Requirements Eng (2013) 18:43–66
DOI 10.1007/s00766-011-0139-7

ORIGINAL ARTICLE

Requirements engineering within a large-scale security-oriented research project: lessons learned

Seda Gürses · Magali Seguran · Nicola Zannone

Received: 18 October 2010 / Accepted: 19 October 2011 / Published online: 6 November 2011

Abstract Requirements engineering has been recognized as a fundamental phase of the software engineering process. Nevertheless, the elicitation and analysis of requirements are often left aside in favor of architecture-driven software development. This tendency, however, can lead to issues that may affect the success of a project. This paper presents our experience gained in the elicitation and analysis of requirements in a large-scale security-oriented European research project, which was originally conceived as an architecture-driven project. In particular, we illustrate the challenges that can be faced in large-scale research projects and consider the applicability of existing best practices and off-the-shelf methodologies with respect to the needs of such projects. We then discuss how those practices and methods can be integrated into the requirements engineering process and possibly improved to address the identified challenges. Finally, we summarize the lessons learned from our experience and the benefits that a proper requirements analysis can bring to a project.

Keywords Requirements engineering practice · Large-scale research project · Cooperative work · Gap analysis · Interaction analysis

S. Gürses
ESAT/COSIC and IBBT, K. U. Leuven, Leuven, Belgium
e-mail: sguerses@esat.kuleuven.be

M. Seguran
SAP Research, Sophia Antipolis, France
e-mail: magali.seguran@sap.com

N. Zannone (✉)
Eindhoven University of Technology, Eindhoven, Netherlands
e-mail: n.zannone@tue.nl

1 Introduction

In the last decades, it has become common for ICT researchers to work in large-scale research projects. These projects are often carried out by consortia that involve different types of partners (e.g., universities, multinational corporations, small and medium enterprises) from various countries. Large-scale security-oriented research projects (the focus of this paper) are an example of such projects.

Security-oriented research projects are expected to have a well-defined set of objectives. Ideally, the requirements of such software systems-to-be are defined when the project is established. The elicitation and analysis of requirements are performed in the context of requirements engineering (RE). RE is the process of identifying system stakeholders and their needs, defining constraints on the software system, and documenting these in a form that is suitable for analysis, communication and subsequent implementation of the system [50, 69]. This phase of the software and system development process is widely recognized as being fundamental to the success of an ICT project [30].

However, in practice, RE activities are often not carried out properly. This is more likely the case when non-functional concerns like the security of the system-to-be have to be addressed. For example, a typical approach to addressing security concerns within a system is to identify security requirements after the design of the system is completed. Moreover, security concerns are often addressed only at the technical level (as opposed to the organizational level). Such an add-on security approach is, however, unlikely to be effective. Security mechanisms are fitted into a pre-existing design, which may lead to conflicts between security and functional requirements of the system. Consequently, additional vulnerabilities may be introduced if

security mechanisms are blindly inserted into a security-critical system. Recent studies show that the situation can be improved if security aspects are taken into account throughout the whole system development process (hence, also during the analysis of the organizational setting in which the system will operate) [24].

It is rarely the case that when the project consortium of research projects are established, partners whose expertise and activities are focused on RE are selected. Consequently, most consortia do not include RE experts. As a result of the absence of such expertise, the popularity of methods for rapid software development (e.g., architecture-driven approaches [22]) and the convenience of such methods for the deployment of immediate research and implementation results, requirements elicitation and analysis are often neglected. Such an approach however can have harmful consequences. The success of research projects requires establishing an integrated common vision among the project partners, adopting a consistent working approach and creating the willingness to share research and business expertise despite the competitive context of a project consortium [30]. In a project in which each partner only focuses on the development of its own solutions, it is difficult to identify and solve conflicting requirements among partners. Consequently, the integration of individual results into a single system may simply fail.

We argue that performing RE activities properly helps to mitigate the issues presented above. However, many factors play a role in the successful completion of RE activities in large-scale research projects. These factors include the background, interests and expertise of the individuals involved, as well as the type, objectives and geographical distribution of the partner organizations in the project [13, 14, 38, 57]. For example, for researchers, projects often offer a means to validate research results, whereas for small and medium enterprises, the main focus is on the development of software products. Such differences in objectives may manifest themselves in conflicting requirements.

Therefore, when we deal with RE activities in large-scale research projects, a number of challenges have to be faced. These can be summarized as:

- Establishing a common understanding of what requirements are.
- Defining a process to elicit, elaborate and validate the requirements with numerous and geographically distributed partners.
- Identifying the required innovation for the project by distinguishing requirements that cannot be fulfilled using existing research and solutions.

These challenges also function as meta-requirements for the process of selecting the appropriate RE methodologies and activities to be used in large-scale security-oriented research projects. Researchers have proposed a number of methodologies and techniques to perform the different RE activities (e.g., [7, 34, 52, 60]). However, there is no single RE methodology that covers all of the RE activities and that address all the challenges to research projects we listed above. For instance, no existing RE methodologies support project partners in identifying the required innovation for the project. Further, RE methodologies are usually designed for the industry and consequently are centered on the customers' needs [56]. Therefore, they may not be suited for research projects in which researchers and developers are the main stakeholders. Finally, the analysis of security requirements demands the inclusion of specialized methodologies (see [19] for a survey); each of them however focuses on particular security aspects. The selection of the appropriate methodology depends on the needs of the project.

The objective of this paper is to describe the approaches that can be used to support a project consortium during the RE process. This description includes an evaluation of the advantages and disadvantages of applying these approaches in a large-scale research oriented project. In particular, the paper analyzes and discusses the challenges faced in the context of the TAS^3 project (http://www.tas3.eu), the RE process adopted to mitigate these challenges and an evaluation of the different RE activities.

TAS^3 is an EU-integrated project focusing on the main security and privacy issues in distributed systems aiming to deploy a generic architecture for managing employability and healthcare personal information services. This project was initially conceived as an "architecture-driven project". As a result, the project faced a number of issues independent from its objectives, which were pointed out during a critical review of the project. Most of these issues can be traced back to the lack of a proper RE approach. After the critical review, the project consortium decided to set up an RE team whose task was to coordinate and support partners in the execution of RE activities. The team, which includes the authors of this paper, consisted of researchers who have previous experience with RE. This paper discusses the experience gained by the RE team.

The paper is structured as follows. Section 2 discusses the challenges of applying RE methodologies to large-scale research-oriented projects. Section 3 presents the RE methodologies and processes applied in the TAS^3 project. In particular, it investigates alternative proposals from researchers as well as industry best practices, for each RE activity. It demonstrates how existing proposals can be adopted, integrated and extended in order to support the RE process in a large-scale research project. Section 4 reports the outcome of the application of these various methodologies and processes within the TAS^3 project. Section 5 presents guidelines for performing RE activities in large-

scale research projects as we derive them from lessons learned. Finally, Sect. 6 discusses related work, and Sect. 7 concludes the paper.

2 Challenges

The TAS^3 project is an EU project that focuses on the main security and privacy issues in an ecosystem of distributed service and identity providers, ranging from authentication and trust management to data protection. Specifically, the goal is to deploy a next generation trust and security architecture and adaptive security services that preserve privacy and confidentiality of individual users, i.e., identity management, in dynamic service environments. The architecture is expected to be general enough to apply to different contexts and comply with data protection legislation (e.g., EU Directive 95/46/EC) and the NESSI reference architecture (http://www.nexof-ra.eu). The TAS^3 consortium is composed of 18 industrial and academic partners.

The project faced many issues in the first year of its lifetime. TAS^3 is an integrated project with the vision of implementing an architecture. Such an architecture-driven approach had its pros and cons. The pros for security and architecture experts lied in the fact that they were able to start discussing technical details without further specification of the project. The cons became apparent due to the delayed kick-off of the project and the resulting time shortage. Below we discuss the main challenges we encountered in the course of the TAS^3 project. These challenges can be generalized to any large-scale research project.

Challenge 1 (Project planning) *How can a research project have an integrated common vision and a consistent working approach?*

Project consortia usually have a Description of Work (DoW) that describes the activities to be performed within the projects, organized in various workpackages. The quality and precision of the DoW and its interpretation by project partners at execution time may require extra alignment and specification of assignments. This re-alignment may not be trivial. In the TAS^3 DoW, the software engineering assignments were under-specified, making it difficult to understand and perform these assignments. For example, the DoW neither defines the objectives of the requirements analysis activities precisely, nor does it prescribe the activities for their achievement. Most importantly, it does not describe the necessary interactions among the workpackages.

The underspecification of objectives, activities and workpackage interactions led to two issues that were critical for the continuation of the project: (1) the architecture was designed independently from requirements analysis, and (2) each partner pursued his/her own understanding of the functionalities to be provided by the architecture. This resulted in inconsistencies in the overall architecture and in requirements that were neglected in the design of the architecture. In particular, requirements coming from the end-users (i.e., from pilot scenarios) were not taken into account as the workpackage responsible for defining the pilot scenarios was not directly involved in the design of the architecture.

Once these problems became evident and they were underlined during a critical project review, a reorganization of the RE activities was planned by a new RE team. Performing these activities, however, raised additional challenges.

Challenge 2 (Requirements definition) *What are requirements?*

Although for an expert the definition of what a requirement is can be trivial, agreeing on a shared definition among project partners may not be instantaneous. Different partners may have different interpretations of what requirements are. For instance, in TAS^3 most partners were unable to distinguish between requirements and design solutions, or between functional and non-functional requirements. This was mainly evident during the elicitation of security requirements: what was seen by some partners as a security requirement was interpreted by others as a security solution.

