Automated Support for Security Requirements Engineering in Practice

Daniel Mellado
Ministry of Work and Social Affairs, Information Technology Centre of the National Social Security Institute, Madrid, Spain.
Daniel.Mellado@alu.uclm.es

and

Moisés Rodríguez, Eduardo Fernández-Medina, Mario Piattini
University of Castilla-La Mancha, Alarcos Research Group, Information Systems and Technologies Department,
Paseo de la Universidad 4, 13071 Ciudad Real, Spain.
{Moises.Rodriguez; Eduardo.FdezMedina; Mario.Piattini}@uclm.es

Abstract
Security Requirements Engineering is a discipline that is emerging as an important branch of Software Engineering since we are becoming more aware of the fact that security must be dealt with early on the requirements phase. However, without a CARE (Computer-Aided Requirements Engineering) tool, the application of any methodology or requirements engineering process will normally fail because it has to be manually performed. Therefore, in this paper, we will present a prototype of SREPTOOL, which provides automated support to facilitate the application of the security requirements engineering process, SREP. SREPTOOL simplifies the management of security requirements by providing us with a guided, systematic and intuitive way to deal with them from the early phases of software development, simplifying the integration of the Common Criteria (ISO/IEC 15408) into the software development process as according to SREP proposes, as well as the management of the security resources repository. Finally we will illustrate SREPTOOL by describing a simple real case study, as a preliminary validation of it.

Keywords: Security Requirements, CARE, Case study, Common Criteria, Security Engineering.

1. Introduction
Nowadays, it is widely accepted the principle which establishes that security construction at the early phases of the software development process is cost-effective and generates more robust designs [17]. Therefore, software security is getting more and more interesting for software engineers [28]. This has caused that the discipline of Security Requirements Engineering is highly considered as part of Security Engineering applied to the process of development of information systems (IS) that so far, has not been paid the necessary attention [19]. This discipline known as Security Requirements Engineering, has become a very important part in the software development process for the achievement of secure software systems, because it provides techniques, methods and standards for tackling this task in the IS development cycle. It also implies the use of repeatable and systematic procedures to ensure that the set of requirements obtained is complete, consistent and easy to understand and analyzable by the different actors involved in the development of the system [18].

In spite of all these considerations, nowadays there are still many organizations that tend to pay little attention to security requirements. One of the reasons is the lack of CARE (Computer-Aided Requirements Engineering) tools that support the application of the methods, methodologies or processes of security requirements engineering. This implies, as described in [2], that the implementation of this kind of processes normally fails since it has to be manually performed.
In [23] we compare several proposals of tools for IS security requirements, concluding that they did not reach the adequate level of integration into the development of IS, nor provided intuitive, systematic and methodological support for the management of security requirements at the early phases of the IS development, with the aim of developing secure IS that conforms to the most relevant security standards with regard to the management of security requirements (such as mainly ISO/IEC 15408 [12] as well as ISO/IEC 27001 [14], ISO/IEC 17799 [13] or ISO/IEC 21827 [10]). With this objective and starting from the former described concept of Security Requirements Engineering, we proposed SREP (Security Requirements Engineering Process) [24].

In this paper, we will describe the prototype of a security requirements management tool called SREPTOOL that we have developed to provide automated support to the SREP application. SREPTOOL will provide a guided, systematic and intuitive way for the application of the security engineering process SREP, as well as a simple integration with the rest of requirements and the different phases of the IS development lifecycle. It also facilitates the integration of the Common Criteria (CC) [12] into the software development process as well as the fulfilment of the IEEE 830:1998 standard [8]. To do so, it is helped by using the functionalities offered by ‘IBM Rational RequisitePro’ (CARE tool which is extended by SREPTOOL). Additionally, this prototype helps to develop IS which conforms to the aforementioned security standards with regard to the management of security requirements and without being necessary to perfectly know those standards and reducing the participation of security experts to get it, in other words, it improves the SREP efficiency. Furthermore, thanks to the Security Resources Repository included in SREPTOOL, it is easier to reuse security artifacts, thus improving quality successively.

The rest of the paper is organized as follows: In section 2, we will summarize some of the basic characteristics of SREP with the aim of understanding the later explanation of the tool. Then, in section 3, we will offer a comparison of CARE tools and other security requirements tools. Later, in section 4, we will illustrate the tool by describing a simple real case study, as a preliminary validation of SREPTOOL, as well we will put forward the lessons learnt. Next, in section 5, we will present the related work and SREPTOOL main contributions. Lastly, our conclusions and future work will be set out in section 6.

