Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach

Ambrosio Toval[†], Joaquín Nicolás[†], Begoña Moros[†], Fernando García[‡]

[†] Software Engineering Research Group. Departamento de Informática y Sistemas

Universidad de Murcia. Campus de Espinardo. 30071. Murcia (Spain) {atoval, jnr, bmoros}@um.es

[‡] Innosec (*Innovative Secure Comunications, Inc.*). Paseo de la Castellana, 118

28046 Madrid (Spain) fernando.garcia@innosec.es, www.innosec.es

ABSTRACT

Information systems security issues have usually been considered only after the system has been developed completely, and rarely during its design, coding, testing or deployment. However, the advisability of considering security from the very beginning of the system development has recently begun to be appreciated, and in particular in the system requirements specification phase. We present a practical method to elicit and specify the system and software requirements, including a repository containing reusable requirements, a spiral process model, and a set of requirements documents templates. In this paper, this method is focused on the security of information systems and, thus, the reusable requirements repository contains all the requirements taken from MAGERIT, the Spanish public administration risk analysis and management method, which conforms to ISO 15408, *Common Criteria Framework.* Any information system including these security requirements must therefore pass a risk analysis and management study performed with MAGERIT. The requirements specification templates are hierarchically structured and are based on IEEE standards. Finally, we show a case study in a system of our regional administration aimed at managing state subsidies.

Keywords: Requirements Engineering; Requirements Reuse; Security; Common Criteria Framework; Risk Analysis and Management Methods.

1 Introduction

There has recently been an increasing interest in information systems security issues. For instance, PITAC (*President's Information Technology Advisory Committee*) [1] has stated the need for a scalable information infrastructure, that is, for techniques which ensure that the United States information infrastructure -including communications systems, the Internet, large data repositories, and other emerging systems- is reliable and secure and can grow smoothly to accommodate the massive numbers of new users and applications requiring high bandwidth which are anticipated over the next two decades.

Nowadays, information systems are vulnerable to many threats, such as new viruses (e.g. *worm* viruses) that propagate through Internet; the threats brought about by unacceptable employees use of the Internet resources (such as non-business activities, accidental or deliberate disclosure of sensitive information, and hacking) [2]; failure to observe the personal data privacy laws (leading, in our country, to fines of up to $700,000 or to important administrative sanctions [3, 4]); even strikes or loss of key personnel are threats that information systems need to be able to cope with. In addition, the general acceptance of e-commerce and the digital signature to perform administrative and commercial transactions means that the security of information systems needs to be ever more reliable. A security breakdown can result in very serious problems for an organization: for example, a recent survey on 1,000 UK organizations [5] shows that the occurrence of a security failure in a business with no contingency plan leads to its shutdown in 80% of the cases, and 60% of UK businesses have suffered an important security breach in the last two years.

A large number of methods and regulations concerning the security of organizations have appeared in response to such a situation. Of particular importance are ISO 15408, *Common Criteria Framework* (CCF) [6], and the following national risk

analysis and management methods: CRAMM in the UK (*CCTA –Carmarthenshire College Of Technology and Art– Risk Analysis and Management Method*) [7], MARION in France (*Méthode d'Analyse de Risques Informatiques et d'Optimisation par Niveau*) [8], and MAGERIT (*Metodología de Análisis y GEstión de RIesgos del MinisTerio de Administraciones Públicas*) [9], which is the Spanish Public Administration's adaptation of CCF.

In the information systems development field, requirements form the foundation for the rest of the software development process, since building a high-quality requirements specification is essential to ensure that the product satisfies the users' needs [10-12]. However, drawing up a specification of *quality requirements* is a difficult task. According to the IEEE 830-1998 standard [13], a requirement of *quality* is that it be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable. We agree with Mili et al. [14] that "there exist few alternatives but software reuse as the (only) realistic approach to bring about the gains of productivity and quality that the software industry needs". It has been agreed (see, for example, [12, 15]) that the benefits of reuse are greater when the abstraction level is increased and not only code, but also designs and specifications, are reused.

The usual practice during the early steps in information systems development has been to pay attention to such aspects as reliability, availability or integrity, while security issues have usually been considered only once the system has been deployed, or at best, during the system design, coding or testing [9, 16].

In this paper, we present a proposal to consider information systems security beginning with the Requirements Engineering (RE) process. The approach is based on the reuse of security requirements which are compatible with MAGERIT (and, therefore, also with a CCF subset) and which are collected in a reusable requirements repository.

This approach complements SIREN (*SImple REuse of software requiremeNts*), a general purpose RE method based on requirements reuse (see Figure 1) that we are currently defining. As this paper explains, the inclusion of requirements from the repository controls risk levels within the information system to be built, so that the information system will fulfill the demands of risk analysis performed with MAGERIT. How this approach has been applied to a real case study concerning an information system in our regional government is shown in this paper.

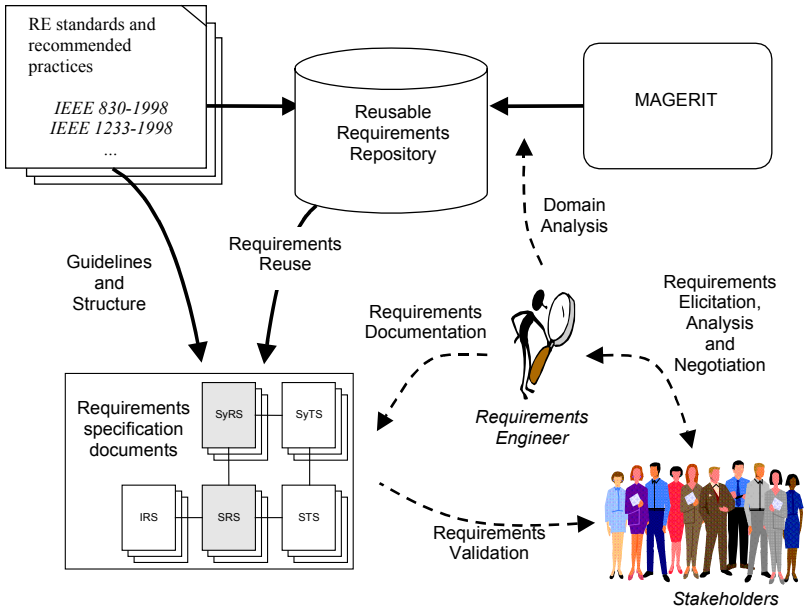

Figure 1. A brief summary of our proposal for requirements reuse, particularized to the security field. The structure of the requirements specification documents is explained in section 4.1.

The structure of this paper is as follows: Section 2 justifies the interest of the problem addressed by our proposal; Section 3 includes a brief description of MAGERIT, the structure of which determines the structure of the security requirements in the repository; Section 4 summarizes our RE process SIREN, and Section 5 describes the case study. Finally, Section 6 shows the conclusions and outlines further research.

2 Reasons behind the study

This research is based on the lessons learned during a risk analysis and management project developed with MAGERIT in our regional government (explained in Section 5).

Our research offers a combination of RE, reuse, and information systems security in order to improve information systems quality and productivity.

2.1 RE and Security

The essential reason behind this proposal is the need to include security explicitly in information systems development, as of the RE process, and thus improve several information systems quality attributes, namely security, safety, reliability, and robustness. Sommerville [12] claims that a high percentage of system malfunctions are the result of errors in specification rather than design. For example, the specification may be incomplete in that it does not describe the behavior of the system required in some critical situations. In a study of errors in embedded systems, Lutz [17] concludes that "(...) difficulties with requirements are the key root cause of the safety-related software errors which have persisted until integration and system testing".

