stage, the aim is to define the architecture of the system with respect to its security requirements. To achieve this, a combination of Secure Tropos and UMLsec models are employed. The Secure Tropos security-enhanced actor and goal models are furthered refined and provide input to the Secure Tropos architectural style model, which defines the general architecture and the components of the system. The Secure Tropos models are then transformed to UMLsec Class and Deployment diagrams, which are used to model the security protocols and properties of the architecture. To support the transformation of the Secure Tropos to UMLsec models, the framework defines a set of guidelines and steps [14]. In particular, two main transformation guidelines have been defined along with eight steps that describe each of the guidelines in detail. During the fourth stage, the components of the system are identified in detail. To achieve this, UMLsec activity diagrams are used to define explicitly the security of the components and UMLsec sequence diagrams or state-chart diagrams are used to model the secure interactions of the system's components. For example, to determine if cryptographic session keys, exchanged in a key-exchange protocol, remain confidential in view of possible adversaries, UMLsec state-chart diagrams can be used to specify the security issues on the resulting sequences of states and the interaction with the component's environment. Moreover, the constraints associated with UMLsec stereotypes are checked mechanically, based on XMI output of the models and using sophisticated analysis engines such as model-checkers and automated theorem provers. The results of the analysis are given back to the developer, together with a modified model, where the weaknesses that were found are highlighted [9].

3 Case Study

The case study is based on experience during a project with healthcare professionals of the University Hospital in Munich, Germany. Some of the authors have long standing project relationships with this establishment and that relationship was the starting point of the project. The project involved around 13 people, including the hospital's head of the computer centre, a number of physicians, a data protection officer, and a number of computer scientists including some of the authors. Gradually every insured patient in Germany is to receive a new smart-card based patient card, which will replace the past insurance cards. This new electronic patient card will be able to carry administrative functions as well as control access to the health data of the patient. As such, the electronic patient card is the central part of a Telematics infrastructure which can provide access to multiple forms of information and can

store data locally. Besides the storage of data on the electronic patient card, other applications are possible. These applications include: drug order documentation, electronic physician letters, treatment cost receipts, emergency case data, general patient data, and an electronic health record. In accordance with the new German health reform, the next phase after the introduction of the electronic patient card will be the electronic patient document. For this reason, the goal is to realize a uniform Telematics platform as a communication turntable for all parties, involved in the health care industry. Many different aspects must be considered during the development and implementation of such a health care Telematics infrastructure. Due to ethical, judicial, and social implications, medical information requires extremely sensitive handling. Guaranteeing the protection of the patient-related information and the health care information-systems is becoming increasingly important. On the other hand, there is an acceptance problem on the part of the end users (patients, care providers, cost units). Data collection and requirements elicitation took place through analysis of existing specifications (that are confidential and cannot further discuss) and a number of interviews with the stakeholders. Our interviews with a number of health care professionals [17] revealed that the main problems were deficient communication between medical practices and hospitals and bad scheduling in hospitals. All interviewees identified an existent demand for IT support in health care networks. But at the same time they expressed some worries about the security level and dependability of using information systems.

Following the steps of our framework, it is important to understand the environment of the system and reason about the security constraints imposed by that environment to the various system stakeholders. To keep the analysis of the case study in a manageable length, for this paper, we focus our analysis on three main stakeholders: the *Patient*, the *Hospital* and the *Physician*. Security constraints related to the distribution of medical information are imposed by the environment (such as German health data protection laws) and also by the *Patient*. As mentioned above, a secure Tropos security-enhanced actor diagram is used to initially model this information, which is later refined by adding the system-to-be, as another actor who has dependencies with the existing actors. This model is shown in Figure 1. As shown in that figure, the *Physician* depends on the *Electronic System* to access patient records. However, there are a number of security constraints imposed both to the *Physician* and to the *Electronic System* for that dependency to be valid.