These differences in perspectives originated from a number of factors. First, the partners were lacking a common understanding of software engineering and requirements due to their diverse backgrounds. Especially for researchers, the role of RE activities in a large project was not self-evident. Second, due to the vision of an architecture-driven project, the classical flow of engineering projects was ignored. Third, the project mainly focused on the development of security and privacy functionality. This blurred the traditional distinction between functional and non-functional requirements.

Additional difficulties were caused by the heterogeneous nature of the requirements (i.e., technical, legal and usability as well as research oriented requirements had to be considered). Moreover, requirements for both the architecture and the pilot scenarios had to be elicited. The pilots are embedded in three countries in two different domains (i.e., healthcare and employability). The integration of these different concerns on the same level of abstraction can be arduous and needs to be addressed during requirements engineering.

Challenge 3 (Requirements elicitation) *How can we elicit heterogeneous requirements at a comparable granularity?*

A number of elicitation techniques have been proposed in the literature [50]. These include *traditional techniques* (e.g., questionnaires, surveys, interviews), *group elicitation techniques* (e.g., brainstorming and focus groups), *prototyping*, *model-driven techniques* and *cognitive techniques*. However, not all of these techniques are suitable for large-scale research projects. For instance, group elicitation techniques are difficult to apply when partners are geographically distributed. In addition, the selected requirements elicitation technique may contain a well-defined schema for the specification of requirements that may not be suited for the different types of requirements of a large research project. The challenge for the RE team is to determine the most appropriate techniques given the circumstances and the heterogeneity of the requirements of the project.

The elicitation of security and privacy requirements presents additional issues. Many efforts have been spent in the last years to extend requirements elicitation techniques to also include security and privacy requirements [19]. However, those techniques often focus on specific security aspects such as design of secure components [36], system vulnerabilities [18], security issues in social dependencies among stakeholders [28, 41], and their trust relationships [44], attacker behavior [63] and attacker goals [40], as well as events that can cause system failures [3]. Adopting only one of these frameworks would emphasize certain types of security requirements, while other security requirements essential for the project may not be captured. The alternative of adopting all these methods is clearly impossible.

The selection of an appropriate granularity for the elicited requirements is also critical to bringing all partners to a common understanding of the main engineering problem. In TAS^3, some workpackages provided dozens of requirements, while others presented only a handful. This discrepancy was due to a number of factors including the heterogeneous nature of requirements and the efforts partners were willing to put into elicitation activities. The RE process needs to account for these discrepancies and include steps to achieve a comparable granularity given the heterogeneity of both the requirements and the project partners' approach to requirements.

Challenge 4 (Gap analysis) *How can we identify the innovation needed while providing a common mission for the project?*

One of the objectives of the RE activities as described in the DoW of TAS^3 is to identify requirements regarding unsolved problems in the field of security and trust in service-oriented open and distributed environments. Specifically, the goal is to identify those elicited requirements that can be translated into research and development activities to be carried out in the project. The process of distinguishing requirements that demand further research is comparable to a *gap analysis* study as it is common in business and economics [33].

Executing a gap analysis, however, is challenging. First, in a large research project with technical, legal and domain-specific research needs, the scope may be difficult to determine. For instance, in software development, gap analysis can be used to document which functionalities have been accidentally left out, which ones have been deliberately eliminated and which ones still need to be developed. From a legal perspective, it can be used to establish what additional legal requirements apply given the planned functionality of the system. Second, there are no methodologies for gap analysis in the RE mainstream. Hence, the RE team is responsible for developing a gap analysis method that is appropriately scoped to the project's objectives.

Challenge 5 (Requirements communication and agreement) *How can we communicate heterogeneous requirements in such a way that partners can understand and agree on each others' requirements?*

Requirements should be represented in a form that is suitable for analysis, communication and subsequent implementation [50]. Different partners may have similar or conflicting requirements or may depend on each other for the achievement of their requirements. Determining conflicts and/or dependencies calls for additional efforts to identify and analyze such *interactions* among requirements elicited by different partners [61]. The representation of the requirements should leverage such analysis.

However, in research projects, partners may have different background and expertise. Consequently, each partner may prefer a different framework (e.g., UML [52], Tropos [7], Problem Frames [34]) for representing requirements. Letting partners use their own framework to specify requirements, however, is not appropriate because it makes the integration of requirements and interaction analysis difficult. First of all, partners have to understand the requirements elicited by other partners in order to assess the interactions between those requirements and the requirements they have elicited. Further, each framework may use different concepts to represent requirements, making it difficult to compare the requirements specified by different partners. Addressing these differences by imposing a single framework to all partners can also have its disadvantages, emphasizing certain aspects over others.

Moreover, each partner has his specific goals and interests within the project, which in most cases are not related to RE. As a result, partners may not be willing to spend a great effort in RE activities. Hence, introducing frameworks previously unknown to the partners, e.g., through training sessions, may intensify resistance to

participation in RE activities. Such resistance may be further amplified when the partners are distributed, as in the case of the TAS^3 project.

Challenge 6 (Business conflicts) *How can we solve conflicts between corporate/business and research interests?*

Project deliverables are public documents. From the perspective of industry partners, this may mean that their innovation becomes accessible to competitors. Moreover, industry partners may not feel comfortable in publicly reviewing the technology developed by their competitors and comparing it with their own. Such analysis could reveal an expert opinion about the weaknesses and strengths of competing technologies and may lead to economic disadvantages or conflicts.

In the course of the TAS^3 project, some partners expressed such anxieties and, as a result, this effected their willingness to provide requirements. These partners were often representatives of small-medium enterprises that worry about the disclosure of information about their technology. Overcoming such challenges is critical for the success of any project where the consortium consists of partners from both industry and academia. However, we will not consider this matter further as this is an issue of intellectual property rights that should be resolved at a much earlier stage, i.e., when the consortium agreement is signed.

3 Approach

This section presents the RE process we followed in the course of the TAS^3 project. The steps we took are aligned with the DoW of the project. This alignment was challenging since the DoW itself introduced complications that we first had to untangle. In particular, requirements elicitation has been described under the heading "Design Requirements" [48].[1] The objective of this first deliverable is defined in the DoW as *"modeling the legal framework and regulatory compliance requirements; collecting and defining the application domain and user requirements from test beds; and, defining the system requirements for all* TAS^3 *components according to software engineering specification standards."* A further related deliverable is also conceived, titled "Requirements Assessment Report" [29]. The main objective of this deliverable is defined in the DoW as *"gathering requirements about unsolved problems in the field of security and trust in service-oriented open and distributed environments."* No clarification was provided with respect to how these two deliverables should be distinguished and how the activities were interdependent.

To manage this ambiguity, we divided our efforts into the following activities (Fig. 1):

1. *Requirements elicitation*, which aims to gain knowledge about the needs of the project and the environment of the software system.
2. *Requirements elaboration*, which aims to elaborate the elicited requirements to the research and development objectives of each workpackage (viewpoints).
3. *Gap analysis*, which aims to identify the research needs of the project by comparing the objectives of the project with the security and privacy solutions available on the market.
4. *Interaction analysis*, which aims to provide a common mission of the project by identifying dependencies and conflicting requirements among workpackages, consolidating the viewpoints and mapping the requirements to architecture components.
5. *Evaluation*, which aims to evaluate the overall project including RE activities.

The planned RE process is iterative. First, functional, security and privacy requirements for the software system to be developed and the pilot scenarios are elicited and elaborated. The elaborated requirements are compared with an overview of the state-of-the-art in research and business solutions for trust and security in service-oriented systems. The results are used to complete a gap analysis to identify the research activities to be performed in the course of the project. At the same time, the interactions among intra- and inter-workpackage requirements are analyzed in order to identify inconsistencies among requirements elicited in the context of the different workpackages. Finally, the whole project (including RE activities) is assessed using evaluation criteria. Ideally, the process is reiterated until all evaluation criteria are satisfied.

There is not a single RE methodology that covers all of these activities and addresses the challenges discussed in Sect. 2. Therefore, we have investigated and incorporated alternative research approaches and industry best practices for each activity where appropriate. In the remainder of this section, we describe those approaches and how they have

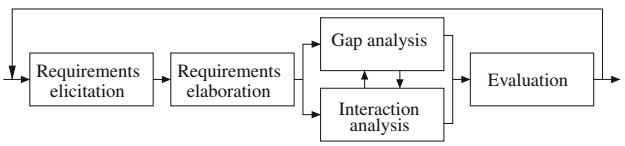

Fig. 1 Requirements engineering process

[1] Notice that the title may bring some confusion as the word "design" can lead to a restriction of the scope of requirements elicitation only to the architecture.

been integrated to provide a unified framework to support the RE activities in the TAS^3 project.

3.1 Requirements elicitation

The aim of the requirements elicitation phase is to capture the problems that are to be solved with the system-to-be. Different techniques have been proposed in literature for eliciting requirements from stakeholders [11, 16, 25, 62]: interviews, questionnaires, user observation, workshops, brain storming, use cases and prototyping. Our approach in TAS^3 is mainly based on interviews and use cases. In particular, requirements are elicited on the basis of pilot scenarios. The interviews are completed during interactions with the stakeholders (i.e., pilot developers and other project partners).

The objective of elicitation through pilot scenarios is to concretely identify a system's future uses. More specifically, a pilot scenario is a description of one or more interactions involving the software system to be developed and its environment. Pilot scenarios are defined as follows:

- *Identification of the scenario context:* This activity consists of describing the concepts of the application domain and identifying the main features to be demonstrated in the scenario. Different features of an application domain may be relevant for a project. In order to capture these, more than one scenario can be defined for every considered application domain.
- *Identification of the actors and their respective tasks:* This activity consists of identifying the main actors of each scenario along with the roles they play, the tasks that they have to perform and their assets.
- *Identification of security threats:* Actors' assets can be the target of attackers. Attackers can be internal or external entities of the system. They are assumed to be able to perform malicious actions, which attempt to break the security of (a component of) the system. The aim of security threat identification is to identify the internal and external threats (i.e., attacks) and determine their impact on system security [18, 41, 63]. This information is then used as a starting point for the security risk analysis.