Although the widespread use of Internet has given rise to very active, multidisciplinary research concerning security issues, and researchers in RE are beginning to focus on e-commerce and other Internet issues, we agree with Antón and Earp [16] that there remains a need to apply proven RE methods in this field. Some interesting approaches are the following:

- The integration of SSADM (*Structured Systems Analysis and Design Method*) [18] with CRAMM [7], and Métrica 3 [19] with MAGERIT (Métrica 3 is the standard development method of the Spanish public administration and is similar to SSADM). These approaches relate standard information systems development methods (including requirements specification) to standard risk analysis and management methods, and thus address the entire system-development life cycle. They do not, however, provide a reusable requirements repository to support reuse of security safeguards during the RE process. This criticism also concerns PFIRES (*Policy*

Framework for Interpreting Risk in eCommerce Security) [20], a non-standard framework for risk analysis in e-commerce which addresses the entire system-development life cycle. The plan phase of PFIRES includes a requirements definition step, but it does not provide specific guidance on how to translate the security recommendations into requirements (safeguards), as does, for example, the integration between Métrica 3 and MAGERIT.

- First Chung [21] and later Antón and Earp [16] focused their previously proposed goal-based RE frameworks on the security field (with the latter specifically addressing e-commerce security issues and built on the PFIRES approach). We think that these approaches can be complementary to ours: as we explain below in this paper, we use MAGERIT in the first place to perform a risk analysis, and then we manage risk by reusing MAGERIT-compatible security requirements. Analogously, we could first specify the goals of the system to be developed by means of one of the goal-based frameworks mentioned before, and then operationalize the goals by making use of our security requirements, and thus reuse generic security design knowledge from MAGERIT. Like this, we would be able to specify the relationships between the security needs and the other functional and non-functional requirements.
- Jones et al. [22] are carrying out a project aimed at defining a framework for trust requirements in e-business. This framework contains three major elements: stakeholders in e-business, a list of common e-business process models, and a conceptual model of the elements involved in an e-business transaction. These authors, moreover, seek to develop a list of generic requirements for e-business to be adapted to specific projects by means of the framework. The purpose of this work is very similar to the research that we present in this paper, but the work focuses specifically on e-business, while our work is focused on information systems

security in general. Their research is still at an early stage, and they do not present a list of requirements, but a list of categories of generic requirements.

Security standards do not adequately address the RE process either. We consider that a significant part of the study of Fenton and Neil [23] on lacks of safety related standards is also applicable to security standards (in particular, to ISO 15408 *Common Criteria Framework*). In particular, we agree with Fenton and Neil that: i) the requirements imposed by the standards are imprecise; and ii) the standards are too complex and difficult to use. In fact, Fenton and Neil show a process for improving safety-related standards through the improvement in the structure and writing of the requirements that the standards impose. These authors, moreover, claim that safety standards are conceived to evaluate systems already created, and they do not consider the problem of providing security for new systems.

2.2 RE process improvement

Our proposal also addresses the improvement of information systems quality through the improvement of the quality of the RE process. In this section we deal with the improvement of the quality of the RE process by means of the main processes life cycle standards and quality standards. To this extent, neither life cycle processes standards (such as ISO 12207 [24]) nor quality standards (such as ISO 9000 [25]) provide precise guidelines for identifying problems and planning improvements in the life cycle processes of systems and software (in particular, ISO 9000 does not include any section specifically devoted to RE [26]).

In contrast, capability maturity models, such as CMM [27] and ISO 15504/SPICE (*Software Process Improvement and Capability dEtermination*) [28], do provide guidelines to improve the quality of the life cycle processes. However, once again, they do not cover the RE process [26]. This circumstance led Sommerville and

Sawyer to propose a *Requirements Engineering Process Maturity* (REPM) model [26], which defined a set of good practices in RE, and allowed organizations to be assessed at three levels, namely 1) *Initial*; 2) *Repeatable*; and 3) *Defined*. Level 1 organizations do not have a defined RE process, while level 2 organizations have defined standards for requirements documents and requirements descriptions and have introduced policies and procedures for requirements management. These two levels correspond to CMM levels 1 and 2. Finally, REPM level 3 is assigned to organizations that have a defined RE process model based on good practices and techniques, with an active process improvement program in place and which can objectively estimate new tools and techniques. This third level corresponds to CMM levels 3, 4, and 5.

REPM is based on the incremental adoption of an *RE good practice guide* [26]. This guide, on which our proposal SIREN is based, contains a spiral process model (see section 4.3) and ten good practice guidelines (see Table 1).

Define a standard document structure
Make the document easy to change
Uniquely identify each requirement
Define policies for requirements management
Define standard templates for requirements description
Use language simply, consistently and concisely
Organize formal requirements inspections
Define validation checklists
Use checklists for requirements analysis
Plan for conflicts and conflict resolution

Table 1 Sommerville and Sawyer's top ten guidelines.

3 MAGERIT

MAGERIT [9] is the information systems risk analysis and management method of the Spanish public administration, which is compatible with ISO 15408, *Common Criteria Framework*. In this paper we show how we have translated the security measures stated in MAGERIT into reusable security requirements. MAGERIT is, therefore, the source of our reusable security requirements repository. MAGERIT, moreover, specifies how risk

analysis has to be performed before selecting the required security requirements (see Section 4.3). This section summarizes the basic elements of MAGERIT beginning with a brief description of the evolution of the risk analysis and management methods.

Baskerville [29] studies the evolution of information systems security design methods. The first generation of methods for analyzing risk in information systems appeared in the United States in the early 1970s, based on checklists. From the mid 1980s, there appeared a second generation of more formalized methods headed by the British CRAMM. In the late 1990s, there began to appear a third generation of methods more linked to information systems development methods and closer to security legislation and regulations, that is, 3rd generation security methods take into account the personal data protection laws as well as some other regulations in the security field. The new version of CRAMM (integrated with SSADM) and the first version of MAGERIT (integrated with Métrica) pertain to this generation. Organizational problems are considered in the third generation and so security is reconciled with the functionality of the system, since the security aspects are taken into account from the first stages of the development process [29].

MAGERIT is, in fact, only applied to one of the steps in information systems security management: that of risk analysis and management. Starting from the objectives, strategy and policy of security in the existing information systems in the organization, MAGERIT is applied in order i) to study the risks that affect each information system and its environment, and ii) to propose the countermeasures (safeguards) that should be adopted to register, prevent, avoid, reduce and control the risks evaluated. These countermeasures are collected in an *information systems security plan* which is then put into practice, and followed up during the maintenance stage. The

application of MAGERIT is iterative, so that new risk analyses are periodically performed, and hence new countermeasures are proposed during maintenance.

The risk analysis and management model of MAGERIT includes:

- the *submodel of elements*, providing the basic entities related to the information system risk analysis: *assets*, *threats, vulnerabilities, effects, risks*, and *countermeasures*;
- the *submodel of processes*, describing the stages in the security project that is to be developed: *planning, risk analysis, risk management*, and *recommendation of countermeasures.*

Risk analysis with MAGERIT involves the following major steps: i) identification of the *assets* of the organization, which are the resources of the information system, and which may either directly belong to the information system or just to the information system environment; ii) study of the *vulnerabilities* of these assets and the *threats* to them; iii) estimation of the *risk* related to these assets, based on the threats and vulnerabilities associated (threats are qualified by the likelihood of their occurring); iv) proposal of the *countermeasures* managing the risk. Therefore, countermeasures manage threats and are transitively linked to assets.

As indicated at the beginning of this section, we have translated the MAGERIT countermeasures into security requirements. Security requirements are thus linked to assets. Once risk analysis has been performed and we know how the assets are menaced, it is useful for the structure of the security requirements in the repository (Section 4.2) to conform to the structure of the assets of MAGERIT in order to find the suitable security requirements (countermeasures). It is, therefore, of interest to briefly outline the organization of the assets in MAGERIT (Figure 2).

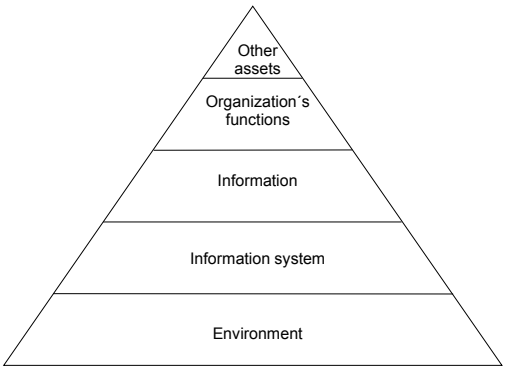

Figure 2. MAGERIT assets hierarchy.