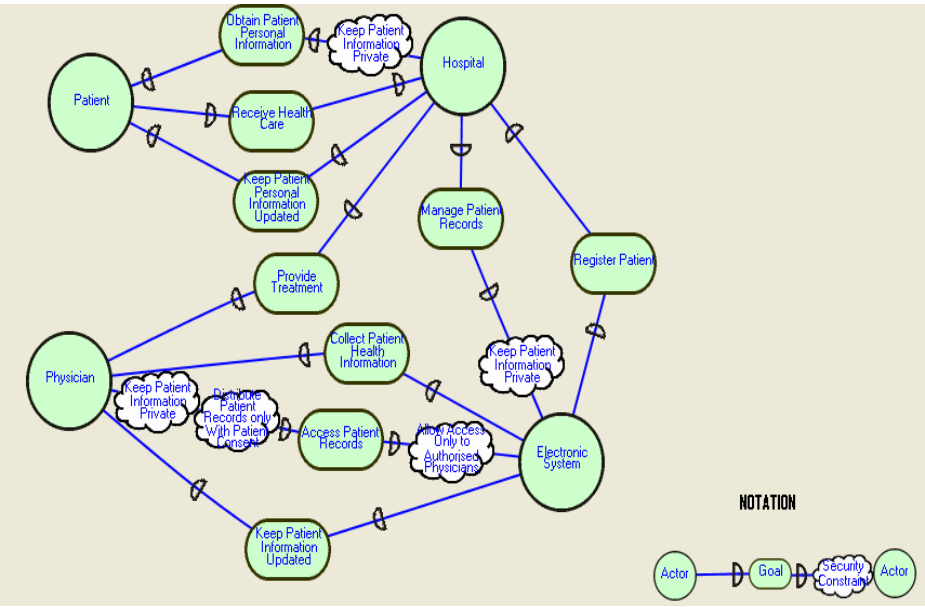

Figure 1: secure Tropos security enhanced actor diagram

The Secure Tropos security-enhanced actor diagram is furthered refined by analysing the internal goals of the *Electronic System*. This analysis results in the Secure Tropos security-enhanced goal diagram that models the various internal goals, tasks and security constraints of the system. In particular, our analysis indicates that for the system to satisfy its security constraints, various secure goals are introduced such as *Ensure System Privacy*, *Ensure Data Integrity*, *Ensure Data Availability*, *Ensure Secure Transfer of Records*. These abstract goals have been analysed further and appropriate secure tasks have been identified such as *Encrypt Data*, *Check Digital Signatures*, *Perform Auditing*, *Transfer Data through Virtual Private Network*, *Enforce Access Control* and so on.

When all the secure goals and secure tasks of the system have been identified, the main aim is the identification of a suitable architecture. The core idea of the physician-hospital architecture that was considered in this application is based on a separation of the central database into two independent, and stand-alone partial databases, whose linking returns the inquired answer, just as is the case with a central database. For the user of the system, the procedure remains transparent. Every kind of electronic communication between the medical practices and the hospitals is fundamentally based on one central storage and processing place: the core database. This core database contains and processes all organizational, administrative, and medical information about the patient. The idea of this architecture is to split this core database into two separate databases: first, the so-called "Metadatabase", and secondly, the "Hospital Information System database" ("HIS-database"). The "Metadatabase" contains all administrative data of the patient (name, first name, date of birth, address, insurance data etc.).

The "HIS-database" contains all medical data (like diagnostic images, data, pictures, treatment, medicines etc.) of the respective patient. This sensitive health information does not have a reference to the individual person; it is stored pseudonymously. Additionally, these two "records" possess an attribute named "ID". With its assistance, the combination of the two suitable entities (the administrative data of a patient and his/her health information) can be realized. The two databases are kept physically separate from each other. They are completely autonomous, i.e. there is no direct connection between the two databases. The access is gained through an encrypted connection and is possible to only one of the two databases at any given point in time.

Following the steps and transformation rules of the framework (see [14]) UMLsec deployment diagrams are constructed from the Secure Tropos models to represent the architecture defined in our analysis. When the components of the system have been defined, the next step involves the verification of the security of the modelled architecture. For this purpose, UMLsec sequence diagrams are employed and security properties, identified as important during the analysis of the system (modelled in Secure Tropos models), such as integrity, secrecy, authenticity are used to evaluate the architecture and to indicate possible vulnerabilities.