Although scenarios are useful for eliciting requirements, they do not necessarily provide the requirements of the system-to-be [55]: they usually describe the system's behavior in specific situations; on the contrary, requirements describe what the system should do in general. Accordingly, scenarios need to be analyzed in order to capture and refine the general requirements of the system. The following are the appropriate steps for such an analysis:

- *Capturing requirements:* This activity aims to elicit requirements from scenario descriptions. As this activity requires communication with stakeholders, particular attention has to be paid on how to specify requirements. Based on classical RE approaches [31, 66], detailed guidelines can be defined for the specification of requirements. These guidelines should include the definition of a controlled vocabulary for formulating requirements specifications (e.g., *shall* and *must* shall be used for the specification of mandatory requirements, *should* for the specification of optional requirements, *notice* for additional explanation of requirements) as well as instructions on how to specify proper requirements (e.g., requirements shall describe problems instead of solutions, amalgamated requirements shall be disjoined).
- *Analyzing and refining requirements:* This activity aims to refine the elicited requirements, removing ambiguity in the specifications and detecting under-specified requirements. A number of methodologies have been proposed to assist system designers during this phase [7, 15, 34]. Given the security nature of the project, we have adopted Secure Tropos [44] for the analysis and refinement of security requirements. This methodology uses the concepts of ownership, permission, delegation and trust to analyze and refine (security) dependencies among actors involved in the system (including the system itself). However, due to its static nature, the analysis of temporal aspects is not possible. To overcome this limitation, UML sequence diagrams can be used to analyze the interactions among scenario actors and identify sequences of activities that may lead to security breaches.

3.2 Requirements elaboration

Once the initial set of requirements are elicited, the next step of the RE process is to elaborate and refine those requirements in order to enable subsequent analysis. The challenges in elaborating requirements in large research projects lie in the number of requirements to manage, the differences in the focus of the various workpackages, as well as the discrepancy in the expertise and interests of the system designers.

One solution to the complexity of managing requirements in large projects is to organize the requirements elaboration process by viewpoints [51, 64]. Each viewpoint aims to elaborate different aspects or concerns from the perspective of different stakeholders [64]. In a project, viewpoints can be used, for instance, to group the requirements by workpackages. Once viewpoints are defined, they should be analyzed for overlaps and conflicts and integrated

into a single requirements document. This integration activity is carried out during interaction analysis (Sect. 3.4).

While breaking down a monolithic requirements document facilitates the management of numerous and heterogeneous requirements, it is important to keep an overview on the overall objectives of the system-to-be. Therefore, viewpoints analysis has to be complemented with the analysis of global requirements that elaborate the problems that the project intends to solve.

One way of representing the viewpoints and global requirements and making them accessible to all stakeholders is to deploy a systematic and comparable notation for documenting the objectives of a viewpoint and elaborating the requirements that have to be fulfilled in order to achieve those objectives. To this end, we decided to use standardized templates. In particular, the templates used in TAS[3] are based on two methodologies for template-based requirements elicitation: Volere [60] and the template given in [65].

However, standardized templates have to be customized to meet the specifics of the project. In the case of TAS[3] from Volere, we employ the elements for assessing the scope of work for each viewpoint: the documentation of a viewpoint's objectives and the identification of the open problems addressed by the viewpoint. Further, the template in [65] defines the following mandatory fields: requirement id, version, author, source, purpose, requirement description, time interval, importance, urgency, comments. Consequently, our viewpoint template employs *reqID*, which is used to uniquely identify the requirements and to indicate the viewpoint from which the requirement is originating (i.e., the 'source'); *justification* (instead of 'purpose', as it better conditions system designers to state why the requirement is necessary); *requirement* (instead of 'requirement description' for brevity). Further fields are addressed through the versioning of the requirements document itself ('version') and the list of contributors ('author'). A sample requirement using the viewpoint template is presented in Table 1.

The template also includes a field called *interaction* that is used to indicate the interactions of a given requirement with other requirements. This information is later used to perform interaction analysis (Sect. 3.4). Interactions are specified using a controlled vocabulary to limit ambiguity:

- *A depends on B*: the fulfillment of requirement *A* requires the fulfillment of requirement *B*, i.e., *B* is a condition for *A*.
- *A supports B*: the fulfillment of requirement *A* is needed to fulfill requirement *B*, i.e., *A* is a condition for *B*.
- *A implements B*: requirement *A* is a specialization of requirement *B*.
- *A abstracts B*: requirement *A* is a generalization of requirement *B*.
- *A is in conflict with B*: requirement *A* and requirement *B* are logically inconsistent or the implementation of both requirements is not feasible.
- *A is similar to B*: requirement *A* and requirement *B* refer to the same problem or their implementation overlap. The relationship between the requirements is assumed not to be one of the other types of interactions.

Notice that *supports* and *abstracts* are the opposites of *depends* and *implements*, respectively. Although providing constructs for representing a relation and its opposite may seem redundant, it makes it possible to capture interactions that might not be seen as bidirectional.

Ideally, the elaboration of requirements should also include other aspects fundamental for the characterization of the requirements problem, such as domain assumptions [70] and the traditional partitioning of requirements into functional and non-functional requirements. Their absence is due to the necessity to decrease the complexity of elaboration activities. A discussion on this issue is presented in Sect. 5.

3.3 Gap analysis

Gap analysis is a process for identifying the delta between the current situation and the future desired situation in a given domain [10]. In a research project, the scope of gap analysis can be defined as the identification and documentation of the research and development activities to be performed within the project. In particular, gap analysis may aim to study which requirements of the project can be fulfilled using existing solutions and which requirements

Table 1 Requirements elaboration template

ReqID	D1.2-4.7
Requirement	Service consumers MUST be able to discover service providers that commit to meeting their policies
Justification	Service consumers are not able to know beforehand which service providers exist, and whether the existing ones can meet the consumers expectations with respect to the policies and functionality they can provide
Interaction	Depends on D1.2–4.8, D1.2–4.9; Supports D1.2–4.1

demand further research and development. Accordingly, our approach for gap analysis is based on the following three activities:

1. Identify and list existing solutions that address the objectives and problems identified during requirements elaboration;
2. Determine which of these solutions are to be employed in the project and specify whether elaborated requirements are fully or partially fulfilled given the selected solutions; and
3. Define future activities necessary to fulfill the requirements that are not addressed by existing solutions and plan activities for the validation of such requirements.

The first step of the gap analysis consists of collecting information on alternative solutions that are relevant to the project. The information is collected using a template that includes the following fields: *name of solution*, which is used to identify the solution; *link*, which indicates where the solution can be downloaded; *functionality*, which describes the functionalities provided by the solution; *limitations*, which describes the limitations of the solution with respect to the needs of the project; *related requirements*, which lists the requirements that the solution fully or partially fulfills; and a *justification of selection*, which describes in natural language the motivation for selecting a given solution to be used in the project (if it is selected for use). The template also includes a field called *access* that indicates whether a considered solution is *open source*, *proprietary*, or subject to both types of licenses, here called a *dual licensing system*. An example of a solution selected for use in the project, specified using the template, is given in Table 2.

The input collected using the template can be summarized in tables that provide an overview of all the solutions considered in the project (Table 3). To better evaluate the relevance of alternative solutions to the project, viewpoints that have shared solutions can be grouped together. For example, Table 3 presents the solutions considered by Workpackage 3 (Securely Adaptable Business Processes), Workpackage 7 (Identity Management, Authentication and Authorization), and Workpackage 10 (Quality Measures and Trustworthiness) in TAS[3]. In the table, columns represent the solutions, i.e., s1–s14, the top row (*access*) represents the licensing scheme of the given solution, while the other rows represent which requirements are fulfilled by the considered solutions. The selected solutions are highlighted with gray columns. The solutions between columns that are delimited using empty narrow columns show which solutions are being considered as alternatives.

Here we do not investigate methods for determining which solutions are more appropriate for the project, e.g., see [1, 39]. In our setting, the selection of the solutions was made by the project partners who have the domain knowledge necessary to evaluate those solutions based on criteria of their interest. The existing solution templates provide a summary of the solutions available on the market for achieving the objectives of the project. Based on this list, the system designers have to negotiate which solutions are more suitable for the project. This decision can be made based on a number of criteria, such as previous experience with a certain software product, social and organizational issues, the platform in which the software product runs, vendor support, performance, the costs and risks associated with selecting an existing solution, etc. [1].

The final activity of the gap analysis consists in defining a plan of the research and development activities that have to be performed to fulfill the requirements that are not or are only partially covered by the selected solutions. The planning and documentation of these activities includes a description of how the partners will validate the fulfillment of the requirements.