As Figure 2 shows, MAGERIT organizes assets in five layers:

- The *information system environment layer* includes the facilities containing the information system, such as furniture and supplies.
- The *information system layer* includes the information related to the software, hardware, and personnel, which make up the information system.
- The *information layer* includes the information managed by the information system: applications data and metadata (such as data structures and data dictionaries).
- The *organization's functions layer* justifies the usefulness of the information system, by providing it with a purpose; it describes the assets and services produced by the information system, and the users of the information system services.
- Finally, the *other assets layer* describes assets that are usually intangible, and do not fit into the lower layers, as, for instance, the organization credibility or an individual's privacy.

Each layer can be refined into *blocks* of homogeneous assets. Besides, each block can be further divided into *sub-blocks.* For instance, Figure 3 shows the structure of the information system layer. The initials (IS, ISHW, ISSW, etc.) in this figure are

used for maintaining the structure of the MAGERIT asset hierarchy in the security requirements repository, as explained in Section 4.2.

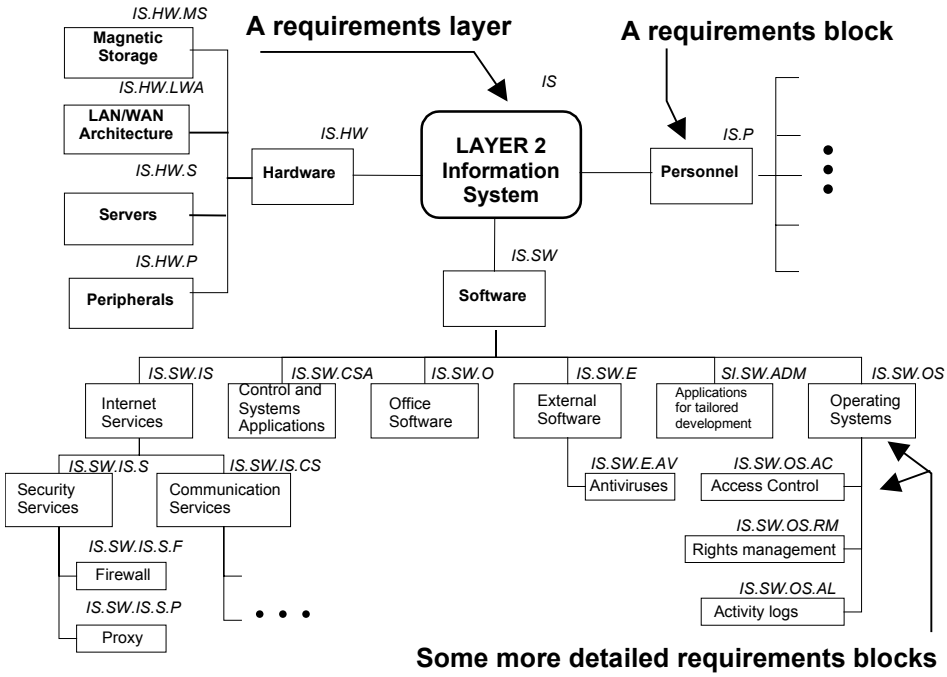

Figure 3. An extract of the MAGERIT information system layer.

4 The SIREN approach to requirements reuse

Our proposal, called SIREN (*SImple REuse of software requiremeNts*), is a method for RE based on requirements reuse. The purpose of development with requirements reuse is to identify descriptions of systems that could be used (either totally or partially) with a minimal number of modifications, thus reducing the total effort of development [15]. In our view requirements reuse can produce more benefit than only design or code reuse because traceability relationships (see Section 4.2) can be established between the high-level requirements of the system and the architecture and implementation which are built from them. Thus, if a system requirement is reused, the subsequent development process can be speeded up. Although a lot of more complex techniques for reusing requirements exist (see the Cybulsky's classification in [15]), our SIREN proposal for requirements reuse emphasizes simplicity and it is therefore more practical.

SIREN also conforms to the most well-known Software Engineering standards for requirements specification (Section 4.1). Requirements have a textual format, but can include any kind of objects as complementary information - for instance, tables or schemas of any type.

SIREN encompasses a process model, some guidelines, techniques and tools. The guidelines that SIREN provides consist of a hierarchy of requirements specification documents together with the templates for each document (Section 4.1). These serve to structure a reusable requirements repository (Section 4.2). Finally, we will present the SIREN process model (Section 4.3) and a discussion on the tools used to support the process (Section 4.4).

4.1 Requirements documents hierarchy

The SIREN requirements documents hierarchy that we propose is depicted in Figure 4. In accordance with Gabb [30] we consider that each document correspond to a different specification level and, therefore, it has different objectives and users.

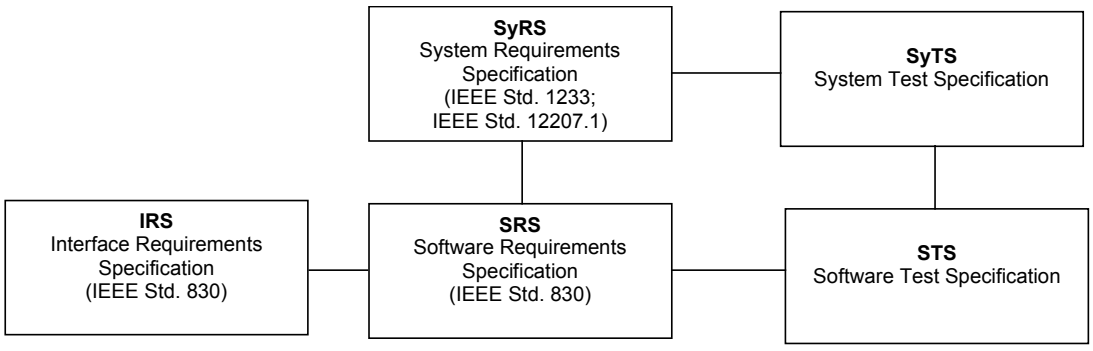

Figure 4. SIREN requirements documents hierarchy.

SyRS. System Requirements Specification

The SyRS describes the function and performance of the whole computer-based system, together with the constraints that will govern its development and the information (data and control) that is the input and the output of the system. It serves as the foundation for

hardware engineering, software engineering, database engineering and human engineering [31].

According to IEEE 12207.0 [24], the SyRS should include: i) functions and capabilities of the system, ii) business, organizational, and user requirements, iii) safety, security, and privacy protection requirements, iv) human-factors engineering requirements, v) operations and maintenance requirements and vi) design constraints. From these features, IEEE 12207.1 [32] details the specific contents of the SyRS template referencing the IEEE Std 1233 [33], *Guide for Developing System Requirements Specifications*, where some guidance on specifying the system requirements and an SyRS outline can be found. This outline is the basis of the SIREN SyRS template.

SRS. Software Specification Requirements

In the SRS, the functions and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints [31].

The SRS includes requirements on the software functionality, external interfaces, performance, design constraints, and software attributes (portability, maintenance, security, availability, and reliability). Most software requirements are directly derived from the system requirements; hence the hierarchical vision presented in Figure 4. The SIREN SRS template is based on the IEEE Std 830-1998 [13] and on the VOLERE template proposed by Robertson and Robertson [11].

SyTS and STS. System and Software Testing Specifications

Any requirement in the SyRS or SRS must be quantifiable in order to be subsequently validated (see, for instance, Robertson and Robertson [11] or Sommerville [12]). Each

requirement in the SyRS and SRS needs to have been linked to a testing criteria (called *fit criteria* in [11]) in the SyTS and STS documents, respectively. These testing criteria specify how to check that the system or software implemented fulfills the requirements defined.

