Eight key issues for an effective reuse-based requirements process

Article *in* Computer Systems Science and Engineering · November 2008

CITATIONS
25

READS
705

4 authors:

Ambrosio Toval
University of Murcia
177 PUBLICATIONS **2,458** CITATIONS
SEE PROFILE

Begoña Moros Valle
University of Murcia
27 PUBLICATIONS **297** CITATIONS
SEE PROFILE

Joaquín Nicolás
University of Murcia
50 PUBLICATIONS **611** CITATIONS
SEE PROFILE

Joaquín Lasheras
Universidad Católica San Antonio de Murcia
20 PUBLICATIONS **256** CITATIONS
SEE PROFILE

EIGHT KEY ISSUES FOR AN EFFECTIVE REUSE-BASED REQUIREMENTS PROCESS

Ambrosio Toval, Begoña Moros, Joaquín Nicolás, Joaquín Lasheras

Software Engineering Research Group. Departamento de Informática y Sistemas. Universidad de Murcia.

30071 Campus de Espinardo, Murcia, Spain.

{atoval, bmoros, jnr, jolave}@um.es

Abstract

One of the most helpful Requirements Engineering (RE) strategies for improving the productivity and quality of software process and products is requirements reuse. In this paper we identify eight key issues to be considered for an effective and practical reuse-based RE process. The issues are the result of the research on the subject and the experience of the authors in defining and applying a reuse-based RE method. Analysis of some of the most popular current CARE (*Computer-Aided Requirements Engineering*) tools has revealed a lack of automated support for these features and, to fill this gap, a practical technical solution is provided. The key issues have been validated in the context of a real case study.

Keywords: Requirements Engineering, Requirements reuse, Computer-Aided Requirements Engineering (CARE).

1 Introduction

The negative consequences of an inappropriate requirements management in the system development cost and functionality are well-known (Glass 1998; Smith 2001; Glass 2002). However, a survey of the state of the practice in RE (Neill and Laplante 2003) reveals that "fifty-two percent of respondents did not think that their company did enough RE". We think this situation could be a consequence of the difficulties encountered by managers when putting the results of current research into practice. The fact is that there is a gap between the state of the art and the state of the practice (Reifer 2003). As stated in (Wiegers 1998), to fill this gap it is not necessary to invent a new method but to make existing methods more attractive and less time consuming, in short, easy to put in practice. Other approaches like the one proposed by (Sindre, Firesmith et al. 2003) also underline the benefits of practical work.

In our view, practical work in RE should take into account reuse. According to (Mili, Mili et al. 1995) "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". There is a consensus —see, for example, (Prieto-Díaz 1993), (Rothenberger, Dooley et al. 2003)— on the many benefits of reuse, and on these being greater when the abstraction level is increased and not only code, but also designs and specifications, are reused (Cybulsky and Reed 2000; Sommerville 2004). This fact has been empirically demonstrated by (Rine and Nada 2000) who have proved that the reuse level determines the effectiveness of productivity, quality and time-to-market improvements. Elsewhere, John Favaro states (Favaro 2002) that "A well-formulated, measurable, reusable requirement […] is every bit as valuable as a reusable software module".

In order to take advantage of the benefits of reuse at the requirements level, our group proposed the SIREN (*SImple REuse of RequiremeNts*) method (Toval, Nicolás et al. 2002). This is a practical approach for selecting and specifying the requirements of a software system based on requirements reuse and software engineering standards. SIREN encompasses a spiral process model, requirements documents templates, a reusable requirements repository which is organized by catalogs and a supporting CARE tool called SirenTool. With SIREN we have

tried to bring together the state of the art and state of the practice by providing a flexible and lightweight approach to reduce the requirements time.

The experience gained through the SIREN method definition and application (Toval, Nicolás et al. 2002; Toval, Olmos et al. 2002), and research on the subject, has led us to identify eight key issues that, in our opinion, have to be taken into account for any reuse-based requirements method to succeed. These key issues are collected, described and justified homogeneously in the paper. Moreover, since these issues should be supported by a CARE (*Computer-Aided Requirements Engineering*) tool, an analysis of the state of the art of some of the most popular commercial tools with respect to requirements reuse was carried out, and the conclusions are presented in this paper. The study revealed the lack of requirements reuse support of the commercial tools analyzed and this led us to propose SirenTool as a solution. The key issues have been validated in a real case study in the context of clinical history management in the intensive care unit of a hospital.

In our view, the above contributions may be of interest to CARE tool builders (concerned with the inclusion of reuse features in their tools), researches in the field and practitioners involved in the adoption of reuse in their requirements engineering processes.

This paper has been structured as follows: Section 2 briefly presents the SIREN method. Section 3 explains the key issues identified for a practical reuse-based requirements process. Section 4 summarizes the study of the CARE tools analyzed. Section 5 shows how the key issues have been implemented in SirenTool. Section 6 presents other approaches related to requirements reuse. Finally, Section 7 gives the conclusions.

2 The SIREN Method

Like (Robertson and Robertson 2006), we believe that by starting from a set of requirements which have been specified for other projects or domains, we can improve the precision and efficiency of the requirements specification for the current project and we can also reduce the time necessary to elaborate this specification. However, systematic approaches to requirements reuse are still scarce (see Section 6). In order to explore the benefits of requirements reuse, we proposed SIREN (*SImple Reuse of softwarE requiremeNts*) as a practical way of dealing with requirements reuse. In this section we summarize the main features of SIREN, a detailed explanation can be found in (Toval, Nicolás et al. 2002).

SIREN encompasses a spiral process model, requirements documents templates and a reusable requirements repository which is organized by catalogs. SIREN requirements catalogs correspond to a set of generic and reusable requirements named *profiles* ("horizontal" application domains, for instance, concerning security, or personal data protection regulations) and *domains* ("vertical" application domains, for instance, insurance or banking). The separation of catalogs into profiles and domains is not relevant beyond helping to identify, organize and group sets of related requirements. These catalogs are organized in a hierarchy of requirements specification documents, which are structured according to IEEE standards (IEEE 1999a; IEEE 1999b). Some examples of catalogs developed in the context of SIREN are:

- The *PDP (Personal Data Protection) profile catalog* (Toval, Olmos et al. 2002), with the requirements coming from the Spanish Personal Data Protection Constitutional Law (This Law is an adaptation of the EU Directive 95/46/CE (EU 1995)).
- The *Security profile catalog* (Toval, Nicolás et al. 2002), with the requirements coming from MAGERIT (MAGERIT is the information systems risk analysis and management method of the Spanish public administration. It is based on ISO/IEC

15408-1999 Evaluation Criteria for Information Technology Security Standard, also known as the Common Criteria Framework –CCF).

- The *TOS (Teleoperated Systems) domain catalog* (Nicolás, Lasheras et al. 2006), modeling the requirements of the product line of teleoperated systems (basically, robotics systems) for ship hull maintenance operations, such as cleaning or painting the ship hull.

Each requirement is labeled with a *type* in the catalogs. A type denotes the catalog the requirement comes from and the requirements specification document where the requirement is included. For example, the types SYRSP and SRSP refer to requirements contained in the SyRS (*System Requirements Specification*) and SRS (*Software Requirements Specification*) documents within the PDP (P) catalog, respectively.