3.4 Interaction analysis

An important aspect of the RE process is the identification of the relationships between requirements [61]. This analysis is an important part of viewpoint integration [64]. Our approach determines and evaluates the relationships between the requirements through the analysis of:

Table 2 Solution template

Name of solution	Trust policy wizard
Link	http://i40virt02.ipd.uka.de/CoSim/
Access	Open source
Functionality	Allows guided interactive formulation of trust policies
Limitations	Only supports behavior-based trust policies
Related requirements	D1.2–5.9 (Fully)
Justification of selection	Providing a wizard is a powerful yet straightforward way of supporting user selected policies. We do not exclude the possibility for more integrated solutions such as natural language policy editors

Table 3 Existing solutions considered by WP3, WP7 and WP10 and the related TAS[3] requirements

Solutions	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
Access	O	Pr	Pr	Pr	O	O	O	O	O	O	O	O	O	O
D1.2-3.1	F	F	F	F	F									
D1.2-3.2	F	F	F	F	P									
D1.2-3.3	F	F	F	F										
D1.2-3.4	P	P	P	P	P									
D1.2-3.5						P								
D1.2-3.6						P								
D1.2-7.1						P								
D1.2-7.2						P								
D1.2-7.3								F	F					
D1.2-7.6						F								
D1.2-7.7												F		
D1.2-7.9						F								
D1.2-7.10												F		
D1.2-7.12						P								
D1.2-7.13						P								
D1.2-7.14						P								
D1.2-7.15						P								
D1.2-7.16					F	P								
D1.2-7.17							F							
D1.2-7.18								F	F					
D1.2-7.21						P								
D1.2-7.23						P								
D1.2-7.24						F								
D1.2-7.26												F		
D1.2-10.1													F	F
D1.2-10.2										F	F	F	F	F
D1.2-10.8										F	F	F		

O indicates that the solution is open source and Pr indicate that the solution is proprietary. F indicates that the solution completely fulfills the requirement while P indicates that the solution partially fulfills the requirement

1. The interactions among technical requirements (both intra- and inter-viewpoints)
2. The interaction among legal and technical requirements
3. The mapping of requirements to the architecture.

As we described in Sect. 3.2, the template for requirements elaboration includes the field "interaction" to capture the interactions among requirements. We visualize these interactions among requirements using *requirements interaction graphs* [43]. In these graphs (see Fig. 2), each node represents a requirement, while labeled and directed edges indicate the type of interaction between two requirements. Circles around graphs indicate the workpackage from which the requirements originate.

The use of requirements interaction graphs is twofold. First, they make it possible to prioritize requirements by analyzing the chains of requirements dependencies in the graph. Second, requirements interaction graphs can be used for requirements validation [23]. In particular, they allow system designers to detect overlapping, redundant or conflicting requirements between viewpoints as well as inconsistencies in interaction specifications (e.g., two requirements depending on each other).

The consistency of viewpoints is achieved by finding and evaluating *inconsistency candidates* in the requirements interaction graph, and then by eliminating those inconsistency candidates that turn out to be actual inconsistencies. Inconsistency candidates include groups of requirements that are either indicated as being conflicting or similar [12], or that include inconsistencies in requirements specifications such as:

– *Homogeneous interaction cycles*: cycles with the same interaction type, e.g., "*A* depends on *B*", "*B* depends on *C*" and "*C* depends on *A*";
– *Heterogeneous interaction cycles*: cycles which are not homogeneous and may be unreasonable, e.g., if "*A* depends on *B*" and "*B* abstracts *A*", it means that a requirement depends on its specialization.
– *Non-cyclic interactions*: combinations of unacceptable multiple edges, e.g., "*A* supports and depends on *B*", as well as unreasonable combinations comparable to heterogeneous interaction cycles, e.g., "*A* supports and abstracts *B*".

Inconsistency candidates can be seen as patterns defining the sequence of edges that should not occur in the requirements interaction graph. If the graph contains a path that matches any of the above patterns, we interpret it to mean that the requirement specification may contain an inconsistency, i.e., an inconsistency candidate. The identified inconsistency candidates need to be analyzed to determine whether an inconsistency between the requirements exists or if the detected pattern is acceptable. In the case that an inconsistency is confirmed, it has to be

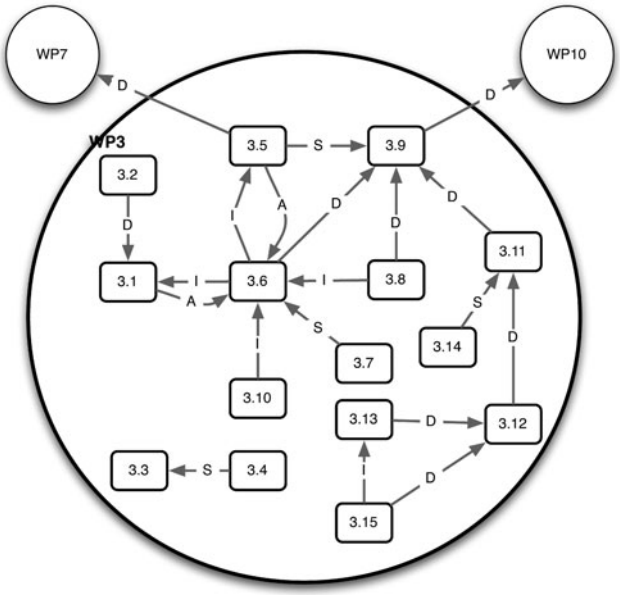

Fig. 2 Requirements interaction graph. The labels are defined as follows: *S* supports, *D* depends on, *I* implements, *A* abstracts

resolved by rephrasing, refining, deleting or merging requirements. The analysis and resolution process are reiterated until no further inconsistencies are detected.

A major challenge of the TAS^3 requirements analysis is to capture the legal (mainly privacy and data protection) requirements of the target architecture and to analyze the relationships between these and technical requirements. The interaction between legal and technical requirements remains an under-researched field. Previous work in this field focuses on the articulation of data protection legislation as requirements [6, 27] but not on how legal and technical requirements can be consolidated during requirements engineering.

Further, due the nature of legal requirements, the relationship between legal and technical requirements need to be expressed differently than the relationship between technical requirements. The fulfillment of a legal requirement may be contingent on matters beyond technology. Hence, the semantics of legal requirements may not be as precise as the semantics of precisely articulated technical requirements. For example, a legal requirement *L* may have different parts, e.g., (*a*), (*b*) and (*c*). There may be technical requirements, that, if satisfied, would fulfill the corresponding parts of the legal requirement. However, the satisfaction of (*a*), (*b*) and (*c*) may not be sufficient for satisfying *L* as a whole. Further, this relationship between the satisfaction of parts and its composition may sometimes be an issue and at other times not relevant.

Despite the difference in the semantics of satisfaction of legal requirements, we identified the following activities to analyze the interactions between legal and technical requirements:

1. Identify data protection requirements that can be fully or partially technically satisfied;
2. Identify data protection requirements that cannot be technically satisfied.

We developed the template in Table 4 to document interactions between legal and technical requirements. The fields of the template are to be interpreted as follows:

– *Is fulfilled by*: a technical requirement fulfills a legal requirement or parts thereof;
– *Is partially fulfilled by*: technical requirement partially fulfills a legal requirement or parts thereof;
– *Not fulfilled*: there is no technical requirement that fulfills the legal requirement or parts thereof;
– *Conflicts with*: (the implementation of) the technical requirement violates the legal requirement;
– *Comments*: describes why the legal requirement is not sufficiently fulfilled (but should be) and states what additional work is needed for the fulfillment of the legal requirement. If additional work is needed, the candidate workpackages responsible for the technical requirements and development activities are indicated.

The last step of interaction analysis is the mapping of the requirements to the architecture. This mapping can be defined using a simple template that for each requirement describes the corresponding feature(s) of the architecture. The mapping of requirements to the architecture is necessary for three reasons. First, there is a danger in viewpoint oriented requirements analysis that global requirements are neglected in the process of consolidating the different viewpoints. Second, the mapping helps to detect missing requirements and overlapping requirements that may have been unnoticed during earlier RE activities, and to determine missing or redundant elements of the architecture itself. Last but not least, the TAS^3 architecture was designed independently from the outcome of RE activities. Therefore, the mapping provides a means to validate whether the designed architecture actually implements all the elaborated requirements.

3.5 Evaluation

The RE process ends with an evaluation phase that encompasses the whole project. The evaluation is performed using a method based on key performance indicators [53]. Performance indicators are a measure of performance, which are commonly used in organizations to evaluate the progress and success of a project [20]. An

Table 4 Legal requirements interaction analysis template

Source requirement	Interaction type	Target requirement
D1.2–6.9, D1.2–6.70, D1.2–6.72	Is fulfilled by	
	Is partially fulfilled by	
	Not fulfilled	X
	Conflicts with	
comments:	Requires additional work by all workpackages: (a) Identification of which actors within the TAS^3 network shall assume these tasks (taking into account separation of duties)	

approach for defining the measure of performance and their evaluation consists of the following activities:

1. Define success indicators;
2. Refine success indicators into measurable evaluation criteria;
3. Identify methods for the evaluation of evaluation criteria.

The first step consists in the definition of *success indicators*, which are the aspects to be measured in order to evaluate the long-term goals of project. In particular, these indicators represent those aspects that are important for the stakeholders and whose achievement provides evidence of the success of the project. The first column of Table 5 presents an excerpt of the indicators concerning RE activities considered in TAS^3.

Success indicators are general; therefore, they have to be refined into measurable *evaluation criteria*. The second column of Table 5 presents some evaluation criteria for each indicator. For instance, a criterion for evaluating the quality of requirements documentation is to verify whether requirements have been expressed according to the controlled vocabulary (Sects. 3.1 and 3.2). The quality of documentation can also be measured by verifying if requirements are articulated comprehensively (e.g., a justification for each requirement is provided) and organized. The status of requirements fulfillment can be assessed by the number of requirements mapped to the architecture and the number of requirements that require new research or development of new components for their fulfillment.

Due to the different nature of criteria, different techniques should be used for their evaluation. The last column of Table 5 shows the evaluation method adopted for each criterion. These evaluation methods can be classified as *quantitative*, *report* and *interviews*. Quantitative methods can be used when the criterion can be characterized by a precise measure. For instance, quantitative methods are suitable to assess the status of fulfillment of requirements. Reports are used when the criterion is both quantitative and qualitative, and its evaluation requires a text description. The use of a report is suitable, for example, to assess the quality of requirements documentation.