2. Overview of SREP

The Security Requirements Engineering Process (SREP) [24] is an asset-based and risk-driven process for the establishment of security requirements in the development of secure IS and whose focus seeks to build security concepts at the early phases of the development lifecycle. Basically, this process describes how to integrate the Common Criteria (CC) [12] into the software lifecycle model together with the use of a security resources repository to support reuse of security requirements, assets, threats, tests and countermeasures. Moreover, it facilitates the different types of traceability relationships (according to the traceability concepts in [25] that are based on [6, 18]): pre-traceability and post-traceability, backward traceability and forward traceability; inter-restrictions traceability and extra-requirements traceability.

In a generic way, we can describe this process as an add-in of activities (that are decomposed into tasks where artifacts to get in and out are generated, and with the participation of different roles) that are integrated into the current model of any organization providing it with a security requirements approach. In [24], we have described in a more detailed way how SREP is integrated into the lifecycle of the Unified Process that, as we all know, is divided into a sequence of phases and each phase may include many iterations. Thus, the model chosen by SREP is a spiral process model, and its associated artifacts evolve throughout the lifecycle and are dealt with at the same time as the others functional and non-functional requirements and the rest of artifacts of the software development process.

SREP proposes a Security Resources Repository (SRR), which facilitates development with requirements reuse. Moreover, reusing security requirements helps us increase their quality for an improved use in subsequent projects [27]. The security requirements can be obtained from security objectives or threats starting from the assets. Furthermore, SREP domains consist of artefacts belonging to a specific application field, such as finance or Social Security. The concept of package refers to a homogeneous set of requirements that can be applied to different domains and that are put together to satisfy the same security objectives as well as to mitigate the same threats, being a bigger and more effective reuse unit.

Finally, we would like to point out the fact that using the CC, a large number of security requirements on the system itself and on the system development can be defined. Nevertheless, the CC do not provide us with methodological support, nor contain security evaluation criteria related to administrative security measures not directly related to the IS security measures. However, it is known that an important part of the security of an IS can
be often achieved through administrative measures. Therefore and according to ISO/IEC 17799:2005, SREP suggests including the set of legal, statutory, regulatory and contractual requirements that the organization, its commercial partners, contractors, services providers and its socio-cultural environment should satisfy. After converting these requirements into software and system requirements format, these requirements along with the CC security requirements would be the initial subset of security requirements of the SRR for any project.

3. CARE Tools Comparative

First of all, we had to initially decide whether to develop a new tool or to extend an existing one. Thus, taking into account the characteristics of the SREP process and the objectives of its application, we considered appropriate to extend an existing tool and to focus the search on the field of CARE tools, discarding tools of document or content management (CMS, Content Management Systems) as well as other type of CASE (Computer Aided Software Engineering) tools which are centred in other phases of the lifecycle. We believe that a major success factor for practical security requirements engineering will be seamlessly integrated with existing tools instead of requiring special purpose tools.

<table>
<thead>
<tr>
<th>Table 1 Summary of the comparative analysis of CARE tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RequisitePro</strong></td>
</tr>
<tr>
<td>Extensibility of the functionality</td>
</tr>
<tr>
<td>Traceability</td>
</tr>
<tr>
<td>Integration with other tools of the life cycle</td>
</tr>
<tr>
<td>Reuse Support</td>
</tr>
<tr>
<td>Project Repository</td>
</tr>
<tr>
<td>Specification Validation</td>
</tr>
<tr>
<td>Specification Standards</td>
</tr>
<tr>
<td>Previous experience. Use easiness and user interface</td>
</tr>
<tr>
<td>Requirement Importation</td>
</tr>
<tr>
<td>Version control and baselines</td>
</tr>
<tr>
<td>User and Role Based Control Access</td>
</tr>
<tr>
<td>Parameterized Requirements</td>
</tr>
</tbody>
</table>

Therefore, in this section, with the aim of obtaining a general view of CARE tools, we will present a summary of the state of the art of these tools. Firstly, we will identify the existing CARE tools and we will select those to be studied in depth; then, we will make a comparison between them with the help of an analysis framework. For the definition of the analysis framework that would facilitate the appropriate selection of a CARE tool, we took into account the general requirements of a good CARE tool [7], as well as the concrete needs that the CARE tool would have to support the correct application of the SREP process.