In SIREN, the textual information of a requirement is complemented by a set of attributes. There is a set of attributes common to all requirements (including *priority, rationale, source, state*, etc.), although additional attributes can be defined depending on the type of the requirement. For example, both SRSP and SYRSP types include an additional attribute called *security level.*

Besides the attributes, different traceability relationships can be defined to relate requirements. These are inclusive, exclusive and parent-child traceability relationships (see issue K5 in Section 3). In SIREN we also have *parameterized requirements*, which contain some parts that have to be adapted to each application or system and that have to be instantiated when it is reused.

The SIREN process model is an adaptation of the spiral process model proposed by (Kotonya and Sommerville 1998), but it includes the repository as a central element of the process. Consequently, new activities appear, such as *Requirements Selection* and *Repository Improvement*, and others are adapted to the new circumstances, such as the *Requirements Elicitation* (renamed as *Specific Requirements Elicitation* in SIREN) and the *Analysis and Negotiation* activities (see Fig. 1):

- *Requirements selection*. The approach for reuse consists of providing the requirements engineer with the specification documents templates, with the requirements filled in, and which are stored in the repository. The requirements engineers together with the other stakeholders (e.g. users, clients, developers, etc.) will reuse those requirements which are suitable for the specific application that is being developed by instantiating them when needed.

- *Specific requirements elicitation*. In the meetings with the rest of the stakeholders, the requirements engineers gather the informal and specific requirements of the current problem domain. These requirements are not initially in the repository.

- *Analysis and Negotiation.* As a result of the requirements analysis and negotiation, we will probably have to go back to the repository to change a previously selected requirement for another less costly one. Thus, the negotiation could lead us to remove some requirements and select new ones from the repository.

- *Repository improvement*. The repository should not be considered as a final static product, but as an evolutionary one to be enriched with new reusable requirements. Thus, the requirements in the repository could be modified in order to improve their quality.

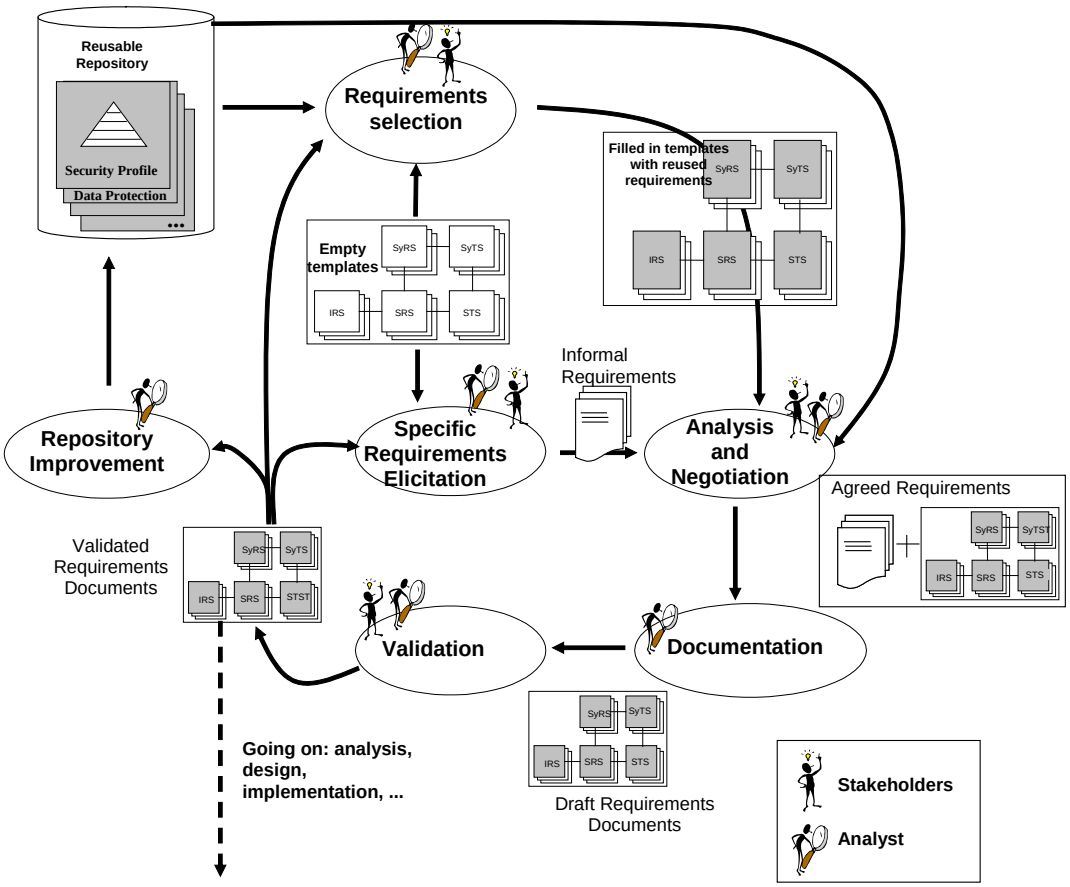

Fig. 1. SIREN process model for requirements reuse.

3 Key Issues in Requirements Reuse

The lessons learned from the SIREN application —in the security (Toval, Nicolás et al. 2002) and data protection fields (Toval, Olmos et al. 2002), as well as other related experiences closer to the domain analysis (Nicolás, Lasheras et al. 2006) or audit (Martinez, Lasheras et al. 2006)— together with the analysis of related research and current requirements management tools led us to state eight key issues that should be taken into account to achieve a practical reuse-based RE method:

K1. Organization of the reusable requirements

K2. Search engine for reusable requirements

K3. Requirements selection and reuse with different granularity levels

K4. Requirement attributes reuse

K5. Traceability relationships reuse

K6. Parameterized requirements management

K7. Repository improvement

K8. CARE support to reuse

We think that these key aspects are simple enough for organizations which are currently immature in relation to RE to adopt easily and that they will affect the business positively. A recent empirical study (Sommerville and Ransom 2005) reveals that improvements in the requirements engineering process led to business improvements. This section seeks to explain

these aspects and justify their importance, as well as giving examples of how the SIREN method supports them.

These key issues (including the feasibility of the tool developed) have been validated in the context of a real case study regarding the development of an information system for the clinical history management in the intensive care unit of the Hospital of Getafe in Madrid. The main lessons learned can be found in Section 7, Conclusions.

K1. Organization of the reusable requirements

We share the view of (Lam, MacDermid et al. 1997) that the process of creating reusable requirements is aided by having a road-map for structuring the domain and for organizing reusable requirements knowledge. Having the requirements structured and organized is a priority in facilitating the reuse process, since the time devoted to search requirements from the repository is reduced.

In SIREN, for instance, we have chosen to organize the reusable requirements by means of catalogs that use some predefined templates of documents. A catalog groups a set of related requirements corresponding to an application domain or a profile (see Section 2). Nevertheless, another structure could be chosen, such as Pattern Book (Robertson 1996), use cases (Alexander and Kiedaisch 2002) or structure maps (Pisan 2000).