To evaluate the criteria that require user interaction, more sophisticated methods are required. Most common evaluation methods are ethnographic approaches [5], interviews [21, 26], focus groups [8], wizard of oz [47], paper prototyping [59], rapid prototyping [32], storyboarding [2] and expert walkthrough [49]. Among these methods, interviews provide an effective method to collect information. For example, interviews can be used to verify whether stakeholders feel that their requirements have been addressed properly in the architecture.

4 Requirements engineering in TAS^3

In this section we report our experience in the application of the approaches presented in Sect. 3 to the TAS^3 project. Here, we only present the main findings and refer to [29, 48] for the complete results.

4.1 Requirements elicitation

The aim of requirements elicitation in the TAS^3 project is to elicit technical, legal and user requirements for the TAS^3 architecture with a particular focus on the needs of the application domains. Accordingly, we first defined some pilot scenarios anchored in the application domains together with the domain holders. In particular, three scenarios in the employability domain and one in the healthcare domain were identified by the workpackage running the pilots. A total of 19 actors were identified and analyzed in these four scenarios. The pilot workpackage also identified 45 threats that can compromise the security of the system. For example, in the healthcare scenario, the analysis showed that any actor, be it a physician, a paramedic or an external malicious attacker, may steal a patient's credentials and impersonate that patient. Such a person could then modify security policies, get access to the data or provide access to other colluding actors. The discovered threats

Table 5 Success indicators, criteria and evaluation methods

Indicator	Criterion	Method
Security and privacy requirements captured	Relevant requirements have been captured	Interviews
	Requirements elicited after initial iteration have been integrated	Report
Fulfillment of requirements	Number of requirements mapped to architecture's components	Quantitative
	Number of requirements that demand new research or development of new components	Quantitative
	Partners see their requirements fulfilled	Interviews
Quality of requirements documentation	Use of a coherent language to express requirements	Report
	Requirements organized and categorized	Report
	Adequate justification and references	Report

provided input to the workpackage that was responsible for executing security risk analysis in TAS^3.

In collaboration with the project partners, we elicited and analyzed requirements for the TAS^3 architecture based on the selected pilot scenarios. As instructed, each partner provided their requirements using the controlled vocabulary. The requirements were collected in a single requirements document although, in order to document the requirement source, they were numbered per workpackage. The elicited requirements have been collected according to the following regrouping: 30 requirements focused on the steps in the scenarios, 40 on global architectural requirements, another 40 on legal requirements and 15 on additional requirements on technical validation relative to the testing phase.

For the analysis and refinement of requirements, we had to solve the problem due to the lack of a common modeling framework among partners. In order to do that, we first modeled the scenarios ourselves (using Secure Tropos and sequence diagrams) and then let the partners validate the resulting models. Models were then revised according to the partners' feedback. In addition, the formal analysis techniques offered by Secure Tropos were used to support the analysis and refinement process.

4.2 Requirements elaboration

In the requirements elaboration step, partners were asked to elaborate the requirements elicited in the previous step based on their workpackage objectives (viewpoints) using the template presented in Sect. 3.2. We assisted the partners in elaborating their viewpoints by providing them with instructions on how to fill in the templates. The templates and the instructions were first tested with a small group of partners and improved on the basis of their feedback. They were then rolled out to all partners. This additional testing step ensured that the instructions in the templates were clear and the assignments were not redundant.

During requirements elaboration, we supported partners mainly through written electronic communication (emails), but also through phone conferences and, occasionally through face-to-face meetings. During the 2 months of intensive communication, we iteratively reviewed the inputs from partners. This was instrumental to reaching a comparable level of requirements granularity among all workpackages. In particular, we rephrased requirements that described 'solutions' to state 'problems', improved justifications and made sure that interactions among the requirements were specified using the controlled vocabulary.

At the end of this first requirements elaboration process, a total of 163 requirements were captured. Of these 163 requirements, 22 are global requirements, 17 are legal requirements, while the rest are from the technical requirements of the various viewpoints (i.e., workpackages). These requirements were later used as input to the gap and interaction analysis.

During interaction analysis, the partners were asked to reiterate the requirements elaboration step. This second reiteration of elaboration activities occurred 1 year after the first iteration. Partners were asked to capture any changes to the requirements due to developments in their research or due to the progress in the development of the architecture. During this second iteration, a total of 21 new technical requirements were captured, 17 existing technical requirements were edited while 13 were deleted. Further, a total of 79 legal requirements were elaborated in the second iteration of requirements elaboration. All requirements were further refined in the following interaction analysis steps.

4.3 Gap analysis

At the beginning of the gap analysis step, partners were asked to provide an overview and analysis of the alternative solutions that can be used in their workpackage using

the template presented in Sect. 3.3. A total of 53 existing solutions were considered by the project partners. Together with the list of candidate solutions, partners indicated which solutions were selected for the project and provided justifications for their selection. Among the 53 solutions considered, 24 were selected for use in the TAS[3] project.

In some cases, multiple solutions with similar functionalities were adopted by the project. This was due to various reasons: in a few of the cases, it was difficult to foresee which solution is more appropriate; in other cases, partners had to guarantee plurality of business models (a global requirement) and support multiple software suppliers with the targeted architecture. It is worth noting that an important criterion in software selection was the license conditions of the solutions: partners preferred open source solutions or open standards when they were available.

The way in which gap analysis was organized underlines the role of the RE team in executing the requirements engineering activities in TAS[3]. Unlike in other projects, rather than making decisions on behalf of the partners, the RE team only supported the project partners through the requirements engineering process. For example, the partners were free to choose one solution instead of another based on a number of criteria (see Sect. 3.3). In case of alternative solutions common to different workpackages, partners negotiated among themselves and decided on the solution to be deployed in the project. The RE team only stepped in when there were inconsistencies or conflicts in the decisions made.

During gap analysis, partners also documented the requirements fulfilled by the selected solutions, distinguishing whether a certain requirement is fully or only partially fulfilled by a certain solution. With the selected solutions, 45 of the 163 requirements were satisfied fully while 18 were satisfied only partially. For those requirements not fulfilled by existing solutions, partners documented research and development activities that had to be completed for the satisfaction of those requirements, including descriptions of plans for validating their fulfillment.

4.4 Interaction analysis

During interaction analysis, we first analyzed interactions within each workpackage (intra-viewpoint interaction analysis). In total, 120 intra-viewpoint requirements interactions were captured. We represented these interactions using the requirements interaction graphs described in Sect. 3.4. Based on the graphs, we prioritized requirements based on the number of dependencies as well level of abstractness. These graphs and the prioritization analysis based on these graphs were then validated by the corresponding partners. We did not performed the analysis of conflicting and overlapping requirements within single viewpoints, since the requirements in a viewpoint were elaborated by a small group of partners collaborating together and such interactions were avoided.

Next, we studied the interactions among requirements across viewpoints. We also used this phase to integrate viewpoints into a monolithic requirements document, Partners were invited to visit the viewpoints of other workpackages and to document the relationship between their requirements to those of other workpackages. A total of 518 inter-viewpoint interactions were captured among the 146 technical and 17 legal requirements. A straight forward visualization of all these interactions was, however, unfeasible (see Sect. 5.5 for a discussion). Therefore, we developed an automated analysis tool for detecting inconsistency candidates in the interaction graph.

We visualized each identified inconsistency candidate in a graph form using the Graphviz DOT[2] format. We then invited partners to use a collaborative and interactive environment in which the partners could see (partial) graphs representing inconsistency candidates. If the inconsistency candidate required changes, these were arranged by the responsible workpackages. Once the necessary changes were completed, the partners could update the requirements interactions graph accordingly. We selected the Trac wiki tool[3] as our collaborative environment to update the requirements interaction graphs. The advantage of this tool is that it supports the editing of Graphviz DOT files in wiki pages and hence the collaborative editing of the inconsistencies in the requirements interaction graphs.

To solve inconsistencies, we also introduced an order in the types of inconsistencies that were addressed. We first asked the partners to discuss those requirements that were indicated as being similar, a total of 20 (no requirements were found to be conflicting). Based on their discussions, requirements and interaction graphs were modified. These changes were used as input to the second iteration of the elaboration step. After these elaboration activities, the partners were asked to update their inter-viewpoint requirement interactions with respect to the new, edited and deleted requirements.

Next, we analyzed the inconsistency candidates based on the pattern catalog presented in Sect. 3.4. The inconsistency detection analysis tool detected 62 homogeneous cycles and 3 heterogeneous cycles. Given the overhead of discussing so many inconsistencies, we organized a face-to-face workshop during which we asked partners to communicate with each other in order to verify whether the inconsistency candidates correspond to actual inconsistencies. Based on

[2] http://www.graphviz.org/.

[3] http://trac.edgewall.org/.

their discussions, the partners updated the requirements interaction graph and we ran our tool to detect whether new inconsistency candidates had emerged.

After three rounds of inconsistency analysis which included numerous additions, edits and deletions to the requirements, we reached a requirements interaction graph free of inconsistencies. This final inconsistency-free graph had 154 technical requirements with 358 interaction relationships. After this activity, all the technical viewpoints were merged in a single technical requirements document.

Once the requirements, and hence the viewpoints, were consolidated in a single document, the legal team reviewed the technical requirements to evaluate the extent to which the legal requirements have technical counterparts. The legal team was supplied with 124 of the 154 technical requirements.[4] The team filled out the legal requirements interaction template (Table 4) for their 79 legal requirements. The results were communicated to the partners during a workshop, after which both the technical and the legal requirements were revised. In particular, 13 new legal requirements were captured, 13 were edited and 4 were deleted. Further, 1 new technical requirement was captured, while 4 others were edited.