For example, consider figure 2 that illustrates the sequence diagram of the transmission of data between the user (e.g. Doctor) and the databases. It allows its secret information to be read using the operation *getMetaData()*, whose return value is also secret. That specification violates the security information flow requirement, since partial information about the time input from the higher sensitivity level operation *getMetaData()* is leaked out via the return value of the lower sensitivity level operation *getHISData()*.

However, our analysis indicated that in order to avoid such violation, the system's architecture should include a wrapper with a function of placing artificial inquiries to the databases. Artificial inquiries are constantly placed against the system in a way that does not simply place them sequentially after each other; instead, they overlap, at best several times over the entire time. Through this variation, it is impossible for the attacker to filter and/or further pursue individual inquiries. Thus, the problem of the possible time inquiries on the part of the attacker would be solved. The attacker is not able to plumb individual-queries and has thus no possibility thereby to extract the numerical data (time stamps).

The refined sequence diagram is shown in Figure 3. If a correct query is posed to the system, the wrapper simply continues to lead the query. If this is not the case, the wrapper generates more own queries. Each time, the wrapper examines whether a query is present, in order to be able to then act accordingly.

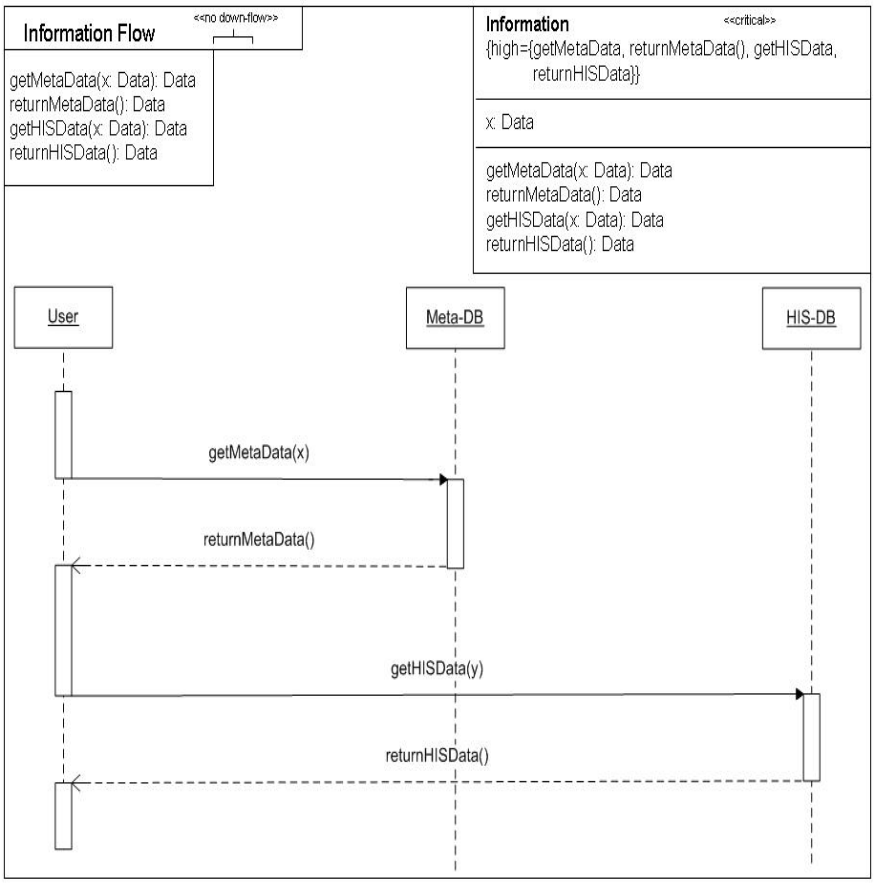

Figure 2: Sequence Diagram illustrating transmission of data

The wrapper constantly sends artificial inquiries to the HIS-Database, with each clock pulse the query is passed on. If a "correct" query against the system has been placed, the wrapper simply passes on the query. One does not know on which clock pulse the query falls. The allocation of a date to the appropriate operation sequence is thus not realizable. The attacker cannot find out which response belongs to which request, and accordingly, which queries from both databases belong together. With this discreteness, a more complex distribution for clock pulses is possible as well. With regards to security verification, there is only one way for the attacker to gain knowledge of the right assignment between the two databases (i.e. Meta-Database and HIS-Database).