We believe that SyTS can be very useful in the security field because the security plan of the organization can be specified directly from the SyTS. This plan will include a list of security questions (related to personnel, organization, and functions) which may be then easily checked.

IRS. Interface Requirements Specification

The requirements related to the interfaces between the software elements and between software and users can be included in the SRS. Nevertheless, in order not to produce overlong documents, it is sometimes useful to collect all this information in a separate document, called IRS. Therefore, traceability relationships (Section 4.2) need to be established between the SyRS, the SRS modules and the interfaces described. The IRS template has the same structure as the SRS section for specifying interfaces.

Documents hierarchy definition and evolution

These five types of documents do not have to be delivered in each project. The project manager (together with the analysts) decides which hierarchy to use, depending on the project dimension and the complexity of the system to be developed: i) The System Requirements Specification (SyRS) and the System Test Specification (SyTS) are unique, but they are not necessary when the problem is simple and when it is clear from the beginning that all the requirements will be supported by software. ii) The Software Requirements Specification (SRS) is compulsory, but can be divided into sub-documents (i.e. one SRS for each subsystem or module). iii) By contrast, the Interface Requirements Specification (IRS) is optional and is used when the interface

requirements are numerous in order to avoid producing a very long SRS. If this is not the case, they will be included in the SRS. iv) Finally, there should be a Software Test Specification (STS) for each SRS in the hierarchy.

The definition of the hierarchy of documents is performed iteratively during the RE process. This hierarchy usually begins with one SyRS and one SyTS (see section 4.1), along with one SRS (and the corresponding STS). Subsequently, the hierarchy of documents will vary gradually during the RE process: for instance, the SRS could be split into several sub-documents, one for each subsystem identified.

4.2 Contents of the Requirements Repository

The repository contains requirements from specific *domains* and *profiles*. SIREN domains consist of requirements belonging to a specific application field, such as accounting or finance. A SIREN profile, on the other hand, consists of a homogeneous set of requirements that can be applied to a variety of domains, such as information systems security (the one presented in this paper), and the personal data privacy law [34]. The structure of the domains and profiles of the repository (see Figure 5) is determined by the previous requirements specification documents hierarchy.

There are two types of requirements in the repository:

- *parameterized:* this kind of requirement contains some parts that have to be adapted to the application being developed at the time. For example:

  SRS.3.5.3.1.S.30[1] *The security manager shall check the user's identifiers every [time in months] to detect which ones have not been used in the last [time in months].*

  If this requirement is chosen, the parameterized part will be instantiated, that is, the information in brackets will be replaced by a specific value according to the current project. The requirement R4 in Figure 6 represents another example.

[1] Although some requirements management tools, such as *RequisitePro*, do not allow the inclusion of points in the labels, they are used in the example to improve legibility.

- *non-parameterized*: requirements that could be applied directly to any project concerning the profiles and/or domains in the repository. The requirements in Figure 6 (except for R4) and in Figure 7 are examples of non-parameterized requirements.

According to the recommendations of the IEEE 1233 standard [33], each requirement has the following attributes: identification (unique), priority, criticality, viability, risk, source, and others which depend on the project. Although the attributes are not shown in Figure 6 and Figure 7, they are included in the repository.

In our approach, the requirements are labeled in such a way that both their profile or domain, and their location in the document are shown at the same time. For instance, in Figure 5, label SRS.3.5.3.1.S.23 refers to security requirement 23 (S23) in the Software Requirements Specification document (SRS), and it is located in section 3.5.3.1, which deals with the system attributes (section 3.5) related to security (3.5.3), and in particular to confidentiality (3.5.3.1).

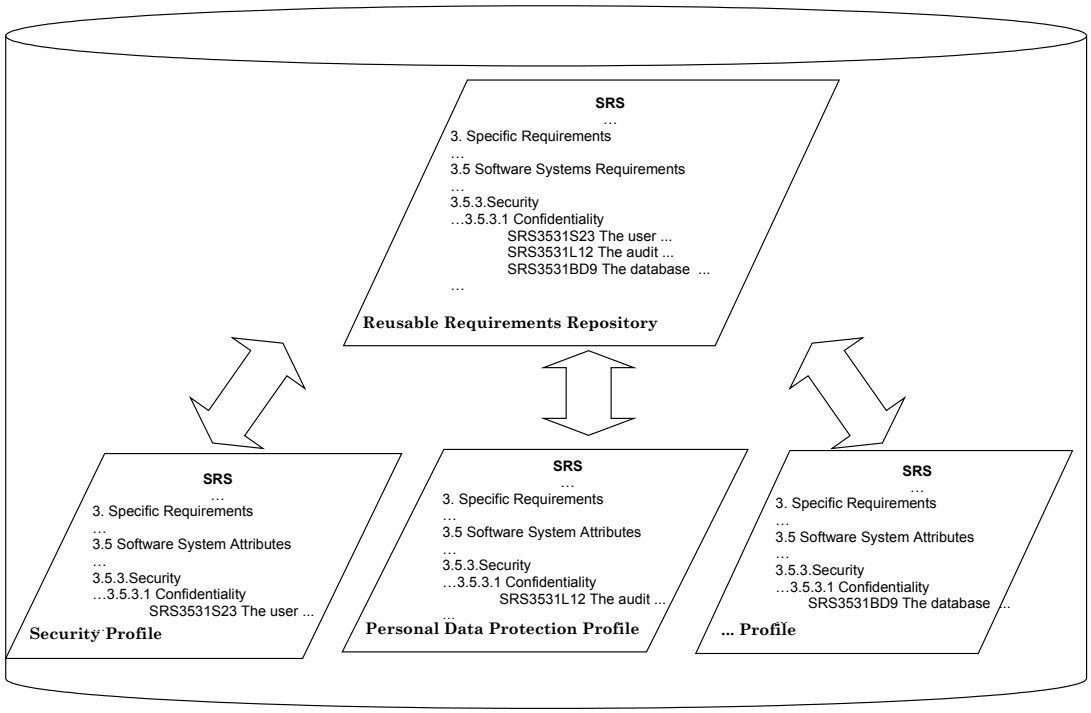

Figure 5 Reusable requirements repository composed of profiles and domains. Each profile or domain is a view of the global repository. In the figure, only the SRS document is shown.

Two additional attributes have been defined in the security profile:

- *MAGERIT-location*, whose values are the initials that specify the position of the security requirement within the MAGERIT asset hierarchy (see Figure 3). For example, the initials IS.SW.IS.S.F (Firewall sub-block in the information system layer, which can be found in Figure 3) are the value of this attribute for the requirements in Figure 7. In this way, the security profile in the requirements repository may be sorted following two criteria: the structure of the requirements specification documents and the structure of the MAGERIT asset hierarchy.
- *Obligement level*, providing the analyst with information about the requirements that have to be included. The possible values for this attribute in the security profile are: compulsory, recommendable and optional.

The *traceability* relationships between the requirements of the repository also have to be collected (see Ramesh and Jarke [35] for a general discussion on traceability). In the current SIREN model, a trace can be established between requirements belonging to the same or different documents (see Figure 6). These traces can imply inclusive or exclusive dependencies:

- An inclusive trace between two requirements A and B means that to satisfy A, B also needs to be satisfied and, therefore, the reuse of A implies the reuse of B. Figure 6 shows an example of an inclusive trace between requirements in the SyRS and SRS. We can observe that for R2 to be achieved, R1 and R3 (and others which are not shown) have to be accomplished, and in order to fulfil R3, R4 also has to be fulfilled. Hence, R2 also depends, transitively, on R4.

SyRS *(System Requirements Specification)*
3. System Capabilities. Conditions and constraints.
3.1. Physical.
3.1.1. Construction.
R1 (SYRS311S68) Documents and diskettes shall be kept under lock and key when they are not being used and out of office hours.
3.6. Policy and Regulation.
R2 (SYRS36S26) The authorized user of the information system shall know their duties related to the access controls and the information under their responsibility. Thus, three conditions are stated:

a) The authorized users have to use their password properly.
b) The authorized users can not for a moment disregard the information that they manage.
c) The authorized users have to follow the security measures enforced to avoid unauthorized accesses to the information managed by them.