The last step of interaction analysis was the definition of a mapping between the elaborated requirements and the components of the architecture. This mapping was completed in collaboration with the architecture team in a number of steps. First, we mapped the global requirements, which were initially defined by the architecture team, to the components of the architecture. The results of the mapping helped the architecture team to discover gaps in the main objectives of the project, reflect on the global requirements and improve the system architecture.

Next, the architecture team mapped the technical requirements to the architecture. The team documented redundancies in the requirements (a total of 8 were indicated), pointed out requirements that are out of the scope of the architecture (a total of 3) and identified one requirement that should have been but had not yet been addressed in the architecture. These were communicated to the partners before the second iteration of the elaboration step. After the inconsistency analysis, the mapping of technical requirements to the architecture was reiterated. Requirements not (yet) satisfied by the existing architecture (5 global requirements, 3 workpackage requirements) were captured in a document, which was communicated to the project partners.

[4] Out of the 154 requirements, 30 stemmed from a workpackage whose main role was the integration of the project results. The requirements of this organizational workpackage had no interactions with the legal requirements.

Finally, the architecture team responded to the analysis of the interaction of legal requirements with the architecture. Specifically, they analyzed those legal requirements that demanded further work from the architecture. Of the 38 legal requirements that demanded additional technical work, 2 were identified as satisfiable only in a domain-specific instantiation of the architecture, 2 were not addressed, 5 described necessary additional work, 1 demanded a refinement of the legal requirement and the rest had already been satisfied. The results of the analysis were reported to the project partners.

4.5 Evaluation

In the TAS^3 project, two sets of success indicators have been defined for the evaluation of the project: *project-based* indicators, which aim to evaluate the project as a whole, and *WP-based* indicators, which aim to evaluate the performance of single workpackages. These success indicators have been extracted from the DoW and represent general goals of the project and workpackages, respectively. Subsequently, partners provided precise, measurable and reachable evaluation criteria by refining those general indicators. Together with the evaluation criteria, partners also identified methods appropriated for their evaluation. A total of 10 success indicators and 20 evaluation criteria for the evaluation of the RE workpackage have been defined.

So far, the project consortium has only defined success indicators and evaluation criteria. Evaluation criteria will be used to measure project performance at the end of the project. Project partners are supposed to evaluate project and workpackage results by providing a competent and impartial opinion on the considered targets.

5 Lessons learned

This section discusses the most important lessons learned from the application of RE methodologies in the context of the TAS^3 project. A summary of the lessons learned is presented in Table 6; they are further discussed in depth in the remainder of this section. Lessons learned are organized according to the challenges in Sect. 2. For each lesson learned, first we describe the advantages and disadvantages of the approaches adopted in the project; then we draw conclusions and provide general guidelines on how to address the issues.

5.1 Project planning

Definition of the scope Most issues we faced during the RE process originated from the DoW. Some activities were under-specified, e.g., gap analysis; other activities that are

Table 6 Challenges-lesson learned

Challenge	Lesson learned
Project planning	The scope of RE activities shall be defined accurately and their importance should be recognized from the onset of the project
	RE activities shall not overload project partners
Requirements definition	The information required from the stakeholders and its level of abstraction shall be appropriate for the needs of the project
	The ability of partners to elicit and elaborate requirements shall be assessed at the beginning of the project
Requirements elicitation	The selected methodologies shall be appropriate for the achievement of the project objectives
	Templates shall be tailored to the needs of the project
Gap analysis	The RE process (and gap analysis) shall help to determine the research direction of the project and drive project partners' activities
	Validation activities shall include analyzing requirements for consistency and completeness, as well testing the fulfillment of individual requirements, refined to an appropriate level of granularity
Requirements communication and agreement	A common format for the documentation of the RE process and artifacts should be decided and enforced from the beginning of the project
	A collaborative environment (compatible with the common format) shall be employed to facilitate the communication among partners
	Graphical representation as well as tools that allow for scalable management and formal verification can be used to improve requirements consistency and completeness
	Viewpoints analysis can be used to better manage a large and complex requirements engineering process. The use of such an approach shall be coupled with steps to capture, analyze and validate global requirements

logically related to RE activities, e.g., risk analysis, spanned across different workpackages. In addition, the terminology used in the DoW was vague and inaccurate, e.g., "design requirements". These issues led to a misunderstanding of the RE activities to be performed.

The shortcomings of the DoW are symptoms of a greater problem: RE activities are often not considered and planned at the very beginning of a project. This problem often establishes itself in the planning of the consortium: unless the project is RE related, it is unlikely that partners with RE expertise are included in the consortium. Instead, RE activities are usually assigned to some project partner that will eventually perform them in addition to achieving their own objectives and interests in the project.

Beyond defining the general objectives of the project, the DoW functions as a reference document in organizing and coordinating the activities to be performed within the project, establishing assignments to each partner, setting priorities and defining a mitigation plan for possible problems. We argue that the DoW should also include a description of RE activities. In particular, it should clearly define the objectives of RE activities, who should be involved, and how the results should be integrated into the rest of the project. If the DoW mentions different RE activities and expected results, these should be checked for overlaps, inconsistencies and missing parts. Finally, it is advisable that the results of RE activities are regarded as an important milestone of the project instead of a by-product with no recognition.

Workload negotiation The execution of the RE process often requires the iteration of RE activities until initial requirements are refined into a verifiable set of requirements. Given the pressure to produce results in a limited time, project partners may consider RE activities and their iteration cumbersome and time-consuming and prefer to focus on their own project activities. Each iteration can be seen by partners as a futile exercise, especially if the benefit of the iteration is not immediate. In addition, reiterations may frustrate those partners that are more engaged with the RE process as they recognize how the contributions of less engaged partners lead to slow and sometimes pointless iterations. Consequently, the global interest partners have in RE activities can decline rapidly.

To address these frustrations, it is important to communicate to the partners that RE process may facilitate the execution of the project. Further, it is important to avoid that RE becomes (or is perceived as) an add-on activity with severe time costs and no returns. To increase the willingness of the partners to participate in the RE process, a well-defined plan of RE activities, instructions on how

such activities should be performed, and the input expected from the partners should be defined and negotiated with the partners before the execution of the RE process. In addition, the RE team should ensure that RE activities are not redundant. In case of conflicting requirements between workpackages, the RE team should facilitate the communication between the involved partners to solve them, while keeping an overview of the possible outcomes of the resolution of the conflict in terms of workload and time planning. In addition, the number of iterations of the RE process should depend on the willingness of the participants as well as the needs of the project. When a discrepancy between the more and less engaged partners becomes evident, measures should be taken and workload should be redistributed.

To compensate for problems arising from a discrepancy of inputs contributed by the partners, we introduced quality checks before distributing intermediary results. This guaranteed that the input provided by all partners was at a comparable level. However, such synchronization of partners may also lead to further problems. For example, if some partner is late in providing his/her input for such a synchronization step, the other project partners may accrue additional work. Concretely, in TAS^3, a partner completed the second iteration of the requirements elaboration step after all the other partners had completed the requirements interaction analysis on the revised requirements. Including the delayed contribution in the requirements document would have required every partner to redo the interaction analysis. Due to a number of reasons (e.g., the short deadline for the deliverable, the work overload and communication overhead that their integration would have led to), we decided not to consider the delayed requirements in that iteration of the requirements document. The RE team should take into account the effects on all project partners when absorbing the difficulties resulting from such delays.

5.2 Requirements definition

Information management The analysis of requirements benefits from the collection of large amounts of information such as information about the environment in which the system-to-be has to be deployed; the perspective of different stakeholders; elaboration of quality requirements; exhaustive evaluations of existing solutions; domain assumptions. However, gathering all this information can be time-consuming and can burden the partners. Therefore, it is necessary to decide on a trade-off between the amount of information to be collected, the needs of the project and the willingness of the participants to contribute to the RE process.

The scope of the information to be collected and processed in the RE process should be identified at the beginning of the project. The scoping of the information to be collected significantly determines the type of analysis that can later be performed. Hence, the information to be collected should be aligned with the project objectives. For example, the licensing scheme under which software solutions are published played a key role in their adoption for the TAS^3 project. Therefore, this information was collected during the gap analysis. In another project, the deployment of a solution may depend on its compatibility with a certain platform. In this case, in line with the project objectives, it is necessary to collect information about the platform in which a software product runs.

In addition, the scoping of the information should exclude the collection of unnecessary details or categories of information. The collection of information unnecessary for the analysis may mislead the project partners. For instance, the traditional partitioning of requirements into functional and non-functional requirements caused a number of problems in TAS^3. In a trust and security project, where the objective is to provide security functionality, such a distinction became ambiguous and confusing to the partners. Therefore, the distinction between functional and non-functional requirements did not benefit the requirements elaboration process and was dismissed during the RE activities in TAS^3.

The necessary granularity of the information collected also plays a fundamental role in the RE process. Although some RE activities (e.g., gap analysis and interaction analysis) can be performed even if the information is collected at a coarse-grained level of granularity, other activities (e.g., the definition of a validation plan for research requirements) necessitate the elaboration of fine-grained requirements. However, arriving at fine-grained requirements requires several iterations for their gathering: a time intensive activity. This again requires an evaluation of the trade-off between time intensive requirements activities and the willingness of the project partners to participate in them. Such an evaluation requires balancing the level of requirements consistency with partners' time and motivations.

Abilities and training Defining the RE activities and the input required from the partners is often not sufficient to carry out the RE process in large projects. When several partners are involved in the RE process, RE activities, their justification and outcomes have to be communicated and negotiated with the partners. This includes the establishment of a common understanding of what requirements are, the necessary granularity of the captured requirements, and the different roles that the RE participants will play in the RE process [14].