K2. Search engine for reusable requirements

Another mandatory issue when talking about requirements reuse is the ability to select the requirements that are most suitable for a new project. The best way to provide this feature is by allowing the requirements engineer to define a wide range of search criteria to facilitate that selection. For instance, in the case of documents templates, like in SIREN, it may be necessary to search for the requirements specified in a particular section of a document or those requirements related to a specific concern of the stakeholder (e.g. “legal issues”). Another example could be to search for the requirements containing a specific word, such as the requirements including the word “firewall” from the Security catalog. Note that this key issue is directly related to *K8. CARE support to reuse*, since the search engine should be implemented in the context of a complete requirements management tool where the ability to choose requirements from the result set of the search has to be provided too.

In short, a reuse-based RE method should provide an easy-to-use search engine which satisfies all the necessities of the requirements engineers at reuse time.

K3. Requirements selection and reuse with different granularity levels

It should be possible to define the granularity level needed at reuse time. This means that the reused requirements set could range from a single requirement or a subset of requirements coming from some of the available catalogs to a complete catalog. There are other ways of reuse, for example, using a "base-document”, that is, a document filled in with requirements. We consider that this latter kind of reuse is not efficient enough because it is mandatory to reuse a fixed set of requirements, when only one subset may be sufficient. If different granularity levels are not taken into account, reuse will lose efficiency because a lot of time will be wasted in eliminating requirements which are not necessary for the new project.

K4. Requirement attributes reuse

A requirement should be described by means of a set of attributes. These attributes should be handled carefully at the time of reuse, so that when a requirement is reused its attributes will be reused too. For the sake of flexibility, the requirements engineer should be able to modify

any attribute at reuse time, so the problem arises when s/he decides to change the type of the requirement (for example, a PDP requirement is reused as security requirement in the case of SIREN catalogs, see Section 2) because the set of attributes belonging to each type may be different. Consequently, it is useful to define a minimum set of attributes which every type of requirement will have in common in order to provide consistency at reuse time. As a result, whatever the type of the requirement reused, it will contain at least the common set of attributes. In SIREN a minimum common set of attributes has been defined based on the IEEE 1233 standard (IEEE 1999b). Note that the value of the requirements attributes could change from one project to another but the list of the attributes involved for a requirement type will remain at reuse time. Later, new attributes could be defined (if needed) throughout the current project development.

K5. Traceability relationships reuse

In general, in a requirements document the requirements are not isolated, but appear as related to each other by means of traces. Consequently, the existence of these traceability relationships should be taken into account at reuse time. (Ramesh and Jarke 2001) propose a wide traceability taxonomy, but they do not state which type of dependencies should be traced to support a goal like reuse (Knethen, Paech et al. 2002). On the other hand, we consider this taxonomy too large to be used in a practical RE process. Thus, we propose a reduced set of horizontal traceability relationships, that is, those which relate entities in the same software artifacts, specifically, relationships between requirements. The related requirements could come from the same document (e.g. all of them from the SRS) or different documents (e.g. one requirement from the SyRS links to another in the SRS). The traceability model includes:

- *inclusive relationships*
- *parent-children relationships*
- *exclusive relationships*

The *inclusive traceability relationships* are relationships between two requirements A and B, which means that to satisfy A, B also needs to be satisfied and, therefore, the reuse of A will imply the reuse of B. As an example of an inclusive trace, we show the following, from the security catalog related to the information system:

> ***R1*** *The authorized users of the information system shall know their duties related to access control and the information under their responsibility. Thus, three conditions are stated:*
>
> ***R1.1*** *The authorized users have to use their password properly.*
>
> ***R1.2*** *The authorized users can not for a moment leave the information that they manage unsupervised.*
>
> ***R1.3*** *The authorized users have to follow the security measures enforced to avoid unauthorized accesses to the information managed by them.*
>
> ***R2*** *Documents and diskettes shall be kept under lock and key when they are not being used and out of office hours.*
>
> ***R3*** *The operating systems used shall provide password mechanisms to control or limit the access of the users.*

We can observe that for R1 to be fulfilled, R2 and R3 (and possibly others which are not shown) have to be accomplished.

The *parent-children relationships* are relationships where the children requirements refine the meaning of the parent requirement. The definitions of the children requirements are next to the parent definition and in the same document. The previous piece of requirements specification shows an example where R1.1, R1.2 and R1.3 refine the information given in R1.

We recommend using parent requirements to describe the object, function or constraint needed, in an abstract and general way, and children requirements to refine the contents of the parent. By separating the abstraction levels in this way, the variability of the requirements can be better controlled.

The *exclusive traceability relationships* mean that the requirements implied are mutually alternative. This relationship is directly related to reuse since it indicates those requirements that cannot be selected from the repository for the same project. An example of two exclusive requirements, extracted from the SIREN Security catalog, is:

> ***R1.*** *"The firewall configuration will be screened host,"* and
>
> ***R2.*** *"The firewall configuration will be screened subnet"*

Although R1 and R2 could be specified as a parameterized requirement stating that "The firewall configuration will be [a configuration]", the specification as two (or more) separated and exclusive requirements allows us to link each requirement to another dependent one by means of an inclusive relationship.

The exclusive relationship minimizes the risk of inconsistencies. Otherwise it would be difficult to perceive that two requirements with alternative meanings have been reused without the advice of an expert in the domain of the catalog from which the requirements are being chosen.

Note that inclusive and exclusive traceability relationships in SIREN are analogous to the mandatory and alternative features of FODA (Kang, Cohen et al. 1990) and FORM (Kang, Kim et al. 1998).

At reuse time, the requirements engineer should be aware of the requirements relationships so as to decide which of them must remain in the new project in order to assure completeness. It would be helpful if the requirements engineer could visualize these traceability relationships in a friendly way, as this would assist in the decision-making process.

When the requirements specification evolves, traceability relationships can be used to assess the impact of changing a requirement on the requirements that depend on it. Thus, change management can also benefit from a suitable traceability relationships management.

K6. Parameterized requirements management

Parameterization is a powerful concept that has been extensively used in programming to favor reuse. In the case of RE, this concept has been translated into *parameterized features* in FODA (Kang et al., 1990) and FORM (Kang, Kim et al. 1998) and *parameterized requirements* (Lam, MacDermid et al. 1997) (Toval, Nicolás et al. 2002) (Firesmith 2004). Parameterized requirements contain some parts that have to be adapted to each application or system and that have to be instantiated at reuse time. This kind of requirement encourages reuse by factoring out system-specific details as parameters of the requirement, thus enabling engineers to specify requirements with a higher degree of flexibility.

An example in SIREN, taken from the PDP catalog, is:

> *The person in charge of the file will use [hardware backup] for the backup.*

In a reuse-based RE process, the requirements engineers should be able to define and instantiate parameterized requirements. At instantiation time it would be advantageous to have the suitable values available; for instance, in the example above, the possible values for "hardware backup" could be: *magnetic tape*, *optical disk* or *magnetic disk* (see Fig. 5).

K7. Repository improvement

While the repository is used, the experience gained will be used to: i) improve the quality of the requirements included in it—see activity "Repository improvement" in the SIREN process model shown in Fig. 1—and ii) to modify the content, so new requirements to be used in further developments should be included. Thus, the repository should not be considered as a final static product, but as an evolutionary one. Consequently, we will take advantage of one of the main benefits of reuse: quality improvement. For instance, in the case of parameterized requirements management, and returning to the example presented above, a new value for the "hardware backup" parameter could be added to the repository. This issue also means that a clear repository change process has to be defined.