Trace-to: R1, R3 (...)

SRS *(Software Requirements Specification)*
3. Specific Requirements.
3.5. Software Attributes.
3.5.3. Security.
3.5.3.1. Confidentiality.
R3 (SRS3531S1) The operating systems used shall provide password mechanisms to control and/or limit the access of the users.

Trace-to: R4

R4 (SRS3531S14) The operating system used shall provide programs to verify the quality of the passwords in the Access Control System. A password is said to be of quality if it fulfils at least these three features:

a) The minimum number of characters is [n, n >= 6].
b) It has at least [n, n>=1] numeric characters and [m, m>=1] alphanumeric characters.
c) It shall be changed every [time in days] by general users and every [time in days] by users with privileges.

Figure 6. Examples of system (SyRS) and software (SRS) requirements. Labels R1-R4 are only used as abbreviations.

- An exclusive trace between two requirements means that they are mutually alternative. An example is shown in Figure 7. Since it is not possible to configure the same *firewall* in two different ways at the same time, only one of the requirements will be selected, according to the security needs of the specific project.

SRS.3.4.3.S.5*. The firewall configuration will be screened host.*
Exclusions: **SRS.3.4.3.S.6**
SRS.3.4.3.S.6*. The firewall configuration will be screened subnet.*
Exclusions: **SRS.3.4.3.S.5.**

Figure 7. Example of exclusive trace between requirements in the SRS. Section 3.4 in the SRS deals with the design constraints and includes the subsection 3.4.3. *Related Applications*.

Traceability can be followed forwards or backwards from a given requirement. Later in the development of the system, forward traces can be used for linking requirements to

the rest of development artifacts (analysis, design and implementation models), thus aiding the evolution and maintainability of the system.

4.3 Process model

According to Kotonya and Sommerville [10], RE basically comprises the phases of i) requirements elicitation, ii) requirements analysis and negotiation, iii) requirements documentation and iv) requirements validation. We agree with these two authors that a *spiral life cycle* [10] (as shown in Figure 8) is the most suitable way to cope with the RE process, since it is not possible to obtain a complete set of the requirements in a single iteration. This model has been adapted in SIREN to take into account requirements reuse explicitly in the RE process.

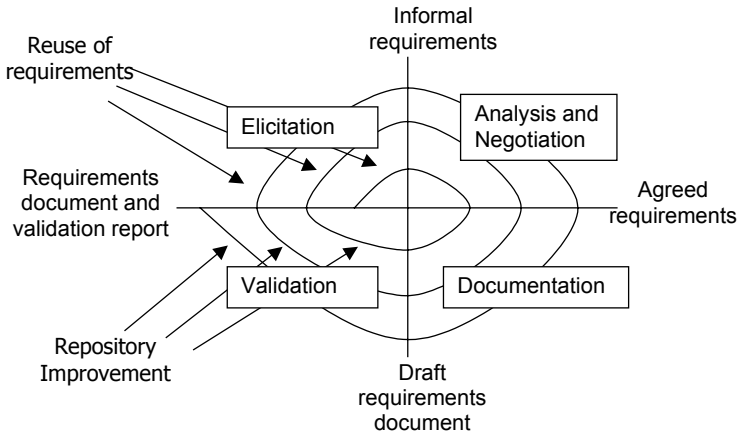

Figure 8. Spiral model for RE adapted to SIREN.

In order to use the security profile properly, the Elicitation phase must start by studying the threats to the assets of the system. Next, the risk concerning those assets will be estimated depending on those threats and the vulnerability of the assets. Finally, a set of requirements from the repository will be selected to establish the countermeasures required to manage the risk analyzed.

The SIREN process model is shown in Figure 9 in more detail. The requirements documents templates (presented in Section 4.1) guide the requirements engineers in the

Elicitation phase since they encompass all the aspects that must be taken into account to write the requirements specification. We divide this phase in Figure 8 into two activities:

a) *Requirements selection*. The approach for reuse consists of providing the analyst with the specification documents templates filled in with requirements. In the domains and profiles related to the current project, the analysts together with the rest of the stakeholders reuse those requirements from the repository that are suitable for the specific application that is being developed. It should be noticed that when a requirement is reused, all the requirements related to it by means of an inclusive dependence will also be selected. In addition, the CARE (*Computer-Aided Requirements Engineering*) tool supporting the process should ensure that two exclusive requirements will not be inserted in the specification.

b) *Specific requirements elicitation*. In the meetings with the rest of the stakeholders, the analysts gather the *informal requirements* of the problem domain as requirements drafts, which contain the specific knowledge about the system that is going to be developed. These requirements are not in the repository initially – they are *specific* for the problem at hand.

The templates with reused requirements and the informal requirements documents are the input for the *Analysis and Negotiation* activity, the aim of which is to discover problems with the system requirements and reach agreements on changes which satisfy all the stakeholders. This negotiation has to remove redundancy and inconsistency. At the end of this activity, the *Agreed Requirements* are identified and then specified in detail in the *Documentation* activity. This latter activity leads to some *Draft requirements documents,* which will follow the selected documents hierarchy. The draft requirements are validated by the stakeholders and thus the *Validated Requirements Documents*, containing the needs and constraints of the information system, are

produced. From these documents, a new spiral in the RE process starts. Eventually, the *Validated Requirements Documents* are considered complete enough to go onto the development process.

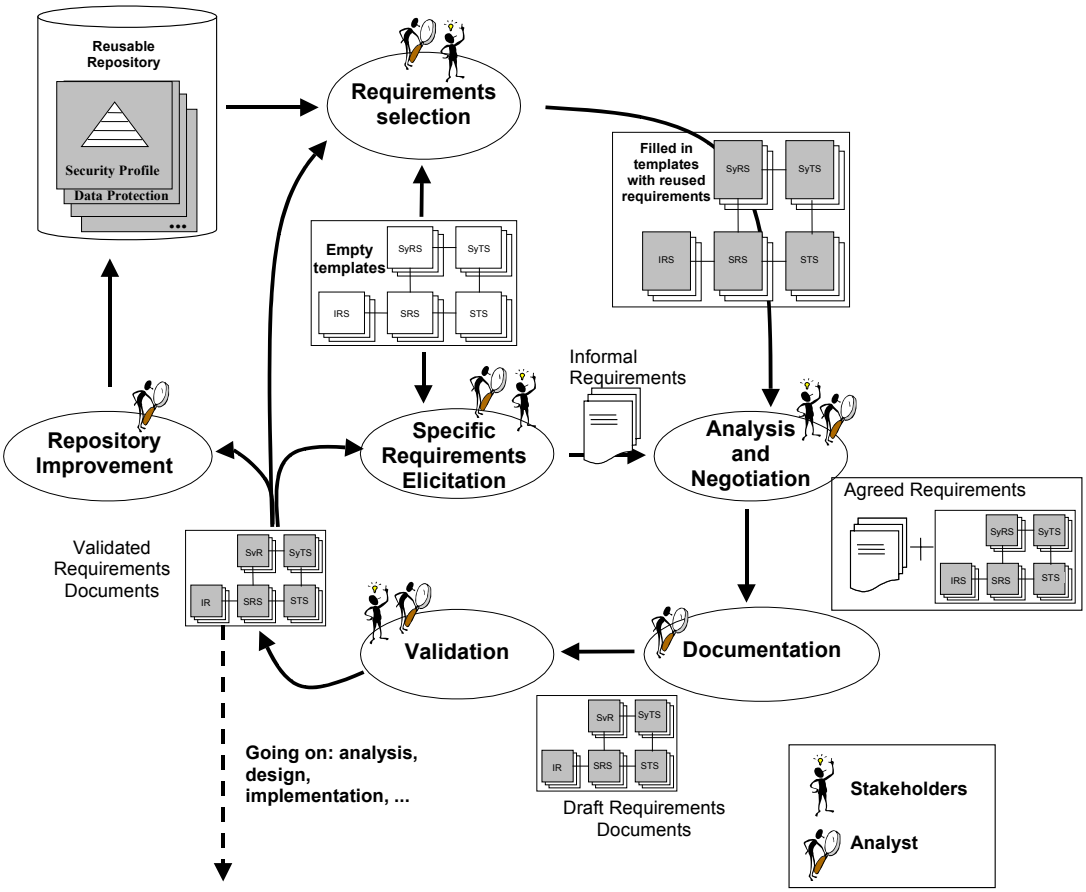

Figure 9. SIREN approach for requirements reuse.