In the case of TAS^3, during the short introductions to the RE activities at project meetings the partners did not demand any negotiation or clarification of the activities. However, they later struggled with the use of templates and

with the command of controlled vocabularies. As a result, the partners made mistakes, which meant they had to repeat the analysis, leading to discontent with the RE process. To overcome these issues, it is highly recommendable to organize RE workshops with the participants of the RE process at the beginning of the project. The objectives of these workshops are (1) to bring the partners and stakeholders to a common understanding and acceptance of what requirements are; (2) to provide them with an overview of the RE activities to be performed together in the course of the project; and (3) to provide them with instructions on how to use the employed tools, e.g., templates, controlled vocabulary, collaborative environments.

However, dedicated RE workshops may be difficult to organize due to the number of partners involved and their geographical distribution. If this is the case, then these should be organized as special sessions that take place during project meetings, e.g., during the kick-off meeting of the project.

5.3 Requirements elicitation

Method selection Several methodologies are available to support the execution of RE activities. However, there is no universal methodology that fits all projects. Each project has different objectives that require focusing on specific aspects of a system and its development. These aspects drive the selection of the methodologies to be adopted in the project.

For example, the TAS^3 DoW requires that the elicited requirements are testable. This demanded the use of methodologies for achieving testable functional and security requirements. Our evaluation of different methodologies showed that only a few of the existing methodologies provided testable security requirements: in [41], security and privacy are treated as *softgoals*, which by definition are not testable; in [35], *quality constraints* are proposed as testable "approximations" of softgoals. Other methodologies (e.g., [36, 63]) were not suitable for TAS^3 because they only focus on the system-to-be without analyzing the organizational context in which the system-to-be operates.

In our methodology of choice, Secure Tropos [44], security requirements are not explicitly represented in the model (as non-functional); it formally verifies the compliance of requirements models with security requirements using security constraints. Further, Secure Tropos pays particular attention to the analysis of security dependencies (i.e., trust relations and permission delegations) among scenario stakeholders and between stakeholders and the system-to-be. Finally, given the importance of trust, delegation and organizational processes in TAS^3, we decided that Secure Tropos was the most appropriate for the project among the available methodologies.

For the elicitation of the security threats, we used a simple threat model (see Sect. 3.1). More sophisticated methodologies for risk analysis (e.g., [3, 18, 42]) exist. However, according to the DoW, an exhaustive risk analysis was not considered to be part of the RE activities. Hence, we sufficed with the simple analysis.

The use of modeling frameworks and methodologies specific to security requirements analysis is beneficial for the RE process [19, 45, 46]. Nonetheless, their application needs some expertise that partners usually do not possess. Such methodologies provide systematic methods for the elicitation and analysis of security requirements, and some of them, like Secure Tropos, automated tools for formal requirements analysis. At the same time, they provide a (graphical) notation that may not be intuitive. To keep the benefits and minimize the efforts required by partners, we decided to collect requirements specifications in natural language. Later, those specifications were used by RE experts to draw requirements models with the selected methodologies. The resulting models were then validated by the partners.

Template definition Given the limited time we had to redo the requirements analysis, some deliverables, including the ones reporting the results of RE activities, were rejected by the EU Commission at the first project review. The second iteration of these deliverables had stricter time constraints. we had to find pragmatic solutions for the elicitation, elaboration and analysis of the requirements. Methods based on templates are suitable for the elicitation and documentation of requirements in large research projects in which partners with different cultures, work rules and languages participate [37]. Our search for validated requirements templates that were directly applicable to our project, however, returned no results. First, it was difficult to find a template that had the right level of granularity or abstraction for the complexity of the project. Moreover, existing templates do not include all (and only) the information relevant for the project at hand.

For these reasons, we developed our own templates based on existing templates [60, 65]. Defining customized templates has an important advantage: It makes it possible to ask exactly those matters that are relevant for the project. For example, in TAS^3 there was a discrepancy between the needs of the pilot developers, who were eliciting requirements for a real world application, and those of the research groups, who were eliciting research requirements while abstracting away domain assumptions. The granularity and complexity of these requirements are difficult to capture in one single template. Yet, having several templates makes it difficult to analyze all the requirements together, e.g., interaction analysis may become infeasible.

At the same time, there is an important disadvantage to developing customized templates: the templates need to be

validated and may require modifications to improve their use. In order to validate the new templates, we defined a process in which templates are developed, tested and rolled out. The objective of the process was to minimize callbacks and rollout iterations. Nevertheless, even after rollout, the templates needed additional adjustments depending on the needs of the various workpackages. Every change to the templates had to be broadcasted to all partners. This lead to a communication overhead and occasional confusion among project partners. Since such modifications are inevitable, even with validated templates, project partners should be warned against possible repetition of tasks due to template refinement during the project, especially if customized templates are going to be introduced.

5.4 Gap analysis

Gap analysis Whereas in some fields gap analysis is an established practice (e.g., marketing, environmental studies), we found no references to gap analysis in the RE literature. Therefore, we had to use literature from other fields to define the scope and objectives of the gap analysis. In particular, we defined the scope of the gap analysis as the identification and documentation of the necessary research and development activities in the project. Accordingly, we developed a method for supporting the analysis (see Sect. 3.3). However, our method has room for improvement. For instance, the requirement for the architecture to be compliant with data protection regulations underlined the need for research on how to do domain-specific gap analysis. We see the development of systematic methods for gap analysis in software engineering as a challenging and intriguing topic for future research.

Validation Finding an appropriate methodology for the validation of research-related requirements is a challenge. We based our validation activities on the definition provided in [23]: validation activities check that the requirements specification captures the actual needs of the stakeholders—this includes checking that the specification satisfies expected internal consistency properties.

To validate the fulfillment of individual requirements, we asked partners to provide a validation plan that explains the activities they will execute to show that the components they develop will fulfill the given requirements. Further, during different steps of the interaction analysis, the partners validated the consistency and completeness of elaborated requirements with respect to the different viewpoints, legal requirements and the architecture.

The granularity of the elaborated requirements and the resulting information trade-off, as discussed in Sect. 5.2, is an issue when validating requirements. For example, the validation activities suggested by the TAS^3 partners are not very detailed. This is partially due to the fact that some of the requirements still need to be refined into testable requirements, an extra iteration step that was difficult to motivate after the consolidation of the viewpoints and given the time constraints. Interestingly, the gap analysis and the interaction analysis are easier to execute when the requirements are at a coarser granularity and the number of requirements are fewer. We therefore observe that during validation there is a difference between the granularity of requirements needed for consistency analysis and for the analysis of the fulfillment of individual requirements.

Based on our experience, we recommend that the validation plans for individual requirements are revisited after the viewpoints are integrated, the requirements are mapped to the architecture components and the requirements are refined into testable requirements.

5.5 Requirements communication and agreement

Common format The execution of RE activities requires partners to interact and to frequently share documents. In our case, the elicitation, elaboration and analysis of requirements were further complicated by the different document formats used by the partners (e.g., Microsoft Word, Latex, and pd and a diverse set of operating systems). Partners were adamant about using their own formats for reasons of convenience, political conviction, security or usability. We initially catered to their needs by providing partners with templates in different formats. This led to a formatting overhead for the RE team at the end of each iteration. In addition, it negatively impacted the collaboration between partners, e.g., collaborative editing was difficult among partners using different document formats.

Therefore, it is advisable to adopt a single common format for the documentation of the requirements. Adopting a common format makes it easier to exchange documents between partners and to collaboratively produce RE artifacts, e.g., fill out requirements templates collaboratively. Choosing a common format, however, is not trivial: it should take into account the operation systems used by partners, the usability of corresponding editing tools, the experience of project partners, etc.

Collaborative environments As discussed in the previous section, it is advisable to adopt a common format to facilitate the communications among project partners. This format should also facilitate collaboration among them and be compatible with the selected (collaborative) communication environment. A number of collaborative environments like SVN repositories and wikis are available on the market. We adopted the Trac wiki tool. In addition to the advantages discussed in Sect. 4.4, the use of wiki addresses the problem related to the adoption of a common format.

We experienced the benefits of the use of wiki particularly in the iterations of the interaction analysis. Solving

inconsistency candidates across viewpoints required well-planned intermediary synchronization among the partners, since changes provided in the requirements of one viewpoint often had an affect on their interactions with the requirements in other viewpoints. The wiki, hence, provided project partners with an environment that allowed them to analyze and resolve inconsistencies collaboratively.

However, the use of online collaborative tools like wikis may cause difficulties in off-line editing and lead to conflicts or confusion between partners with varying editing rights. Further, version control may prove a significant overhead. At the beginning of a project, partners should decide on a collaborative environment after being informed about the advantages and disadvantages of selecting the different environments in the project [17]. Finally, they should agree on acceptable practices for using those environments.

Scalability The outcome of different requirements analysis steps needs to be validated by the partners, i.e., check for consistency and completeness. However, in a large project with numerous requirements, such validation activities can be difficult to master. It is largely recognized that graphical models can improve the readability of requirements interactions and execution of the analysis [9]. Consequently, we adopted and used the requirements interactions graphs to visualize the interactions between requirements, and to facilitate the analysis of consistency and completeness among the viewpoints.

However, as the size and complexity of the targeted system increases, graphical models run into scalability issues [58]. In the requirements interaction graphs, the one-to-one visualization of the interactions fell apart after approximately 20 requirements. The graphs were useful for intra-viewpoint analysis where the number of requirements and the ratio of interactions per requirement were low. However, these graphs proved to be unsuitable for the representation of inter-viewpoint interactions, for which the ratio of interactions per requirement was much higher. Consequently, detecting inconsistency candidates through manually analyzing the graphs was fruitless.