K8. CARE support to reuse

CARE tools facilitate the RE process or, simply, make it possible (Sommerville 2004). Without automatic support, we think that the application of any RE method concerning reuse becomes slow, cumbersome and thus impractical. In particular, several of the previous key issues (namely K2, K3, K4, K5, and K6) only make sense in practice if automated support is available. That is the reason why SirenTool is proposed (see Section 5 for a complete description of the tool). SirenTool is a repository-based requirements tool developed as a way to improve the requirements specifications that are produced. It is in the spirit of Firesmith's work (Firesmith 2003) that considers requirements reuse a critical property of the requirements tools. SirenTool allows the requirements engineer to select old requirements from the reusable requirements repository and accept them as they are or change whatever the engineer wants (see Section 5).

4 CARE Tools Analysis

In this section we present an analysis of several CARE tools with the aim of checking the reuse capabilities that they provide (Section 4.1). The conclusions and design decisions for developing SirenTool are then discussed (Section 4.2).

4.1 Comparative Framework

In order to find out how current CARE tools support the key issues described in Section 3, the state of the art was studied. For this purpose a comparative framework was established. Two aspects were considered for its definition: (i) the key issues identified for a practical requirements reuse, and (ii) the general needs of a requirements management tool. The survey on requirements management tools carried out by INCOSE (INCOSE 2006) was used as the main reference for the second aspect since it keeps a wide updated comparison of tools from the responses provided directly by the tool vendors.

The tools chosen for the comparison were: Caliber-RM (Caliber-RM 2005), DOORS (Doors 2003) and RequisitePro (RequisitePro® 2006). They were selected because they are well known (especially RequisitePro and DOORS, although CALIBER-RM is growing in importance). It is no coincidence that they appear in all the comparisons we studied, such as (Alexander 2003; Matulevicius 2005; INCOSE 2006; Volere 2006), and they are also the only three tools for documenting requirements mentioned in a remarkable paper on Requirements Engineering and Technology Transfer (Kaindl, Brinkkemper et al. 2002). In addition, the

developing companies provide wide support and, what is more important, all the tools include the capacity to extend their functionality.

Table 1 summarizes the features considered necessary for our reuse-centered requirements management tool. The table only shows the key issues for reuse described in Section 3 since the three tools meet, to a certain extent, the main features of a requirement management tool—see (INCOSE 2006).

ISSUE	***REQUISITEPRO***	***CALIBER-RM***	***DOORS***
K1. Organization of the Reusable requirements	Yes, by requirements type		Yes, by object type
K2. Search engine	Requirements cannot be selected from the search result.		
K3. Different reuse granularity levels	Copy -paste and base documents		Copy -paste and "module-based" reuse
K4. Attributes reuse	When the requirement is copied and pasted		
K5. Traceability reuse	Only parent-children traces	No	
K5.1. Exclusive requirements	No		
K5.2. Shows existing traceability links	Yes, graphically		
K5.3 Inconsistencies management	Yes, graphically with a traceability matrix		
K6. Parameterized requirements	No		
K7. Repository improvement	Not explicitly addressed		

Table 1. Summary of the comparison.

4.2 Conclusions of the analysis and design alternatives

Although the three tools analyzed are general requirements management tools, our study concludes that requirements reuse is not provided explicitly by any of them. None of them encompasses the key issues identified for automated support for requirements reuse. The only way to reuse is through the options of the *Edition* menu of the tools, that is, you can only "copy and paste" requirements already defined. For instance, none of the three tools offers parameterized requirements management, requirements traceability relationships reuse, or exclusive requirements management. However, all these tools allow you to reuse by means of base documents, although, in our view, reuse with different degrees of granularity is more suitable, since it allows the requirements engineer to reuse in a wider range, from single requirements to complete catalogs. Another recent study (Matulevicius 2005) draws the same conclusion, stating that one of the limitations of the commercial RE-tools is the weak support of requirements reuse.

As a result of the study the SRS document of a CARE tool to support the SIREN method was developed iteratively. As long as any of the tools analyzed was suitable to be adopted directly to support requirements reuse, two options were considered: (i) adapting one of these tools, or (ii) developing a new tool from scratch. The first option was preferred since it was found to be cost-effective and it satisfied our requirements. RequisitePro was chosen as the base tool that would be extended in order to solve the problems identified in Table 1. The main reason for choosing RequisitePro was the previous experience of the group, since RequisitePro had been used as support tool in previous research related to SIREN (Toval, Nicolás et al. 2002; Toval, Olmos et al. 2002).

5 Automated support for key issues

By developing an add-in for RequisitePro, we took advantage of the general requirements management capabilities provided by RequisitePro and thus only the reuse capabilities needed to support the key issues had to be implemented. SIRENA (*SIREN Automated*) is the name we have given to the add-in programmed in VisualBasic using the extensibility interface of RequisitePro. Thus, SIRENA is the name of the menu option in the Tools menu of RequisitePro that gives access to the reuse capabilities (see Fig.2). We have called the whole solution SirenTool; that is, RequisitePro plus the add-in SIRENA, since this is the solution proposed to support the SIREN method.

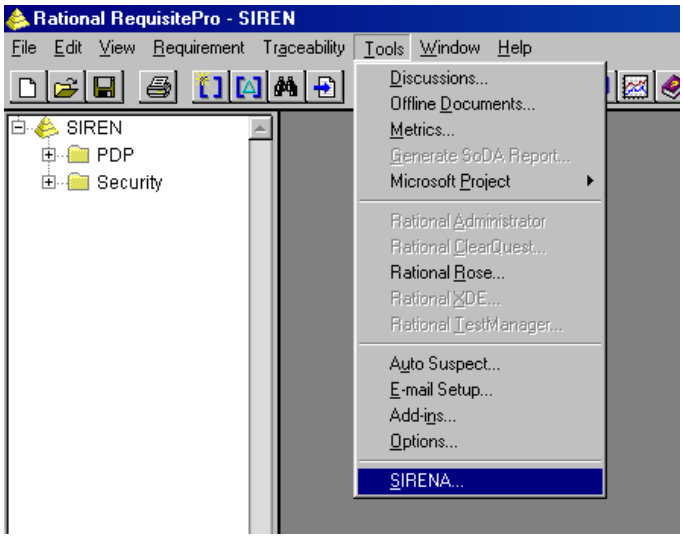

Fig. 2. Main screen of the application.

At reuse time, SirenTool would be used as follows: firstly, the project called SIREN, which corresponds to the reusable requirements repository, is opened in RequisitePro. More precisely, the SIREN project includes one folder for each catalog defined in SIREN (Fig.2 shows the Security and PDP catalogs). Next, the SIRENA menu option is selected and a search window to facilitate the requirements selection is opened (see Fig. 3). Finally, the requirements to be reused will be available in the project chosen by the requirements engineer, which is processed as a standard project in RequisitePro. Hereafter, the requirements engineer will be able to include the requirements in the suitable documents within the project, modify them or delete them.

The following sections show how SirenTool —since both RequisitePro capabilities and SIRENA capabilities are needed— supports the key issues identified in Section 3, using examples from the requirements sessions of the case study.

K1 Organization of the reusable requirements

In order to structure and organize requirements, RequisitePro allows us to define templates of documents which can be adapted to the current project (see Table 1). Besides, new types of requirements can be defined in RequisitePro in order to better organize the reusable requirements.