The attacker could succeed if he can extract which tuple of information has been queried at which time. Then, the attacker could use this information to find out about the linking of the information between the two databases.

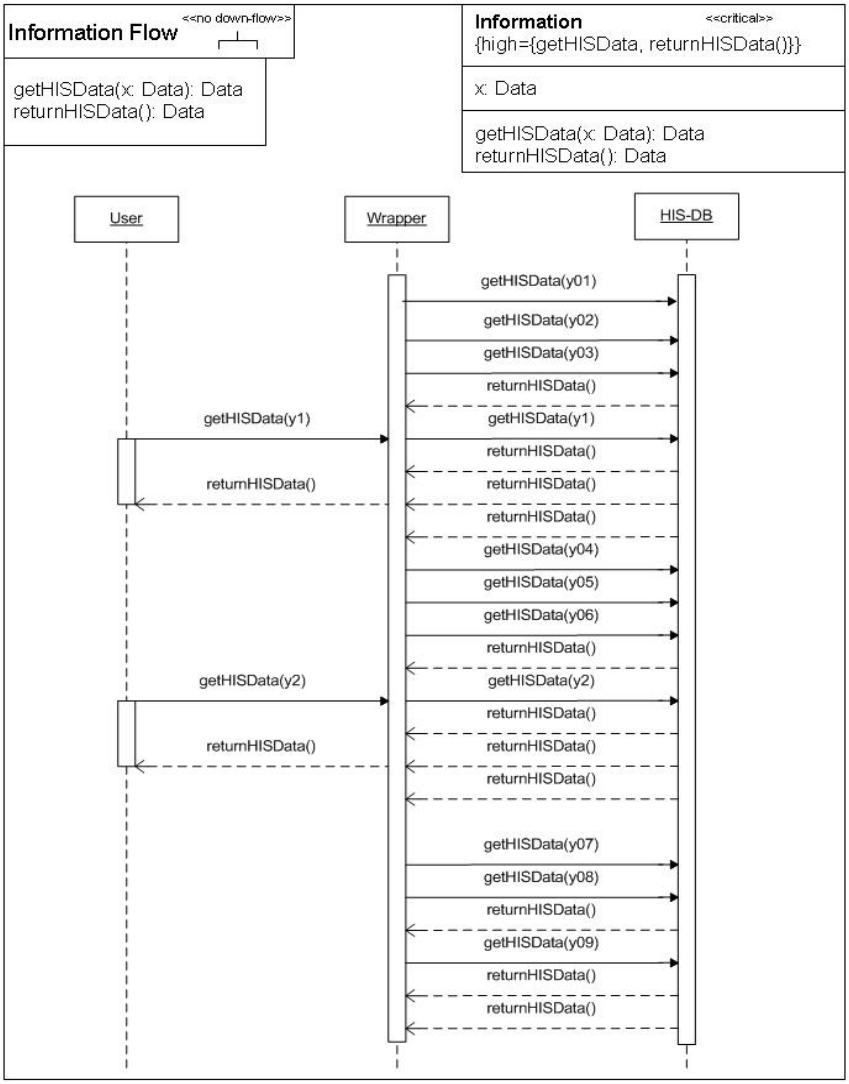

Figure 3: Refined Sequence Diagram

Here, the concern is about an indirect loss of information since following our analysis the system encrypts and securely keep the information. The date of the query procedure indirectly reveals partial knowledge of the confidential information. The attacker can possibly assume the information which was queried after each other and/or time near briefly by the two databases could belong together. The assumption is based on the fact that in the normal case, after the client requests something from the server, the server responds relatively timely. It is the same with the two databases, the HIS-Database and the Meta-Database. First, the user requests something from the