Based on the INCOSE survey [9], we carried out a first selection of tools that fulfilled the majority of the key functions of a CARE tool, and that have shown us that, with their degree of penetration into the market, provide effective solutions as it is stated in diverse studies on the subject such as [1, 9]. In table 1, it is summarized the
comparative analysis performed on the selected tools, using as comparison criteria those characteristics considered necessary for the satisfactory application of the SREP process.

These tools provided almost all the needs required for a CARE tool. However, we observed with the performed analysis that none of them covered all the needs to provide automated support for SREP. In fact, some of their limitations were critical for a satisfactory application of SREP. Among other found weaknesses, we can highlight the fact that none of the tools made it possible requirements instantiation through parameterization. In the same way, they did not provide adequate techniques for either security requirements specification (such as security use cases [3]) or threats specification (misuse cases [26]). Also, they did not facilitate either automated methodological support for security requirements management, lacking of support for fundamental activities such as risk assessment or the fact of conforming to the current most relevant security standards with regard to the management of security requirements (such as ISO/IEC 15408, ISO/IEC 27001, ISO/IEC 17799 or ISO/IEC 21827).

Finally, we decided to extend RequisitePro as a support for our prototype mainly due to next factors:

• Extensibility: RequisitePro allows us to access the data stored in it (project, requirements, attributes, etc.), as well as let us control the RequisitePro user interface and Microsoft Word documents as well. Despite being a little more limited than in the other tools, it was clearer and more simple to adapt to SREPTOOL needs.

• Automated integration with the rest of the life cycle activities. RequisitePro as it is integrated into the “Rational Suite Analyst Studio” package facilitated a key aspect for SREP, the integration not only with the other requirements but also with other artifacts of the IS lifecycle (as its integration with “Rational Rose” modelling elements).

• Previous Experience. RequisitePro tool has been widely used as support tool in projects previous to SREPTOOL development (being used for example in [22]) since it is the corporative tool of the Information Technology Center of the National Social Security Institute (organism which the first author belongs to). So that this fact could be very interesting for carrying out real case studies of SREPTOOL to validate the tool.

• Use easiness and multiuser. One of the most highlighted characteristics is its integration with Microsoft Word text processor as well as the possibility of watching all its functions in a unique view. Furthermore, it provides the possibility of multiuser access to the project and a collaborative web interface.

• Traceability. RequisitePro allows the creation of traceability relationships between different types of requirements and it is visualized through a traceability matrix.

• Other relevant factors: RequisitePro allows a certain reuse by using documents patterns. Besides, its repository supports several well-known commercial relational databases and it offers control of versions of requirements.

4. SREPTOOL: Case Study

In this section, we will describe how SREPTOOL can be applied in practice. First of all, the technology used in the tool is put forward. Then, the functionalities of the tool will be described with the help of a simple and real case study to illustrate them. Due to space constraints, this case study is unrealistically simple to enable functionalities of SREPTOOL to be easily illustrated in this paper.

4.1. Developing the Tool

To create the prototype, we have generated a library dll ActiveX that will be linked with RequisitePro. In this way, the objects of RequisitePro will be visible from SREPTOOL and on the other hand, the artifacts generated by the prototype will be visible from RequisitePro. Thus, the prototype functionality would be accessible from the main window of Rational RequisitePro, through the menu Tools→SREPTOOL. For the integration with RequisitePro, SREPTOOL has been developed as an add-in of it. To do so, we have used the RequisitePro extensibility interface, which allows us to access data stored in RequisitePro and controls the RequisitePro user interface and also allows us to control Microsoft Word documents. In addition, it will use two databases. The first one will contain the users information and it will be encoded. The second one will be the repository and its access will be controlled through the use of password.

4.2. Case Study

The case study presented here is a representative case of a security critical IS in which security requirements have to be correctly treated in order to achieve a robust IS. It will be studied a real case of an e-government service available on the web of the Social Security of Spain (http://www.seg-social.es) which consists of an application (called ePension) that basically allows us to provide information about the status of the pension/s of a concrete
citizen. Previously studied in [22], under a different perspective, in this case, it was carried out in the context of a reengineering process, which was performed over the application to adapt it to a new technical environment, so it was critical that the “new” application continued being secure. SREPTOOL helped us to obtain all the security requirements of the application.

In this first iteration, as functional and non-functional requirements, which were registered in the IBM-Rational RequisitePro, we identified the following:

- **Req1**: On request-1 from a User/Citizen, the system shall display the list of current existing dossiers about his/her pension (social welfare provision). This request shall include data that identify the User/Citizen without mistake, like the social security number.
- **Req2**: On request-2 from a User/Citizen, the system shall display the details of the previously selected dossier (type of pension, disability, amount of money, bank account number, etc.). This request shall include data that identify the User/Citizen without mistake, like the social security number and the dossier number.
- **Req3**: ePension must have W3C-WAI level double ‘A’ conformance (Usability).
- **Req4**: Performance according to the organizational policy for a web application.