It is important to point out that we are not proposing a waterfall model. These specification documents will evolve during the rest of the life cycle; for instance, during the design, the specification could be enriched with requirements related to the technological environment or the user manuals (such as organization and appearance). We agree with Nuseibeh [36] that the RE and architecture design processes are concurrent and influence each other.

A potentially inherent problem to any requirements specification is premature design. In particular, security requirements tend to propose design solutions (e.g. *password-based validation* in Figure 6) and these may not be appropriate in all

circumstances. In our approach, security requirements specify the countermeasures prescribed by MAGERIT after the risk analysis. Therefore, it is the MAGERIT risk analysis and management which determines the security mechanism to be used in each circumstance. In this way, the security of the information system relies on the MAGERIT risk management and on the proper design and implementation of the security requirements.

The repository shall not be considered as a final static product, but as an evolutionary product that has to be enriched with those new requirements found throughout the development of the current system and which are considered as likely to be used in forthcoming applications. Furthermore, the requirements already in the repository could be modified in order to improve their quality. This is the aim of the *Repository Improvement* activity in Figure 9.

4.4 Tools

To support the process presented in the previous section, we need tools both for managing the requirements and for supporting reuse.

There are many commercially available tools for managing requirements, such as CORE by Vitech Corporation [37], DOORS by Quality Systems & Software Inc (QSS) [38] and Requisite Pro by Rational [39], but none of them provided us with all the required functionality. All of these tools share the same problem: neither parameterized requirements nor exclusive relationships (section 4.2) are supported. In addition, they lack features for supporting reuse.

It is in such a context that we are considering the implementation of our own tool. We want to evaluate Sommerville's recommendation [26] that the easiest way to provide reuse is by means of a documentary database (such as INVESDOC by IECISA [40]), so that the requirements could be searched by key words, and parameterized and exclusive

requirements could be easily implemented. Until now, we have used Requisite Pro because it is widespread, good value for money, it is integrated with the rest of the Rational development tools, and it is easy to use.

5 Case study

As we said at the beginning of Section 2, the approach that we present in this paper stems from the lessons learned during the development of the CARMMA project [41]. In this project our regional government contracted our research group to develop a risk analysis using MAGERIT in the Regional Information Systems and Telecommunications Office (RISTO). The staff at RISTO were very interested in the improvement of the security of their systems since their information must be constantly available to the other regional government offices, and a huge amount of resources (equipment, people and money) are involved in RISTO. A few figures will enable the reader to realize how critical a security breach could be: our regional government has roughly 22,000 employees and a budget for the year 2000 of nearly $1,275 million, while RISTO employs 40 people and has a budget of $2.89 million.

The risk analysis took one year and involved five analysts and some fifty stakeholders. The stakeholders included the following roles: managers and administrative staff (as final users) and technical staff. From the results of the study we realized that:

- the cost of implementing the countermeasures to reduce the risk of most of the critical assets identified would have been lower if these assets had been developed taking into account security issues from the beginning;
- although the countermeasures in MAGERIT are linked to threats, instead of to assets, we think that the implementation of each countermeasure depends

on the specific asset threatened and, therefore, the countermeasures should be linked to assets.

These conclusions lead us to the definition of a SIREN security profile (Section 4.2) structured in line with the MAGERIT asset hierarchy. This repository helps to make security issues explicit from the early steps in the system development process.

We considered that it would be very interesting to validate our proposal in a new case study. We preferred to carry out the new case study in our regional government in order to make use of the knowledge gained in the previous project and so we would be working as security consultants for the office selected. Moreover, we wanted to participate in the requirements specification of a new information system so that we could define the security requirements from the beginning of the development process. This new information system would be secure after the proper implementation of its requirements.

A case study in the Regional Ministry of Labor and Welfare, satisfying the requirements stated above, was then selected to experiment our approach. The project dealt specifically with the management of state subsidies awarded to put into effect activities including the promoting of the business realm and career advice, rural development plans and occupational safety and health measures.

The work group was formed by three people, one from our research group (working as analyst) and the other two from the regional government (working as stakeholders). It is worth pointing out the following two activities carried out by the work group:

1) The performance of a *risk analysis* to detect security needs (threats, critical assets and countermeasures). An asset is critical either because it is very valuable although it is barely threatened or because although it is of little value it is heavily menaced.

The value of an asset is calculated according to the loss that it would cause if it failed, that is, the value of an asset is an accumulated value as shown in Figure 10. For example, the estimated value of the power hub in Figure 10 is $35,897 but its accumulated value is $24,174,348, corresponding to a function of the weighted sum of the values of the assets that rely on the power hub (not all the assets involved are shown in Figure 10 for reasons of simplicity). These values can be used to prioritize the asset (and therefore the countermeasures related to it).

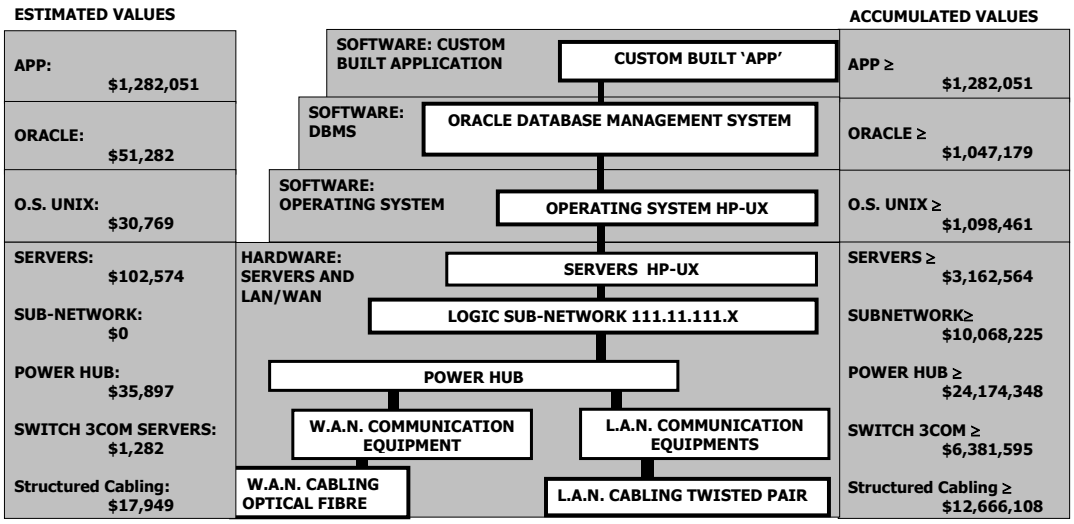

Figure 10. Accumulated values in an extract of the asset hierarchy in our case study. To preserve privacy, the name of the assets and IP addresses have been changed.

In our case study, given that the information has to be constantly available, we identified two main threats concerning the MAGERIT LAN/WAN Architecture block of assets (which can be found in Figure 3): T1) *physic or logic malfunctions* and T2) *main supplies and services interruption*. For instance, we found the power hub in Figure 10 among the assets threatened by T1 and T2. This asset is critical because most communication systems depend on it and its accumulated value is high. The countermeasure defined to reduce the risk was C1) "*Precaution and care of the communication equipment*", which dealt with recommendations for avoiding malfunctions in communication equipment, such as duplicating strategic elements and cleaning security areas.