Meta-Database and then simultaneously (so the physician does not have to wait) the HIS-Database is queried. If the attacker wants to extract the combination of these two databases, he just has to wait from the Meta-Database-Request until the HIS-Database-Request and the subsequent answers and would then probably be able to extract the correct combination. In the above sequence diagram, the secret information is allowed to be read using the operation *getMetaData()*, whose return value is also secret. The data object is supposed to be prevented from indirectly leaking out any partial information about *high* via *non-high* data, as specified by the stereotype <<*no down-flow*>>. It is important that the observable information on the time of query allows no conclusions about the information that are being requested. Therefore, by applying the UMLsec tool suite [9], one can now make sure that the proposed design in fact fulfils its security requirements. More importantly by following the transformation rules and guidance of the framework, we are able to track specific security solutions (mechanisms) to specific security requirements.

The solution above developed improves on a number of security problems that current telematics platforms, such as HealthBase, TempoBy, MeDaCom, IHE, Intermediation platform, RITHME, PICNIC, and NHSnet, suffer. A large number of telematics platform exist. For our research, we have empirically compared a number of them, against the system developed by employing our framework. Most of the telematics concepts use the same security standards and techniques. The assurance of data security and data integrity is based on the electronic communication with the following six points: (1) There is only one central storage and processing place - the database in the clinic/hospital; (2) A special software is installed and implemented at all attached and entitled points of the network entrance; (3) There is a smart-card reader at each entitled point of network entrance; (4) The connection is based on a virtual private network. There is a VPN router and the required clients for it; (5) Hospitals as well as the medical practices communicate through special firewalls which have been devised for this kind of electronic communication; (6) The medical and administrative personal data is transmitted in encrypted form. Having just one, generally used, central database could be a possible security weakness open to exploitation, since all stored patient information is concentrated in a single location. Even though there are several security mechanisms in place for the protection of the records database as well as for the connection to and from the database; there are no further protection mechanisms for the data in the case of capture and decryption by an attacker. The eavesdropping or interception of the transmitted data could happen via interception of the transmitted packages as well as a direct hardware infiltration of the database connection within the hospital. An administrator, or a person who is responsible for the setup and maintenance of the database system, could be an attacker. The attacker could intercept the transmitted packages, save them locally in his hard drive and decrypt them without time- or place-restrictions. In the above mentioned Telematics platforms, the central database contains complete data for each patient. This means it contains purely administrative data as well as medical information about the patient. This is the reason, why the protection of the sensitive information and its access restriction/non-readability to attackers cannot be presupposed anymore. However, these issues are all dealt with by the architecture described in the previous section.

4 Reflection

In this section we reflect on the framework based on its application to the case study described above, as well as a second case study again from the health care domain. However, due to lack of space we cannot explicitly describe the application of our framework to the case study. The project involved health and social care professionals, such as General Practitioners and Nurses, specialist health professionals, such as Social Workers, and health care IT professionals. Its main aim was to analyse and specify the requirements of a software system to deliver the Single Assessment Process (SAP) [12], a health and social care needs assessment process for older people in England, with particular emphasis on its security. Two important conclusions were drawn. First of all, it became obvious from the discussions with various social care professionals and patients, that privacy was the number one security attribute required by this system. Secondly, the project identified the lack of security-aware methodologies that could assist developers in analysing the electronic Single Assessment Process (eSAP) system with security in mind.

So the following sections discuss our reflections from both these case studies. Our discussion is mainly sub-divided into three main areas: *Framework Development*, which discusses challenges faced during the development of the framework, along with the solutions provided at that point, and it reflects on whether the given solutions are satisfactory based on the application of the methodology to the case study; *Lessons Learned*, which discusses the lessons learned from applying the methodology to the case study; and *Improvements*, which provides an insight about how the methodology can be improved for the application in large industrial context.