Fig. 1 Activity 1 of SREPTOOL

In addition we assume that the Organization has already introduced some elements into the Security Resources Repository (SRR), such as legal and regulatory requirements that the organization has to comply with, and their socio-cultural environment. After converting these requirements into software and system requirements format, these requirements along with the CC security requirements will be the initial subset of security requirements of the SRR, which together with their associated security elements (security objectives, assets, threats, …) will be the initial subset of security elements of the repository of SREPTOOL.

4.2.1 Activity 1: Definitions Agreement. As we can see in Fig. 1, SREPTOOL helped us to reach an agreement upon a common set of security definitions like: Information security, threat, confidentiality, etc., by providing us with the definitions of these concepts according to ISO/IEC 17799:2005 and ISO/IEC 27001. In addition, SREPTOOL allows us to define new standards as well as their concepts, which will be registered in the repository. It also allows us to state the evaluation assurance level (EAL) of the Common Criteria, such as EAL-1 (Functionally Tested) in this case study; and to set the stakeholders, that will participate in the project, and their roles, according to the available Human Resources previously introduced into the SRR. Furthermore, we collected the SREP starting artifacts in the Security Vision, such as the Organizational Security Policy, Legislation, Context and Security Environment and Previous Assumptions. Finally, SREPTOOL can automatically generate the “Security View Document” with all the data that we had already introduced into this activity (and which conforms to CC format, as all the automated documentation generated by it).
4.2.2 Activity 2. Assets Identification. After analysing the functional requirements (Req 1 and Req 2) and according to CC assurance requirement ADV_FSP.3.1D, we identified the Information as the most relevant asset type. Other assets would need to be considered in a case study without space constraints, including tangible and intangible assets such as reputation. Then we introduced the domain of the project (which is an input of SREP), and the tool showed us all the assets related to the domain. In case an asset is not in the repository, it could be introduced into this tab. As we can see in Fig. 2, in this case, the domain is “Web Application of Social Security” and the assets are as follows (in this first iteration of SREP):

- Personal information about the pensioner: name, social security number, address.
- Personal information about the pension/s: kind of pension (old-age / disability (type of disability) / widow’s pension.), amount of money, bank account number.

4.2.3 Activity 3: Security objectives Identification. Selecting the assets one by one, the tool showed us their related security objectives available in the selected domain on the repository. In Fig. 3, we selected the next security objectives:

- SO1: To prevent unauthorised disclosure of information. (Confidentiality).
- SO2: To prevent unauthorised alteration of information. (Integrity).
- SO3: To ensure availability of information to the authorised users.
- SO4: To ensure authenticity of users.
- SO5: To ensure accountability.

Additionally, for each one of the security objectives introduced into the project, SREPTOOL lets us establish a valuation. Moreover, we could create new security objectives that there were not at the SRR and we could define dependencies between them, but this case study is only a first iteration, thus we only used the security objectives defined in the SRR. Finally, it could be automatically generated the “Security Objectives Document” (with CC format).
4.2.4 Activity 4: Threats Identification. SREPTOOL allowed us to retrieve the threats associated with the assets of the project automatically. As we can see in Fig. 4, we identified the following threats that could prevent to reach the formerly identified security objectives:

- Threat 1: Unauthorised disclosure of information.
- Threat 2: Unauthorised alteration of information.
- Threat 3: Unauthorised unavailability of information.
- Threat 4: Spoof user
automatically generate the “Security Problem Definition Document” with the help of the CC assurance class “ASE”.

4.2.5 Activity 5: Risk Assessment. Having identified the threats, then we carried out the risks assessment. In order to carry out this task, SREPTOOL uses a technique proposed by the guide of techniques of MAGERIT [21] and which is based on tables to analyse impact and risk of threats (and which conforms to ISO/IEC 13335 [11]). Firstly, and with the help of the stakeholders, we estimated the degradation that produces each selected threat for each one of the security objectives. Then, we introduced the likelihood with which each threat takes place. And, with these data the tool automatically calculated the risks, as we can see in Fig. 5. Finally, SREPTOOL generated the “Risk Assessment Document”.