As mentioned in Section 2, in SIREN we have chosen to organize the reusable requirements by means of catalogs that use some predefined templates of documents according to IEEE standards. RequisitePro can be considered a repository based tool but also as a document based tool, so it helps to organize the SIREN catalogs as a hierarchy of documents.

K2 Search engine for reusable requirements

As explained in Section3, in order to select the most suitable requirements for the new project not only a search engine has to be provided but also the capacity to choose requirements from the result set. Next, the selected requirements will be added, in a controlled way, to the current project. However, although RequisitePro allows us to define complex search criteria, the result set obtained can only be copied and pasted and cannot manage the attributes and traces of the requirements as proposed in key issues K4 and K5.

An easy-to-use search engine has been developed in SirenTool in order to satisfy all the necessities of the requirements engineers at reuse time. Searches of requirements can be carried out through the *Search interface* (Fig. 3) according to the contents of the text of the requirement or according to the values of some of its attributes. Fig. 3 shows the set of attributes referred in Section 3, key issue K4.

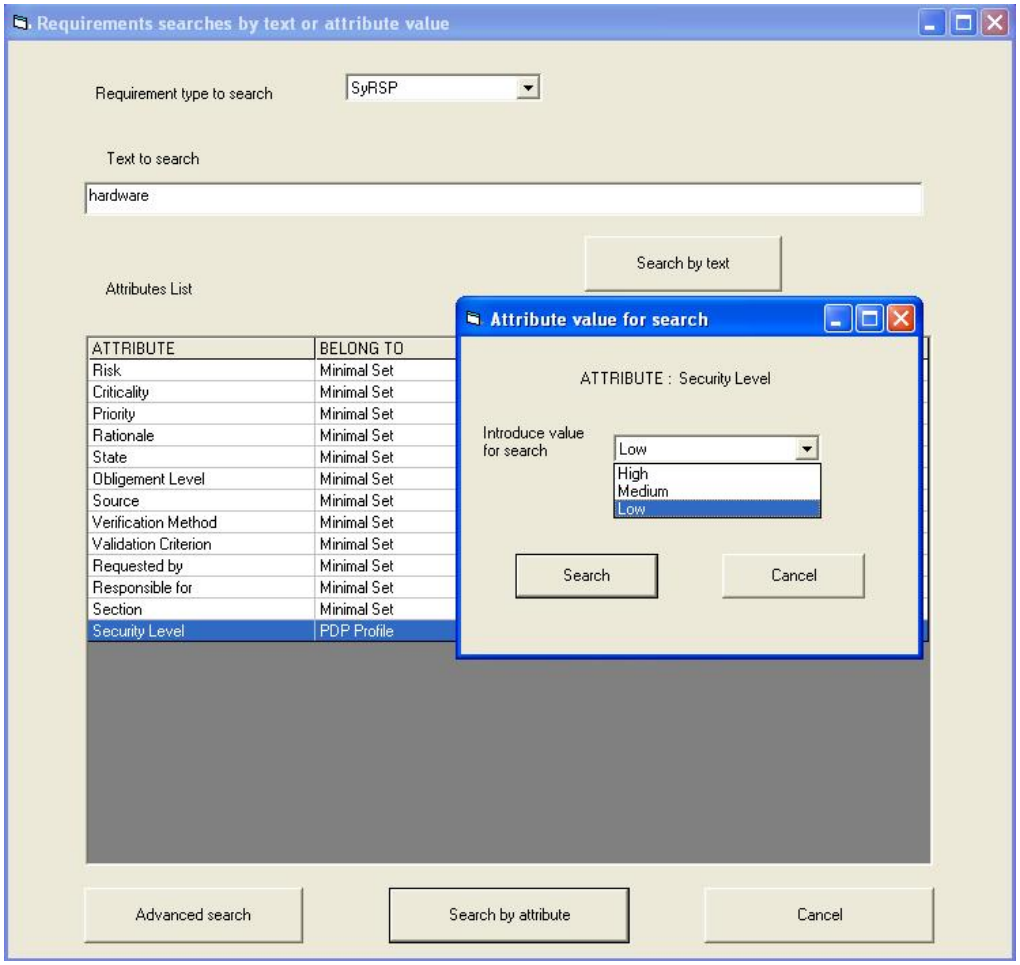

Fig. 3. Search interface.

Firstly the requirement type to be sought has to be selected, for instance *SRSS*, *SRSP*, *SyRSS* and *SyRSP*, which refer to the requirements contained in the SRS and SyRS documents, corresponding to the security (S) and PDP (P) profiles. The option of searching through all the requirements defined in the SIREN repository is also possible.

The case study used to validate the key issues concerns clinical history management in the intensive care unit of a hospital. Therefore, this unit manages sensitive data. In this context, SirenTool would help the requirements engineer to take from the PDP catalog those requirements enforced by the personal data protection law. According to the running example, Fig. 3 shows the search criteria regarding requirements in the SyRS of the PDP profile,

containing the word “hardware” and security level with value equal to “low”. Although the data managed by the information system that is going to be developed for the hospital correspond to a high security level, requirements from the low security level have to be accomplished too, according to the law.

K3 Requirements selection and reuse with different granularity levels

Once the search criterion has been introduced, a new interface with the result will be shown in SirenTool. For each requirement belonging to the result, the interface will show: type and text. The requirements which we want to reuse can be selected from this interface, from a single requirement to a result set that could correspond to a complete catalog.

Next, the *Interface for requirement reuse* (Fig. 4) will be shown once for each requirement chosen, and it will be used to support K4, K5 and K6. This interface shows the details of a requirement: type, text, attributes, and traceability relationships (inclusive, exclusive and parent-children). The reason for showing all this information is to make reuse easy. At reuse time the context of a requirement is needed to facilitate the analysis and negotiation.

To maintain consistency, the tool ensures that the same requirement is not reused more than once in the same project.