4.1 Framework Development

In this sub-section we reflect on the issues and challenges we faced during the development of our framework and we discuss, by reflecting on the application of the framework to the case studies, whether the decisions taken at the framework's development stages were successful.

Challenge 1: Integration. A major challenge on the development of the framework was the seamless transition from the secure Tropos models to the UMLsec models.

Solution: To achieve the above challenge, we decided to employ a functional integration [15], where individual approaches' models stay intact and guidelines to translate the models from one approach to another and indicate the inputs and the outputs of these models are defined. A number of guidelines and steps were also defined to support the integration [14].

Reflection: The initial guidelines and the steps defined assisted in the translation of the models from secure Tropos to UMLsec. However, during the development of the Telematics system we identified some inconsistencies between the secure Tropos and the UMLsec models. By investigating this issue, we found out that the problem existed due to some errors in some of the guidelines. In particular, initially our guidelines suggested that actor related resources of the secure Tropos security-enhanced actor models should be translated in a 1-to-1 analogy to attributes of the UMLsec class models. However, by applying the framework to the Telematics case

study it became obvious that this was not the case, and in most of the cases, a resource will result in more than one attributes. This was mainly because the secure Tropos security-enhanced actor models contain analysis information whereas the class UMLsec models contain design information. Another issue that was raised during the application of such guidelines was the possible need to formalise them using a transformation language. This clearly constitutes area of future work.

Challenge 2: Process. An important issue of our work was the development of a process to support the development of framework.
Solution: We decided to base the development process on the Secure Tropos development process. In particular, the development process enables the construction of an early requirements model that is furthered refined, following a top-down approach, to a security model that is amenable to formal verification with the aid of automatic tools [9]. The refinement process is governed by a set of rules and activities [14]. It is worth mentioning that the process is highly iterative.
Reflection: The application of our framework to the case study indicated some problems. In particular, initially all the information from the early and late requirement models was effectively refined to the design models. However, during the application of the framework to the case study, it became apparent that this should not be the case. This was mainly because the analysis models contain reasoning information which should not be transformed to the design models. For instance, analysis models can contain information on different alternatives for satisfying a particular security constraint. At the initial development of the framework, we would transform all these alternatives to the design models. However, by applying the framework to the case study and when trying to transform all the reasoning information to design, we were faced with a large number of design goal conflicts. Currently, such conflicts need to be overcome manually but we envisage that in the future some automatic support can be provided.

4.2 Lessons Learned

In this sub-section we discuss with the aid of a number of questions all the lessons learned from the application of the methodology.

How easy is the framework to learn?

The described approach results from the integration of two existing security-aware approaches. As such, a number of software engineers are familiar with their concepts and notation. Especially, the UMLsec approach effectively uses UML concepts and notation and therefore it is easy to understand by developers familiar with UML. The Secure Tropos approach on the other hand, is based on the i* [19]/Tropos [3] notation and concepts that although not as popular as UML, it is well known in the requirements engineering area. Therefore, although an initial effort is required to understand the framework, we expect that developers familiar with UML and/or Tropos will be able to grasp the concepts and notations of the methodology easily.

Did you come across any unexpected obstacles during the application of the framework to any of the case studies?

The application of the framework to the case studies did no yield any unexpected obstacles. As discussed above, there were some inconsistencies between the models due to some errors on the guidelines, but we were expecting something like this, since it was the first time the framework was applied to real-life health care case studies. On the other hand, once these inconsistencies were solved, the framework worked as expected and we were able to analyse the environment of the system in terms of its security and transform this analysis to a design, which we could verify.

Was the framework modified to enable its application to the case studies?

The framework was not modified to fit the case studies, but the application of the framework to the case studies resulted in a number of modifications as discussed in the previous section.

How the framework helps the analysis and design of the system with respect to security?