4.2.6 Activity 6: Security Requirements Elicitation. In order to derive security requirements, each security objective was analysed for possible relevance together with its threats which imply more risk, so that the suitable security requirements or the suitable package of security requirements that mitigate the threats at the necessary levels with regard to the risk assessment were selected.

Once the relevant threats to the project have been selected, we selected those security requirements that the user believed that are necessary. To do so, we had three options, as we can see in Fig. 6:

- Having selected a threat, the prototype will show the requirements related to that threat in the SRR. The user will only have to select those requirements he/she considers relevant.
- Having selected a class and one of its families (according to the definition of both concepts in the Common Criteria), the prototype will show the security requirements associated with such family (the security and assurance requirements of the CC must be previously inserted into the repository during SREPTOOL installation). The user will be able to select and add to its project the desired requirements.
- Selecting one of the requirements packages and within the package, the desired requirements.

In this case study we used the first option, and we selected and added to the project those security requirements that we considered relevant to the threats previously defined. These security requirements were:

- SR1: The security functions of ePension shall use cryptography [assignment: cryptographic algorithm and key sizes] to protect confidentiality of pension information provided by ePension to a User. (CC requirement FCO_CED.1.1).
- SR2: The security functions of ePension shall identify and authenticate a User by using credentials [assignment: challenger-response technique based on exchanging of random encrypted nonces, public key certificate] before a User can bind to the shell of ePension. (CC requirements FIA_UID.2.1 & FIA_UAU.1.1).
SR3: When ePension transmits pension or pensioner’s information to the User, the security functions of ePension shall provide that user with the means [assignment: digital signature] to detect [selection: modification, deletion, insertion, replay, other integrity] anomalies. (CC requirement FCO_IED.1.1).

SR4: The security functions of ePension shall ensure the availability of the information provided by ePension to a User within [assignment: a defined availability metric] given the following conditions [assignment: conditions to ensure availability]. (FCO_AED.1.1 requirement).

SR5: The security functions of ePension shall require evidence that ePension has submitted pension information to a User and he/she has received the information. (CC requirement FCO_NRE.1.1).

SR6: The security functions of ePension shall store an audit record of the following events [selection: the request for pension information, the response of ePension] and each audit record shall record the following information: date and time of the event, [selection: success, failure] of the event, and User identity. (CC requirements FAU_GEN).

Moreover, SREPTOOL allows us to relate security requirements to the functional and non-functional requirements of the project. It also facilitates the definition of new security requirements by means of a security use case specification, with the help of a template, as well as the selection and/or creation of countermeasures and security tests.

4.2.7 Activity 7: Priorization. The purpose of this activity is that of automating the security requirements prioritization according to the risk of the threats mitigated by them and the dependences between other functional and non-functional requirements. For each one of the security requirements established in the project, we selected which level of priority we will assign it (Critical, Standard and Optional). Then by pressing the “Prioritize” button, SRETOOL sorts the security requirements list from more to less priority.

4.2.8 Activity 8: Requirements Inspection. In this activity, SREPTOOL facilitates the task of verifying that the security requirements conformed to IEEE 830:1998 and ISO/IEC 15408, because it made easier for the user the verification and validation of security requirements through checking those threats for which we have not specified security requirements in the project, together with the assurance requirements that have not been added to the project according to the assurance level defined in activity 1.
As we can see in Fig. 7, SREPTOOL advises the quality assurer and the inspection team that we had forgotten to add some assurance requirements like: ADV_FSP.1 Basic functional specification; AGD_OPE.1 Operational user guidance; AGD_PRE.1 Preparative procedures; ALC_CMC.1 Labelling of the TOE; ALC_CM.1 TOE CM coverage.

Finally, the tool can generate the “Validation Report” and the “Security Requirements Rationale Document” (under CC format).

4.2.9 Activity 9: Repository Improvement. SREPTOOL allows us to select those security artifacts modified/generated in the iteration and considered interesting for being introduced into the SRR.

At last, in this activity, the tool generates the Security Target Document conforming to the Common Criteria (ISO/IEC 15408) that integrates all the information related to the rest of artifacts generated by SREPTOOL in the previous activities.

4.3 Lessons Learnt.

Among the most important lessons learnt we may stand out from the case study presented above we can highlight the following ones:

• The application of this case study has allowed us to improve and refine several functionalities of SREPTOOL. Furthermore, we refined SREP with regard to the participation of some roles in some activities.
• Tool support is critical for the practical application of this process to large-scale software systems due to the number of handled artifacts and the several iterations that have to be carried out.
• Integration with other tools of the lifecycle is essential to get an appropriate traceability of the security requirements. And an appropriate implementation of the security requirements engineering into an organization.