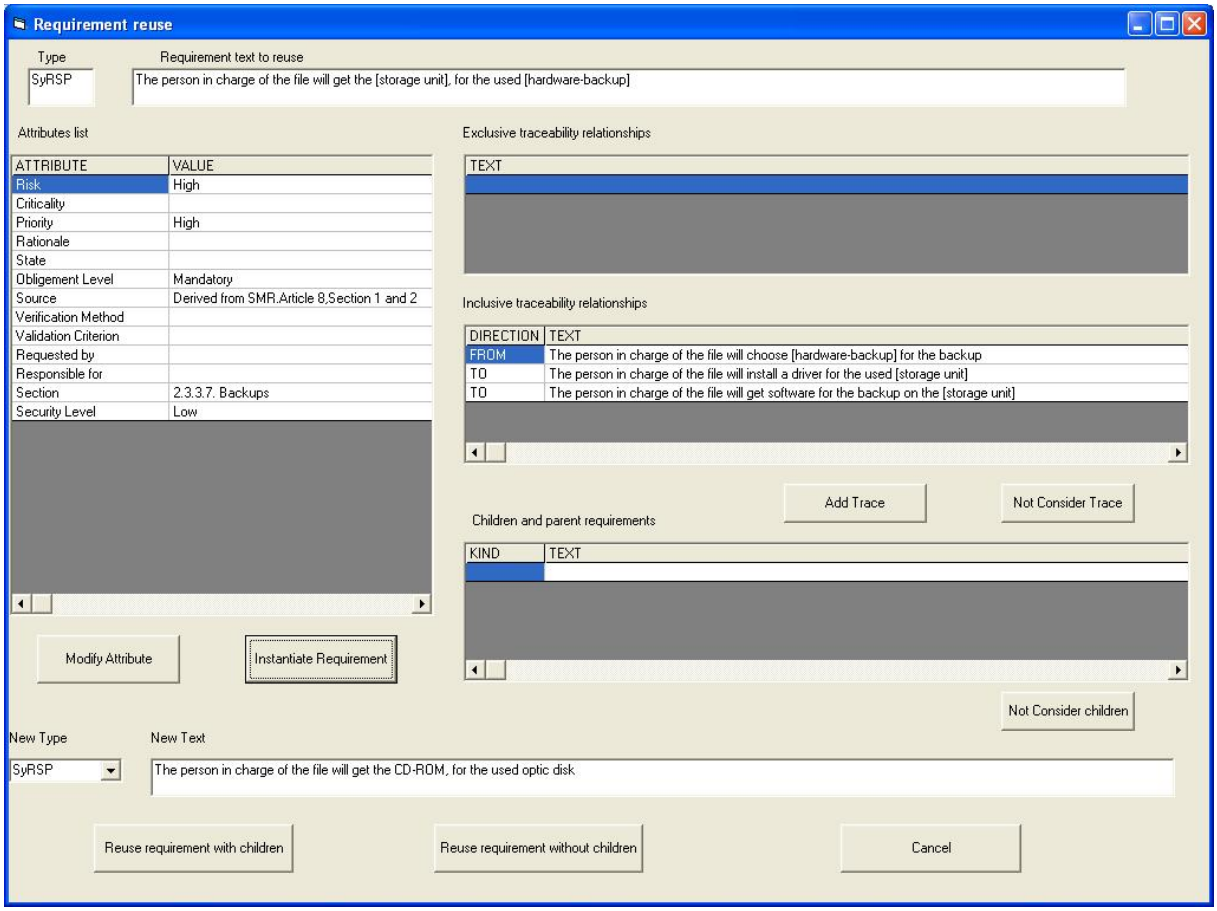

Fig. 4. Interface for requirement reuse.

K4 Requirement attributes reuse

Once a requirement has been selected, the requirements engineer can decide whether to adapt it to the new project. This functionality is provided through the *Interface for requirement reuse* (Fig. 4). For instance, the requirements engineer could modify the requirement text, the type of the new reused requirement (which by default is the same one), or the value of any attribute. The requirement type influences the set of attributes. If the type of the requirement

is maintained when it is reused, then this requirement will have the same attributes as the *source requirement* (the original requirement stored in the repository). Otherwise, only the intersection of the attributes will be copied, including, at least, the attributes from the common set.

A *source* attribute is defined to keep track of the relationship between the source and reused requirements. The value of this attribute for the requirement which is going to be added in the current project is always established automatically at reuse time. As a result, given a requirement, any requirements engineer can check what the original requirement specification was and whether some values were modified when the requirement was selected.

K5 Traceability relationships reuse

SirenTool takes into account traceability relationships defined in SIREN at the time of reuse. Thus, if a requirements engineer tries to add a requirement which has inclusive trace relationships to others which have not yet been reused, the requirements engineer will be notified. The requirements engineer can then either not consider these trace relationships (*Not Consider Trace* button, see Fig. 4) or insert the related requirements (*Add Trace* button). In the second case, the details of the related requirements through the trace relationships will be shown, one by one, in an interface like the one described in Fig. 4. Finally, if these requirements had other trace relationships, the process would be repeated until all the related requirements had been shown.

Similarly, we have two options with regard to the children relationships: reuse the requirement considering them (button named *Reuse requirement with children*, Fig. 4) or without considering them (*Reuse requirement without children* button). In the first case, each of the children requirements selected in the *Children Requirements* list will be shown one by one again in the *Interface for Requirement Reuse* (Fig. 4) so that the requirements engineer can change them accordingly.

The exclusive traceability relationships are also maintained in SirenTool, so the requirements engineer is notified when s/he tries to reuse a requirement which has an exclusive relationship with another already reused one.

According to the running example, Fig. 4 shows the requirement chosen by the requirements engineers from the result set of the search, the specification appears in the field labeled “Requirement text to reuse”. This requirement is related inclusively to three others requirements. The first one has the direction “FROM” that means that it should have been reused previously for the fulfillment of the current requirement. The other two requirements have the direction “TO”, which means that they depend on the current requirements and the requirements engineer should consider whether to include them. For the selected requirement neither exclusive nor parent-children traceability relationships have been defined.

Before reuse, the requirements engineer is able to visualize traceability relationships through the Interface for Requirements Reuse (Fig. 4). After reuse, the requirements engineer will be able to visualize traceability relationships using RequisitePro (see issue K5 in Table 1).

K6 Parameterized requirements management

In the case that the reusable requirement is parameterized, it is possible to instantiate it before adding it to the current project. By clicking on the *Instantiate requirements* button (Fig. 4) a list with all the parameterized parts of the requirement with the associated type will be shown (Fig. 5). The types of parameter will have been defined previously when the requirements were included in the catalog, and those types will have some predefined values for the requirements engineer to choose from.

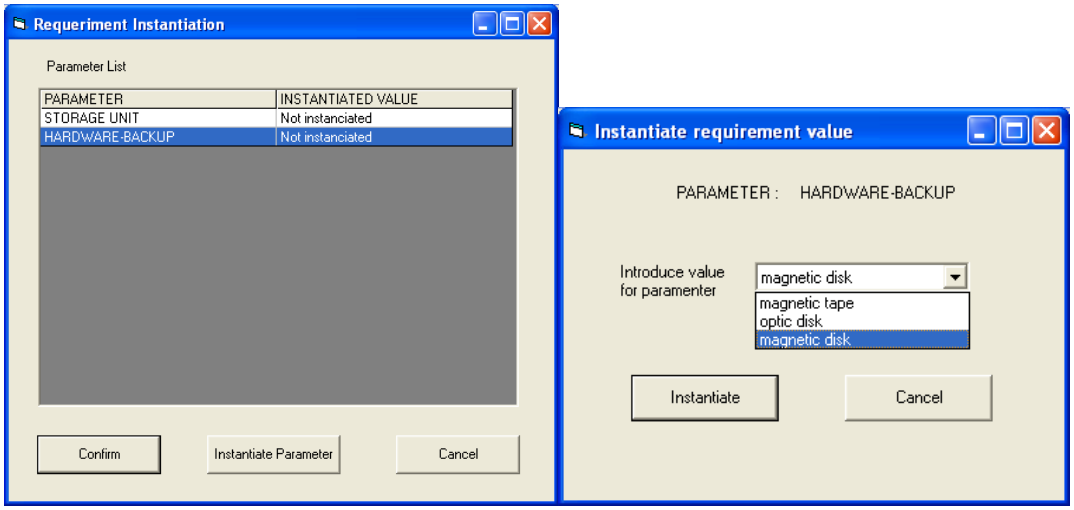

Fig. 5. Interface for requirement instantiation.