2) The selection of the *security requirements* from the repository related to the countermeasures defined for each critical asset. The security requirements were selected according to their priority, their obligement level (*compulsory*, *recommendable* or *optional*) and, of course, the budget assigned to the development of the information system. Since the requirements selection was made together with the stakeholders, we negotiated each requirement at the same time as it was selected. In the case of the previously mentioned critical asset, the power hub, one of the security requirements corresponding to the countermeasure C1 which was selected was:

   SyRS.3.5.2.S42. The maintainability contract of the electronic equipment shall include a clause enforcing the supplier to make a commitment to solve any failure in less than [time in minutes].

   If the critical asset had been cheaper than the hub, the security requirement would have been simply to keep a duplicate of the asset to change it whenever it fails.

The number of requirements from the security profile reused is shown in Table 2. It can be noted that the percentage of reuse was high, above all in the requirements of the information layer because it includes the requirements concerning the personal protection data, which are legally enforced in Spain. Most of the environment requirements were also selected, since they were already satisfied in the existing environment (buildings, rooms, furniture, etc.) and therefore their inclusion did not constitute an additional cost. The requirements of the organization's functions layer were those least selected because their scope concerns security policies which exceed our case study.

Requirements layer	Chosen	Totals	% Chosen
Environment	111	136	81.6
Information System	91	121	75.2
Information	48	56	85.7
Organization´s functions	9	29	31
Intangible asset	7	10	70
TOTAL	266	352	75.6

Table 2 Results of the application of SIREN for security requirements

The use of the SIREN security profile was useful:

- in developing a secure information system from the very beginning, taking into account security issues in the RE process that, otherwise, might be overlooked;
- in improving productivity, since there was a decrease in the time dedicated to the RE process.

The experience has been positive for both parties, since the goal of the research group was the transference of knowledge from the University to industry in order to validate academic research in practice, while our regional government counterpart found the approach useful. They suggested to us the inclusion of a new attribute called *responsible* that would indicate the person or team in charge of the realization of the requirement. In that way the workload of each member of a team could be controlled.

This experience has shown that this approach to the RE process is a good, realistic starting point for any organization that has not applied any RE method yet. We agree with Jaaksi [42] in that if the development method is too complicated, the engineers will not commit themselves to working with it, and a repeatable process cannot be achieved. A Level 1 organization (according to the REPM, see Section 2.2) would reach easily Level 2 by applying our SIREN approach.

6 Conclusions and further work

This paper presents a practical approach for managing the security of the information systems from the RE process. We believe it helps in the development of more complete and robust information systems. This approach is a particularization of a general purpose process for requirements reuse called SIREN, which shortens the development process since the analyst starts from a reusable set of requirements. The particularization of SIREN to the security profile has been based on the risk analysis and management method MAGERIT (conforming to IEEE 15408 *Common Criteria Framework*). SIREN provides a spiral model process based on Kotonya and Sommerville's proposal [10] which is enriched by explicitly taking reuse into account. The immediate benefit for an information system development is that the inclusion of these security requirements ensures that the proper implementation of the system will fulfill the demands of risk analysis performed with MAGERIT.

Together with the defined process, SIREN provides a requirements specification documents hierarchy (system, software and tests) based on the IEEE specification standards. The IEEE 830-1998 standard for SRS has been extended with items from the VOLERE template [11], but maintaining the conformance with the standard. The inclusion of a system tests specification document (SyTS), dealing, in particular, with security, was also found to be of interest, because the SyTS can serve as a checklist which encompasses all the information system features that must be checked for it to be considered secure. The traceability relationships between the requirements of the repository imply inclusive or exclusive dependence relationships. This approach has been tested in a case study in our regional government and the simplicity of the process and the fast implication of the staff involved have been proved.

As mentioned in Section 2.1, the requirements imposed by the security standards are generally imprecise. Likewise, we think that some of our security requirements, expressed in natural language, may also be imprecise. As a first step, we consider that we can manage imprecision informally through i) the basic, good practice guidelines [26] (Section 2.2) regarding requirements writing; ii) the Repository Improvement activity in the SIREN process model (Figure 9); and iii) the fit criteria included in the SyTS and STS documents (Section 4.1). Nevertheless, we are conscious that complete imprecision management requires more advanced techniques to formalize both functional and non-functional requirements. Two lines of future research to tackle this problem are the following:

- In a use case-driven development process, functional requirements will be mapped into use cases or scenarios. Given that another interest in our research group consists of the formalization of the UML (*Unified Modeling Language*) (see [43] and [44]), further work is required to link the requirements, in particular the most critical, to the formal models obtained from them, so that they can be rigorously validated and verified.
- Further work is also needed to specify the interactions between security requirements and other types of non-functional requirements, as well as between functional requirements. To this extent, we can adopt a formal framework for integrating goals and goals refinement in requirements models, such as the KAOS methodology (*Knowledge Acquisition in Automated Specification of Software*) [45], the GBRAM methodology (*Goal-Based Requirements Analysis Method*) [16], or the NFR methodology (*Non-Functional Requirements*) [46] in order to capture and evaluate the alternative goals decomposition. Chung, for example, has already adapted the NFR framework to the security field [21].

Inconsistency management is another interesting aspect to be considered. Inconsistency can appear, for example, because the domains and profiles are not mutually exclusive. We think that in well defined profiles such as those coming from MAGERIT and the personal data protection law, inclusive and exclusive dependencies are enough to avoid most of the inconsistencies. Nevertheless, inconsistencies cannot be easily avoided in real applications dealing with multiples and less-formalized perspectives of the system.

Finally, further work will be the link from the SIREN requirements specification to the rest development artifacts (analysis, design, implementation), since we think that the integration of SIREN with the rest of the development process will improve the usefulness of the approach. In this regard, it is necessary to study the combination of SIREN with other methods, such as the Unified Process [47], and our previous research dealing with conceptual modeling (the integration of throwaway and evolutionary prototypes from formal requirements specifications to realize a rigorous and seamless design [48], and the concurrent definition of use cases and conceptual models from the business model [49]).

Acknowledgements

This research has been partially financed by the CICYT (*Science and Technology Joint Committee*), Spanish Ministry of Science and Technology, project SIRENrm TIC2000-1673-C06-02 "SImple REuse of software requiremeNts and rigorous modeling".

We would like to thank Vicente Rodes from the *Regional Ministry of Labor and Welfare* in our Regional Government and José Manuel Marqués from the University of Valladolid for their unselfish collaboration and their useful comments during this research work.

7 References

1. PITAC. (*President's Information Technology Advisory Committee*). Interim Report to the President. Setting Federal Research Priorities: Findings and Recommendations. 1998. *http://www.ccic.gov/ac/interim*

2. Lichtestein, S and Swatman, PMC. *Effective Internet Acceptable Usage Policy for Organisations*. In. *10th International Conference on Electronic Commerce (Bled'97)*. Slovenia. 1997. pp. 503-522.
3. *Constitutional Law 15/1999, of December 13, on Protection of private data of individuals in Spain. (Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal en España)*. 1999. (In Spanish)
4. *Real Decreto 994/1999, of June 11, in which the Ruling on security measures of automated files containing private data on individuals is passed. (Real Decreto 994/1999, de 11 de junio, por el que se aprueba el Reglamento de medidas de seguridad de los ficheros automatizados que contengan datos de carácter personal)*. 1999: p. 24241. (In Spanish)
5. Infosec. Information Security Breaches Survey. 2000. *http://www.infosec.co.uk*
6. *ISO/IEC Std. 15408. Evaluation Criteria for Information Technology Security*. 1999.
7. CCTA, *SSADM-CRAMM Subject Guide for SSADM Version 3 and CRAMM Version 2*. Central Computer and Telecommunications Agency, IT Security and Privacy Group, Her Majesty's Government, London. 1991
8. CLUSIF, *MARION version 98*. La Commission Méthodes du CLUSIF (*Club de la Securité des Systèmes d'Information Français*). 1998
9. MAP. *Metodología de Análisis y Gestión de Riesgos del Ministerio de Administraciones Públicas Español, MAGERIT v.1.0*. 1996. (In Spanish)
10. Kotonya, G and Sommerville, I, *Requirements Engineering. Processes and Techniques*. John Wiley & Sons. 1998
11. Robertson, S and Robertson, J, *Mastering the requirement process*. Addison-Wesley. 1999
12. Sommerville, I, *Software Engineering (6th edition)*. Pearson Education Limited. 2001
13. *IEEE Std 830-1998. Guide to Software Requirements Specifications (ANSI)*. The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection. 1999.