The application of the framework to the case studies revealed that it helps the analysis and design of the system with respect to security in various ways:

(i) Developers are able to consider security both as a social aspect as well as a technical aspect. It is widely known that security is most often compromised not by breaking dedicated security mechanisms but by exploiting vulnerabilities in their usage. Therefore, as argued widely in the literature, it is not enough just to consider security mechanisms and protocols, but an understanding of the human factor and the environment of the software system is also required. By considering both the social aspect and the technical aspect of security, our framework allows developers to obtain a clear understanding of any potential security vulnerabilities that might raise from the interplay of the two security aspects and therefore minimize, leading to the development of secure software systems.

(ii) The framework allows the definition of security requirements at different levels and as a result it provides better integration with the modelling of the system's functionality.

(iii) Security is not considered in isolation but simultaneously with the rest of the system requirements. Such treatment of security helps to minimize the number of conflicts between security and other requirements. Such conflicts are usually the reason for security vulnerabilities, therefore by minimizing these conflicts, the security vulnerabilities of the system are also minimized.

(iv) The framework allows the consideration of the organisational environment for the modelling of security issues, by facilitating the understanding of the security needs in terms of the real security needs of the stakeholders, and then it allows the transformation of the security requirements to a design that is amenable to formal verification with the aid of automatic tools. This introduces a well structured approach to model-based security engineering.

Was the framework appropriate for the health care domain case studies?

In general, the framework was appropriate for the two studies from the health care domain. The health care sector is quite complex and security issues are affected not only by related technologies but also from the human factor. The framework allowed us, in collaboration with the health care professionals involved, to analyse both these security dimensions by (i) analysing the security issues imposed to the system by its environment (various stakeholders); (ii) reasoning about different possible solutions that satisfy the system's security requirements.

What useful conclusions did you derive for model-based security engineering?

It is worth mentioning that the case studies were not set up to assess the usefulness of model-based security engineering and/or compare it with other security and non-security engineering approaches. However, some useful conclusions were drawn. First of all, it became obvious that, as with all the security-aware approaches, some extra effort and knowledge is required due to the security aspect. In particular, basic knowledge is needed about security terminology and theory and extra effort is required by the developers to analyse and model the security concerns. Therefore, we expect that model-based security engineering will be an attractive option to developers looking to develop security-critical systems rather than a general option for any software system development. Secondly, the production of models that integrate security concerns, as opposed to the production of models without security concerns, allows developers to (i) reason in a conclusive way and by taking into account simultaneously the general requirements of the system together with the security requirements and therefore identify any conflicts; (ii) develop a extensive and precise security-aware documentation, something that it is required by common security standards, such as the Common Criteria [4].

4.3 Improvements

In this section we discuss how the framework can be improved to enable its application in large industrial context.

Tool Support

Currently the framework is supported by different types of tools corresponding to the two approaches integrated (i.e. Secure Tropos and UMLsec). The transformation from the secure Tropos models to the UMLsec models takes place manually (the Secure Tropos tool produces details of the models developed in XML, which can be used to feed information on the UMLsec tool). Although, this does not prevent the application of the framework to large industrial projects, it is a concern in terms of efficiency and time. The development of a tool to support the transformation of the models will substantially reduce the time that is required to transform the models and therefore increase the applicability of the framework to that type of projects. Such tool will also support the analysis and resolution of design goal conflicts.

Documentation

Currently, the only documentation about the framework is a set of research papers and internal reports describing some of its aspects, as well as documentation that describes the two original approaches, i.e. secure Tropos and UMLsec. It is important, however, to produce a complete documentation that will explain the original approaches, the advantages of the integration, the transformation steps, the models and the new development process. Such documentation will help to understand the framework and to make it accessible to a larger number of developers. In other words, it will help to improve the usability of the framework.

Model related improvements

There are two main issues related to the improvement of the models. The first is related to the version of the UML in which the UMLsec definition is based. Currently, UMLsec is based on UML1.5. The subsequent release of UML 2.0 raises the question to what extent the UMLsec approach is dependent on a particular version of UML, or whether it can be used flexibly with different UML versions (including UML 2.0). This would be a very interesting question to explore. The second issue is related to the linkage between the models and the implementation (code). In particular, automatic generation of text-sequences from the models to provide assurance that the code correctly implements the models and thus satisfies the security requirements of the system would be desirable.