In the running example, the requirement selected in Fig.4 has two parameterized parts (left side of Fig.5). After selecting the `hardware-backup` parameter and clicking the "Instantiate Parameter" button, another interface (right side of Fig.5) will show the predefined values for that parameter. In this case the value of "optic disk" was selected by the requirements engineer for the hospital management tool for the `hardware-backup` parameter and the value of "CD-ROM" for the `storage unit` parameter. After the instantiation, the new requirement is shown in the field labeled "New text".

K7 Repository improvement

The experience gained through the project development will be useful to improve the quality of the requirements in the repository. Thus, a user *administrator* should receive the improvement suggested by a requirements engineer, and decide if the change should be added to the repository. At the moment, the requirements engineer has to send the request for change to the administrator by e-mail.

K8 CARE support to reuse

Not only is a CARE tool for managing requirements needed but it has to be one which supports reuse. Therefore, RequisitePro has been extended, overcoming its limitations with regard to the eight key issues concerning reuse identified in Section 3. Nevertheles, SirenTool is just one example of how the key issues could be supported by a CARE tool. With a similar adaptation, other tools like those considered in Section 4 could have been used for this purpose.

6 Related requirements reuse approaches

Our paper follows the line of Lam et al. (Lam, MacDermid et al. 1997) where they present some criteria towards systematic requirements reuse. We agree with Lam et al. in the usefulness of parameterized requirements (key issue K6), by formalizing the set of parameters values when possible, and the necessity of reuse products organization (key issue K1). Our key issues extend the ten criteria proposed in (Lam, MacDermid et al. 1997) with technical issues needed to manage reuse such as: i) inclusive trace relationships to specify the set of requirements included in a requirement pattern; ii) exclusive trace relationships for prevention of reuse misuse; and iii) catalogs structuring the repository and acting as a requirements focal points.

The idea of reusable parameterized templates, presented by the authors in (Toval, Nicolás et al. 2002; Toval, Olmos et al. 2002) (key issue K6), is also used by (Firesmith 2004) to specify security requirements. This author collaborates in the definition of a reuse-based approach focused on the security field (Sindre, Firesmith et al. 2003), which is quite similar to that described in (Toval, Nicolás et al. 2002), where the particularization of the SIREN method for the security field is explained.

Other research exists which is closer to technical solutions for reuse and which consists of translating the concept of patterns to the early stages of the development process. Patterns have been widely applied —for instance in design (Gamma, Helm et al. 1995)— to take advantage of reuse benefits. This reuse technique was adopted by the requirements engineering community which started to write about "requirements patterns" (Robertson 1996) (Konrad and Cheng 2002), also called "analysis patterns" by the patterns community (Fowler 1997). A comparison between requirements patterns and analysis patterns can be found in (Pantoquilho, Raminhos et al. 2003). Regardless of the name of the patterns, the important thing is that they use a template (with different attributes depending on the approaches) to specify a related set of requirements that could be used as the starting point for any similar system. Consequently, the templates have to be cataloged and stored in a repository for consultation in further developments.

These proposals could be taken into account in any reuse-based RE process when organizing the requirements in the repository (key issue K1). Usually a pattern corresponds to a use case specification, so a requirement pattern would group all the functional and non-functional requirements related to the use case.

(Alexander and Kiedaisch 2002) suggest using use cases to solve the problem of recycling requirements[1] and they have extended the DOORS CARE tool to support their approach (Scenario Plus Toolkit). The approach relies on accurate traceability to link use cases to requirements. Consequently, key issues K1, K2, K3, K7 and K8 are covered by the approach. In our view, uses cases are not necessarily the most suitable reuse unit, since they represent a slanted requirements specification.

The importance of defining a traceability model to face requirements reuse is also pointed out by (Knethen, Paech et al. 2002). In this case, the traceability model proposed is used at reuse time to infer the recycling candidates, so it differs from the one proposed in key issue K5 in the entities linked by the trace relationships. In K5, the traceability model links requirements statements whereas the proposal of Knethen et al. links *logical entities* that are represented by each *documentation entity*. One example showing the difference between logical and documentation entities could be: a paragraph of an "Overview Description", that is a documentation entity, represents a function "seat control", that is a logical entity. Nevertheless, Knethen et al distinguish three types of relationships:

- *Refinement relationships* are relationships between logical entities on different levels of abstraction.
- *Dependency relationships* are relationships between logical entities on the same level of abstraction.
- *Representation relationships* between two documentation entities that represent the same logical entity

[1] These authors prefer to talk about "recycling" instead of "reuse" since they consider that the high effort in making components reusable and the need for sophisticated retrieval mechanisms used by other approaches such as software reuse are unnecessary for requirements.

The first two relationships have their counterpart in the traceability model proposed in K5, since a document hierarchy with different levels of abstraction is defined and relationships linking requirements from different documents are allowed (Refinement relationship) as are linking requirements in the same document (Dependency relationship). The traceability model defined in K5 goes further and distinguishes three types of dependency relationships: parent-child, inclusive and exclusive relationships. On the other hand, it would be interesting to take into account the representation relationship to control inconsistencies at reuse time.

Finally, and related to key issue K2, there are some approaches that point to analogical reasoning engines as a flexible mechanism to retrieve and adapt past specifications and to support requirements engineering in generating specifications (Maiden and Sutcliffe 1996) (Pisan 2000). Taking into account key issue K1 (requirements organization) and K5 (traceability reuse), it is also possible to generate a new requirements specification instead of inferring requirements by analogy.

7 Conclusions

To improve the RE process, reuse must be taken into account. In this paper, eight issues regarding requirements reuse have been identified as key features in achieving an effective and practical reuse-based RE process. While some of these features involve new concepts (such as the need to define a common set of attributes), others consist of the explicit adaptation of existing concepts to the requirements reuse realm (parameterized requirements, exclusive traceability relationship, traceability relationships reuse, or reuse granularity). These issues are deliberately kept simple so that a software development organization with an immature requirements management process can afford to put them into practice within a short time.

To manage requirements reuse properly a CARE tool supporting most of the key issues described should be used by the requirements engineer. Besides the identification of these key issues for requirements reuse, an analysis of some of the most popular commercial CARE tools has revealed their current limitations in supporting issues related to requirements reuse. Therefore, it was decided to extend one of them, RequisitePro, which led us to a huge saving in implementation effort. As a result, a new software solution, named SirenTool, has been built illustrating how supporting the key issues described is feasible with a reasonable programming effort. Note that the work carried out is applicable to CALIBER-RM and DOORS, since both provide an extensibility interface.

The eight key issues proposed have been validated in the context of a real case study regarding clinical history management in the intensive care unit of the Hospital of Getafe in Madrid. The main lessons learned include the following:

- The reuse of the subsets obtained from different search criteria is preferred to the reuse of a full base document which contains all the requirements of a specific domain. Nevertheless, some “expert knowledge” should be provided to assist in the use of the catalog (for example, in search criteria definition).
- The specification of parameterized requirements is fundamental for requirements reuse, but the dependence relationships between the parameters should be managed. For instance, going back to Fig.5, the selection of a specific value of hardware backup determines the value of the storage unit parameter. In this case, if we chose “optic disk”, only “CD-ROM” or “DVD” could be chosen for the “storage unit” parameter.

- The validation of the key issues confirms the need of automating them, since none of the commercial tools provides reuse capabilities. SirenTool shows how the automatization of the key issues could be addressed.