5. Related Work

Extensive work has been carried out on security requirements during the last few years as it was presented in [23], and there are several works that deals with security requirements management tools, similar to SREPTOOL. There are proposals particularly similar to ours:

SirenTool [20] is an add-in of Rational RequisitePro supporting the SIREN (Simple REuse of software requirements) method [27], which is a method to elicit and specify the security system and software requirements including a repository of security requirements initially populated by using MAGERIT and which can be structured according to domains and profiles in a similar way to SREPTOOL. Although SirenTool focuses on requirements lists and it only reuses requirements, which are retrieved via MAGERIT asset hierarchy or via the aforementioned
repository structure. A distinguishing property of our suggestion is that we suggest reusing specifications of requirements and threats, as well as security objectives, assets, countermeasures and tests, so that the requirements can be retrieved via assets, security objectives or threats.

ST-Tool [5] is a CASE tool developed for modelling and analysing functional and security requirements, it allows us to design and verify them. ST-Tool has been designed to support the Secure Tropos methodology [4]. It is an agent-oriented software development tool, which manages the concepts of actor, service and social relationship. In contrast to SREPTOOL it does not deal with security resources reuse, nor incorporate into its steps the CC and it does not facilitate the generation of reports.

UMLsec-Tool [16] supports UMLsec [15]. There are presently several projects at the Munich University of Technology aimed at the development of automated tools for processing UMLsec models to make the methodology directly applicable in the IT industry. They provide an extension to the conventional process of developing use-case-oriented process for security-critical systems. They consider security aspects both in the static domain model and in the functional specification. For the elaboration of the functional aspects they introduce a question catalogue and for the domain model an UML-extension, UMLSec.

In brief, the main contributions of SREPTOOL with respect to the former proposals are as follows:

• SREPTOOL is a standard-centred tool. It helps us to develop IS which conforms to the most current important security standards in different activities of the requirements engineering process. It integrates the CC (ISO/IEC 15408) security functional requirements into the elicitation of security requirements and it also introduces the CC security assurance requirements into the requirements inspection. In addition, it conforms to ISO/IEC 13335 (GMTS) to carry out the risk assessment. Moreover, it facilitates the conformance with the sections about security requirements of ISO/IEC 17799:2005 (sections: 0.3, 0.4, 0.6 and 12.1) and ISO/IEC 27001:2005 (sections: 4.2.1, 4.2.3, 4.3, 6.a, 6.b and A.12.1.1).
• It is a reuse-based tool based on a security resources repository, so that threats and requirements and their specifications, security objectives, assets, countermeasures and tests, are reused. Thus, their quality is successively increased.
• SREPTOOL integrates the latest security requirements specification techniques (such as security use cases [3], misuse cases [26], and UMLsec in the next version).
• It facilitates the automated integration of the security requirements with the rest of the security artifacts and lifecycle activities, not only with the other requirements but also with other artifacts of the IS lifecycle.
• It automates the reports generation.
• It is conducted by assets or threats and the risk

6. Conclusions and Future Work

Nowadays, software security is generating a growing interest. Moreover, security requirements engineering has become a very important part in the software development process for the achievement of secure software systems. While traditional requirements management tools are not able to directly support the above-exposed security requirements management. We have shown in this paper that a seamless integration of security requirements engineering concepts and the most relevant security standards with regard to the management of security requirements (such as ISO/IEC 15408, ISO/IEC 27001, or ISO/IEC 17799) in these tools is possible. Thus, tools like SREPTOOL are actually a critical enabler for the industrial uptake of security requirements engineering, fact which was shown in the real case study performed at the National Social Security Institute (Spain) [22].

Finally, there is a set of aspects planned for the future of this prototype that will allow us to increase the level of automation of SREP application and so, a better efficiency of the organizations requirements engineering process. Among them, we can highlight the following: to extend the type of supported requirements specifications in order to support UMLSec [15]; to refine the integration with RequisitePro and to extend the tool for it to be supported in other CARE tools; to automate the creation of security use cases by using misuse cases created in SREP activity 4.

Acknowledgments

This paper is part of the ESFINGE (TIN2006-15175-C05-05) and RETISTRUST (TIN2006-26885-E) projects of the Ministry of Education and Science (Spain), and of the MISTICO (PBC-06-0082) and DIMENSIONS (PBC-05-012-2) projects of the Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla- La Mancha and the FEDER.
References