5 Conclusions

This paper presented an experience report from the application of a model based security engineering framework to two case studies from the health and social care domain (German Hospital Telematics and electronic Single Assessment Process). Apart from the reflections described above, our experience of employing such framework to the health care case studies has indicated two important issues related to each of the individual methodological components of the framework. Secure Tropos concepts are more intuitive and comprehensible than for instance Object Oriented concepts for people without information systems engineering background, such as the majority of health care professionals. As such, this provides an advantage during the requirements elicitation stage for health care case studies. Using UMLsec, the extension of the Unified Modelling Language (UML) for secure systems development and the concept of model-based security requirements, security requirements are handled as an integrated part of the development and derived from enterprise information such as security policies, business goals, law and regulation as well as project specific security demands. These are then updated and refined iteratively and finally refined to security requirements at a technical level, which can be expressed using UMLsec, and analyzed mechanically using the tool-support for UMLsec by referring to a precise semantics of the used fragment of UML. This allows one to validate design against security requirements early in the development cycle.

References

1. M. Alam, M. Hafner, R. Breu: Constraint based role based access control in the SECTET-framework A model-driven approach. Journal of Computer Security 16(2): 223-260 (2008)
2. D. Basin, J. Doser, T. Lodderstedt, Model Driven Security for Process Oriented Systems. In Proceedings of the 8th ACM symposium on Access Control Models and Technologies, Como, Italy, 2003
3. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, A Perini. TROPOS: An Agent Oriented Software Development Methodology. In Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Publishers Volume 8, Issue 3, Pages 203-236, 2004
4. Common Criteria, http://www.commoncriteriaportal.org/
5. P. Devanbu, S. Stubblebine. Software Engineering for Security: a Roadmap. In Proceedings of ICSE 2000 (track on "The future of Software engineering"), 2000.
6. G. Hermann, G. Pernul, Viewing business-process security from different perspectives. International Journal of electronic Commence 3:89-103, 1999
7. N. R. Jennings, An agent-based approach for building complex software systems, Communications of the ACM, Vol. 44, No 4, April 2001
8. J. Jürjens, Secure Systems Development with UML, Springer, 2004
9. J. Jürjens and P. Shabalin. Tools for Secure Systems Development with UML. FASE 2004/05 special issue of the International Journal on Software Tools for Technology Transfer, Springer. 2007.
10. J. McDermott, C. Fox, Using Abuse Case Models for Security Requirements Analysis. In Proceedings of the 15th Annual Computer Security Applications Conference, December 1999.
11. H. Mouratidis, P. Giorgini, G. Manson, Modelling Secure Multiagent Systems, in the Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne-Australia, pp. 859-866, ACM 2003.
12. H. Mouratidis, I. Philp, and G. Manson (2003) "A Novel Agent-Based System to Support the Single Assessment Process of Older People" Journal of Health Informatics 9(3) pp. 149-162, SAGE Publications.
13. H. Mouratidis, P. Giorgini, Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing, 2006.
14. H. Mouratidis, J. Jürjens, J. Fox, Towards a Comprehensive Framework for Secure Systems Development, CAiSE 2006, Lecture Notes in Computer Science 4001,pp. 48-62, Springer-Verlag, 2006
15. W. Muhanna, W., An Object-Oriented Framework for Model Management and DSS Development, *Decision Support Systems*, **9**:2, pp. 217-229, 1993
16. G. Sindre, A. L. Opdahl: Eliciting security requirements with misuse cases. Requir. Eng. 10(1): 34-44 (2005)
17. A. Sunyaev, Telematik im Gesundheitswesen - Sicherheitsaspekte,tech. rep., TU Munich, 2006
18. M. Wooldridge, P.Ciancarini, Agent-Oriented Software Engineering: The State of the Art, In P. Ciancarini and M. Wooldridge, editors, Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957, January 2001
19. E. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. Thesis. Dept. of Computer Science, University of Toronto. 1995