8 Acknowledgements

The research reported in this paper has been supported by the Spanish Ministry of Science and Technology, sub-project TIC2003-07804-C05-05 PRESSURE (*Precise Software Models and Requirements Reuse)*. This is part of the DYNAMICA (*DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures*) project involving five Spanish universities and ten TIC companies. DYNAMICA will develop a variety of models, languages and CASE tools for the construction of architectural models of the partner companies, while PRESSURE will incorporate Requirements Engineering methods and tools for the artifacts developed under DYNAMICA. Part of the research project has also been funded by ERFD (*European Regional Development Fund*).

We would also like to thank Manuel Campos for his collaboration in the validation process of the key issues presented in this paper. He is member of the development team in the project "Technological transfer of a patient management system for the intensive care unit" which is under contract between Murcia University and the Foundation for Biomedical Research in the Getafe Hospital (Madrid) in the period 2006-2007.

9 References

Alexander, I. (2003). Requirements Engineering Tool Vendors and Freeware Suppliers.http://easyweb.easynet.co.uk/~iany/other/vendors.htm

Alexander, I. and F. Kiedaisch (2002). Towards Recyclable System Requirements. Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden

Caliber-RM (2005). http://www.borland.com/us/products/caliber/index.html, Borland.

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

Doors (2003). http://www.telelogic.com/doors, Quality Systems & Software.

EU (1995). Directive 95/46/CE of the European Parliament and Council, dated October 24th, on people protection regarding personal data management and the free circulation of these data. DOCE no. L281, 23/11/1995,

Favaro, J. (2002). "Managing Requirements for Business Value." IEEE Software **9**(2): 15-17

Firesmith, D. (2003). "Modern Requirements Specification." Journal of Object Technology **2**(2): 53-64.http://www.jot.fm/issues/issue_2003_03/column6

Firesmith, D. G. (2004). "Specifying Reusable Security Requirements." Journal of Object Technology **3**(1): 61-75

Fowler, M. (1997). Analysis Patterns-Reusable Objects Models, Addison Wesley.

Gamma, E., R. Helm, et al. (1995). Design Patterns: Elements of Reusable Object-Oriented Software Addison-Wesley Professional Computing Series.

Glass, R. L. (1998). Software Runaways: Monumental Disasters, Prentice Hall.

Glass, R. L. (2002). Software Engineering: Facts and Fallacies, Addison-Wesley.

IEEE (1999a). Std 830-1998 Guide to Software Requirements Specifications (ANSI). Volume 4: Resource and Technique Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection.

IEEE (1999b). Std 1233-1998 Guide for Developing System Requirements Specifications. Volume 1: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection.

INCOSE (2006). Requirements Management Tools Survey. . International Council on Systems Engineering.http://www.incose.org/ProductsPubs/products/toolsdatabase.aspx

Kaindl, H., S. Brinkkemper, et al. (2002). "Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda." Requirements Engineering **7**(3): 113-123

Kang, K., S. Cohen, et al. (1990). "A. Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-021, ADA235785)." Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University

Kang, K., S. Kim, et al. (1998). "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures." Annals of Software Engineering **5**(5): 143-168.

Knethen, A. v., B. Paech, et al. (2002). Systematic Requirements Recycling throug Abstraction and Traceability. IEEE Joint International Conference on Requirements Engineering (RE'02), University of Essen. Germany

Konrad, S. and B. H. C. Cheng (2002). Requirements Patterns for Embedded Systems. IEEE Joint International Conference on Requirements Engineering (RE'02), University of Essen. Germany

Kotonya, G. and I. Sommerville (1998). Requirements Engineering. Processes and Techniques, John Wiley & Sons.

Lam, W., J. A. MacDermid, et al. (1997). Ten StepsTowards Systematic Requirements Reuse. 3rd IEEE International Symposium on Requirements Engineering (RE'97), Annapolis, MD

Maiden, N. A. and A. G. Sutcliffe (1996). "Analogical Retrieval in reuse-oriented requirements engineering." Software Engineering Journal **11**(5): 281-292

Martinez, M. A., J. Lasheras, et al. (2006). An audit Method of Personal Data based on Requirements Engineering. IV International Workshop on Security In Information Systems (WOSIS 2006), Chipre

Matulevicius, R. (2005). Process Support for Requirements Engineering. A Requirements Engineering Tool Evaluation Approach. Department of Computer and Information Science. Faculty of Information Technology, Mathematics and Electrical Engineering. Trondheim, Norwegian University of Science and Technology.

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

Neill, C. J. and P. Laplante (2003). "Requirements Engineering: The State of the Practice." IEEE Software. 2003 **20**(6)

Nicolás, J., J. Lasheras, et al. (2006). A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance. Learning Software Organizations + Requirements Engineering (LSO+RE 2006), Hannover. Germany

Pantoquilho, M., R. Raminhos, et al. (2003). Analysis Patterns Specifications: Filling the Gaps. The Second Nordic Conference on Pattern Languages of Programs, ViKingPLoP 2003, Bergen. Norway

Pisan, Y. (2000). Extending Requirements Specifications Using Analogy. ICSE 2000, Limerick, Ireland

Prieto-Díaz, R. (1993). "Status Report: Software Reusability." IEEE Software **10**(3): 61-66

Ramesh, B. and M. Jarke (2001). "Toward Reference Models for Requirements Traceability." IEEE Transactions on Software Engineering **27**(1): 58-93

Reifer, D. J. (2003). "Is the Software Engineering State of the Practice Getting Closer to the State of the Art?." IEEE Software 2003. **20**(6)

RequisitePro®, I. R. (2006). http://www-306.ibm.com/software/awdtools/reqpro/, Rational Software.

Rine, D. C. and N. Nada (2000). "An empirical study of a software reuse reference model." Information and Software Technology **42**(1): 47-65

Robertson, S. (1996). "Requirements Patterns via Events/Use Cases." http://www.systemsguild.com/GuildSite/SQR/Requirements_Patterns.html

Robertson, S. and J. Robertson (2006). Mastering the Requirements Process (2nd edition), Addison-Wesley.

Rothenberger, M. A., K. J. Dooley, et al. (2003). "Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices." IEEE Trans. on Soft. Eng. **29**(9): 825-837

Sindre, G., D. G. Firesmith, et al. (2003). A Reused-based Approach to Determining Security Requirements. The Ninth International Workshop on Requirements Engineering: Foundation for Software Quality - REFSQ'03

Smith, J. M. (2001). Troubled IT Projects prevention and turnaround. London, IEEE.

Sommerville, I. (2004). Software Engineering (7th edition), Addison-Wesley.

Sommerville, I. and J. Ransom (2005). "An Empirical Study of Industrial Requirements Engineering Process Assesment and Improvement." ACM Transactions on Software Engineering and Methodology **14**(1): 85-117

Toval, A., J. Nicolás, et al. (2002). "Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach." Requirements Engineering Journal. Springer **6**(4): 205-219

Toval, A., A. Olmos, et al. (2002). Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection. IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02), Essen, Alemania

Volere (2006). VOLERE, Requirements Tools. http://www.volere.co.uk/tools.htm

Wiegers, K. E. (1998). "Read My Lips: No New Models!." IEEE Software **15**(5): 10-13