14. Mili, H, Mili, F, and Mili, A, *Reusing Software: Issues and Research Directions.* IEEE Transactions on Software Engineering, 1995. 21(6): p. 528-562.

15. Cybulsky, J and Reed, K. *Requirements Classification and Reuse: Crossing Domains Boundaries*. In. *6th International Conference on Software Reuse (ICSR'2000)*. Springer, Lecture Notes in Computer Science. Viena. 2000. pp. 190-210.

16. Antón, AI and Earp, JB. *Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems*. In. *1st ACM Workshop on Security and Privacy in E-Commerce (CCS 2000)*. Athens, Greece. 2000.

17. Lutz, R. *Analyzing software requirements errors in safety-critical embedded systems*. In. *Requirements Engineering (RE'93)*. San Diego, CA. 1993. pp. 126-133.

18. Skidmore, S, Farmer, R, and Mills, G, *SSADM version 4 models and methods, second edition*. NCC Blackwell. 1994

19. MAP. *MÉTRICA versión 3*. Ministerio de Administraciones Públicas. Secretaría de Estado para la Administración Pública. Consejo Superior de Informática. 2001 (In Spanish)

20. *Policy Framework for Interpreting Risk in eCommerce Security*. Technical Report CERIAS (*Center for Education and Research in Information Assurance and Security*). Purdue University. 1999. http://www.cerias.purdue.edu/techreports/public/PFIRES.pdf

21. Chung, L. *Dealing with Security Requirements during the development of Information Systems*. In: C. Rolland, F. Bodat, and C. Cauvet (eds.). *5th International Conference on Advanced Information Systems Engineering (CAISE'93)*. Spring Verlag. Paris. Berlin. 1993. pp. 234-251.

22. Jones, S, Wilikens, M, Morris, P, and Masera, M, *Trust Requirements in e-Business.* Communications of the ACM, 2000. 43(12): p. 81-87.

23. Fenton, N and Neil, M, *A Strategy for Improving Safety Related Software Engineering Standards.* IEEE Transactions on Software Engineering, 1998. 24(11): p. 1002-1014.

24. *IEEE/EIA Std 12207.0-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995. Standard for Information Technology- Software Life Cycle Processes*, in *Volume 1: Customer and Terminology Standards*. The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection. 1999.

25. *ISO 9000 International Standars for Quality Management. 4th edition.* ISO (*International Organization for Standarization*). 1994

26. Sommerville, I and Sawyer, P, *Requirements Engineering: A good practice guide*. John Wiley & Sons. 1997

27. Paulk, MC and Weber, CV, *The Capability Maturity Model: Guidelines for Improvement the Software Process*. Addison-Wesley. 1995

28. El Emam, K, Drouin, J-N, and Melo, W, *SPICE. The Theory and Practice of Software Process Improvement and Capability Determination*. IEEE Computer Society. 1997

29. Baskerville, R, *Information Systems Design Methods: Implications for Information Systems Development.* Computing Surveys, 1993. 25(4): p. 375-414.

30. Gabb, A. *The Requirements Spectrum*. In. *First Regional Symposium of the Systems Engineering Society of Australia (SE'98)*. Australia. 1998.

31. Pressman, RS, *Software Engineering. A practitioner's approach. Fifth edition.* Mc Graw Hill. 2000

32. *IEEE/EIA Std 12207.1-1997. Guide for Information Technology - Software life cycle processes - Life cycle data*. The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection. 1999.

33. *IEEE Std 1233-1998. Guide for Developing System Requirements Specifications*. The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection. 1999.

34. Toval, A, Olmos, A, and Rodero, JA. *The Role of Requirements Reuse in Protecting Personal Data*. In. *5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001)*. Orlando, Florida (USA). 2001.

35. Ramesh, B and Jarke, M, *Toward reference models for requirements traceability.* IEEE Transactions on Software Engineering, 2001. 27(1): p. 58-93.

36. Nuseibeh, B, *Weaving Together Requirements and Architectures.* IEEE Computer, 2001. 34(3): p. 115-117.

37. Vitech Corporation. CORE. 2000. *http://www.vtcorp.com*

38. Quality Systems & Software. DOORS. 2000. *http://www.qssinc.com*

39. Rational Software. Requisite Pro. 2000. *http://www.rational.com*

40. INVESDOC. IECISA. *http://www.ieci.es*

41. *MAGERIT risk analysis and management in the Regional Information Systems and Telecommunication Office.* Contract CARMMA 3142-UMU. CARM (*Autonomous Community of Murcia*) and the Software Engineering Research Group of the University of Murcia. 1999.

42. Jaaksi, A, *A Method for Your First Object-Oriented Project.* Journal of Object Oriented Programming (JOOP), 1998. 10(8): p. 17-25.

43. Fernández, J and Toval, A. *Can Intuition Become Rigorous? Foundations for UML Verification Tools*. In: T. FM (eds.). *The 11th Int. Sym. on Software Reliability Engineering*. IEEE Computer Press. San José (California). 2000. pp. 344-355.

44. Toval, A and Fernández, J. *Improving System Reliability via Rigorous Software Modeling: The UML Case*. In. *2001 IEEE Aerospace Conference (track 10: Software and Computing)*. IEEE Computer Society. Montana, USA. 2001.

45. Dardenne, A, van Lamsweerde, A, and Fickas, S, *Goal-Directed Requirements Acquisition.* Science of Computer Programming, 1993. 20: p. 3-50.

46. Chung, L, Nixon, B, Yu, E, and Mylopoulos, J, *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers. 2000

47. Jacobson, I, Booch, G, and Rumbaugh, J, *The Unified Software Development Process*. Addison-Wesley Longman, Inc. 1999

48. Moros, B, Nicolás, J, García, J, and Toval, A, *Combining Formal Specifications with Design by Contract.* Journal of Object Oriented Programming (JOOP), 2000. 12(9): p. 16-22.

49. García, J, Ortín, M, Moros, B, Nicolás, J, and Toval, A. *Towards Use Case and Conceptual Models through Business Modeling*. In: L.S. Laender AHF, Storey VC (eds.). *19th International Conference on Conceptual Modeling, ER'2000.* Springer. Salt Lake City (Utah, USA). 2000. pp. 281-294.

Figure Legends

Figure 11. A brief summary of our proposal for requirements reuse, particularized to the security field. The structure of the requirements specification documents is explained in section 4.1.

Figure 2. MAGERIT assets hierarchy.

Figure 3. An extract of the MAGERIT information system layer.

Figure 4. SIREN requirements documents hierarchy.

Figure 5. Reusable requirements repository composed of profiles and domains. Each profile or domain is a view of the global repository. In the figure, only the SRS document is shown.

Figure 6.. Examples of system (SyRS) and software (SRS) requirements. Labels R1-R4 are only used as abbreviations.

Figure 7.. Example of exclusive trace between requirements in the SRS. Section 3.4 in the SRS deals with the design constraints and includes the subsection 3.4.3. *Related Applications*.

Figure 8.. Spiral model for RE adapted to SIREN.

Figure 9. SIREN approach for requirements reuse.

Figure 10. Accumulated values in an extract of the asset hierarchy in our case study. To preserve privacy, the name of the assets and IP addresses have been changed.

Table legends

Table 3 Sommerville and Sawyer's top ten guidelines.

Table 2. Results of the application of SIREN for security requirements