To address scalability issues in our graphical model, we developed an automated analysis tool that detects inconsistency candidates by comparing the requirements interactions graph against the patterns presented in Sect. 3.4. We then only visualized the relevant parts of the graphs, allowing the partners to immediately focus on the problem of interest.

Generally, it is advisable to use graphical models for representing analysis results. It is further recommendable to enhance these models through the development of automated tools that analyze large diagrams and that focus on particular aspects of the graphical model.

Viewpoints analysis In TAS^3, we decided to use viewpoints, which represent the perspectives of workpackages, for a number of reasons. First, in a large-scale project with distributed partners, it is easy to loose overview of a large requirements document. Second, the specification of a complex system is unlikely to be discovered by considering the system from a single-perspective [64]. Finally, given the different backgrounds, concerns, interests and assignments of the different partners in a project, it is organizationally valuable to capture and integrate their different perspectives during the requirements engineering process.

However, viewpoints analysis comes with its own shortcomings and risks. Each viewpoint created by the designers and stakeholders may be at a different granularity and quality. Not only the interests and backgrounds, but also the proficiency in eliciting and documenting requirements may differ among the different participants, leading to a large discrepancy in the viewpoints. This may make it difficult to integrate the different viewpoints, identify inconsistencies and consolidate the requirements. Next, the viewpoints may be useful in capturing partial concerns, but the global view may fall out of scope. Disregarding global requirements may lead to problems in the alignment of the system requirements with the main objectives of the project. The lack of a global perspective may impede upon the analysis and validation of global requirements.

It is however possible to introduce steps to address some of the shortcomings of viewpoints analysis. Additional synchronization can be introduced during the viewpoints analysis to align the quality, content and scope of the different viewpoints. Further, designers with an overview of the system-to-be, in the case of TAS^3 it was the architecture team, can be asked to provide a set of global requirements. These can be discussed with the different participants of the RE process. Further, whether the system-to-be fulfills these global requirements can be validated. Finally, the integration analysis can be organized such that possible conflicts, overlaps and gaps can be identified and addressed by the different partners, enabling a multilateral analysis of such inconsistencies. Hence, we highly recommend the use of viewpoints analysis in large projects as a strategy to better command the complexity and size of the project requirements. However, a viewpoint oriented RE process should also capture, analyze and address the global requirements of the project.

6 Related work

The development of large-scale systems is largely recognized as a critical problem in the RE community. Although several efforts have been devoted to define and improve modeling methods, process technologies and software tools

for supporting RE activities, practical guidelines that scale to the development of large-scale systems are needed. This problem has spurred researchers to identify the challenges that can compromise the success of large-scale projects and to define methodologies and guidelines able to support the RE process in such projects. In the remainder of this section, we analyze the challenges identified in other studies [4, 38, 54, 57, 67, 68] and discuss their impact on the TAS^3 project. Then, we investigate the guidelines proposed in those studies to address the challenges in Sect. 2. The results of these analyses are summarized in Tables 7 and 8.

Challenges for large-scale projects. Table 7 shows some of the most important challenges to be faced in large-scale projects [4, 38, 57, 67, 68]. Notice that these challenges focus on particular problems, whereas in Sect. 2 challenges are grouped with respect to RE activities. As a consequence, some of the challenges in Table 7 are included in the ones discussed in Sect. 2. The most common challenges are scalability and geographical distribution of project partners. The large number of requirements to be specified, analyzed and managed is often recognized as the main scalability factor in large-scale projects [38, 57]. We addressed this challenge by eliciting and managing requirements using viewpoints. We also used automated analysis tools for requirements analysis and for identifying requirements conflicts and inconsistencies that might have been overlooked by executing requirements analysis manually. To deal with the geographical distribution of project partners, we established a common understanding of what requirements are among project partners and defined a common format for their specification during face-to-face meetings. In addition, to facilitate the interaction with and between partners during RE activities, we adopted a collaborative environment (i.e., Track wiki) as well as graphical representations (i.e., Tropos models, UML sequence diagrams and requirements interaction graphs).

Another critical challenge in large-scale projects is scope changes in requirements [38, 67]. Further, these changes may produce traceability challenges and lead to confusion about which version of the requirements should be implemented and by whom [54]. Scope changes had a minimal impact on the TAS^3 project for two reasons. First,

Table 7 Challenges for large-scale projects

Challenges	Konrad et al. [38]	Bergman et al. [4]	Wnuk et al. [57,67,68]	Petersen et al. [54]	TAS^3
Scalability	√		√	√	√
Management of customer expectations	√			√	√
Changing technology	√			√	
Distributed teams	√		√		√
Formal interface to customer	√				
Traceability	√			√	√
Scope change	√	√	√	√	
Resource fluctuation	√			√	
Requirements definition	√			√	√
Requirements dependency				√	√
Gap analysis			√		√
Political ambiguity		√			√
Heterogeneous requirements		√			√
Phase Interactions				√	√
Documentation centricity				√	

Table 8 Guidelines for addressing challenges in Sect. 2

Challenges	Approaches			
	Konrad et al. [38]	Bergman et al. [4]	Wnuk et al. [57,67,68]	Petersen et al. [54]
Project planning				not use waterfall model
Requirements Definition	separation between requirements and design solutions			
Requirements Elicitation	well-structured feature list		requirements architecture	
	specification approaches that scale			
Gap analysis				
Requirements Communication and Agreement	well-structured feature list	political ecology model	scope tracking visualization	
	specification approaches that scale		requirements architecture	
	project status visualization			
	effective documentation standards and review process			

the scope of workpackages was defined in the TAS^3 DoW. More important, scope changes in requirements are usually caused by modification requests from customers. In the TAS^3 project, customers were represented by domain holders which are project partners. In this setting, user requirements were defined at the beginning of the project during the definition of pilot scenarios and no additional demands were made by the domain holders in the course of the project. In addition, the lack of "real" customers made it possible to use informal communication, and the management of customer expectations was addressed in the evaluation phase by interviewing domain holders about the proper implementation of user requirements in the architecture. Last, challenges with respect to who implements which version of a given requirement was addressed through the viewpoints in which the responsibility of implementing the requirement was part of the documentation. Conflicts with respect to implementation responsibilities was addressed explicitly during interaction analysis.

Large-scale projects often fail because system scope and design are unclear. In particular, the lack of a clear understanding among project partners regarding the interaction between technical, business and legal requirements leads to political ambiguity [4] in which each partner interprets the requirements in his favor. In a situation in which the project consortium consists of several partners, partners may disagree on goals or at least on how to achieve them. Therefore, the RE process becomes a political process in which the goals that will be addressed (and those that will not be) are selected. In the TAS^3 project, the RE team acted as a mediator by facilitating the communication between the partners and assisting project partners in reaching an agreement, avoiding large conflicts that could have stalled the project.

Further challenges may occur in the interaction between the requirements phase and the rest of the engineering process. The requirements engineering process may take too long or the scope of the requirements may be too large for the rest of the project to handle [54]. In TAS^3 the requirements engineering phase was initially too short, since the project was architecture driven. The reiteration of the requirements engineering activities was done efficiently under heavy time constraints. Whether the scope of the requirements in TAS^3 were too large, and whether this led to resource fluctuation is a matter of evaluation that can be addressed at the end of the project through interviews and through an analysis of the number of requirements that were discarded.

Although we also faced other challenges identified in other studies [38, 57, 67], most of them had a negligible impact on the TAS^3 project due to its research orientation. For instance, the changes in the technology were limited; resource distribution, milestones and project risks (called resource fluctuation in [38]) are well defined in the TAS^3 DoW, and changes were minimal. Despite that, the TAS^3 project faced additional challenges such as the lack of RE methodologies for gap analysis. Notice that the problem of gap analysis was also identified in [68]; however, in that work gap analysis is used to align market and supplier requirements rather than identifying project innovation.

Guidelines for addressing challenges in Sect. 2. Table 8 presents guidelines proposed in various studies [4, 38, 57, 67, 68] to address the challenges identified in Sect. 2. Konrad and Gall [38] found the customers and system architects tend to describe problems in terms of solutions. They argue the importance of separating between requirements and design solutions, as requirements containing design details need to be updated to accommodate design changes. We also faced this challenge (Sect. 2) and addressed it at the beginning of the RE process by establishing a common understanding of what requirements are among project partners.

Konrad and Gall [38] propose to develop a well-structured feature list and organize software requirements specifications according to features. A feature is a required, externally accessible service of the system. Similarly, Regnell et al. [57] introduce the notion of requirements architecture. A requirements architecture is a structure of requirements which includes the data model of requirements with their pre-conceived and emerging attributes and relations. These approaches are comparable to viewpoints adopted in the TAS^3 project. Indeed, each viewpoint corresponds to a workpackage; each workpackage is responsible to develop a limited number of features. In addition to the development of a well-structured feature list, Konrad and Gall [38] propose to use specification approaches that scale. Specifically, they create gray-box use cases, which are white box on the system level and black box on the subsystems level. The advantage of this approach is that it allows one to capture not only the external view of the system behavior, but also the interaction between software subsystems. As in the TAS^3 project, the pursuit of modular and scalable approaches is necessary to manage the elicitation and analysis of a large number of requirements.

Scalability issues not only affect requirements elicitation, but they also have a negative impact on requirements communication and agreement. Therefore, the use of modular and scalable approaches is expected to have a positive effect on the interaction between project partners. However, further aspects should be taken into account. Konrad and Gall [38] argue the need for effective documentation standards for facilitating the communication and for mutual understanding between partners. In particular, similar to our approach, they use customized templates to maintain consistency among requirements specifications produced by several requirements engineers. The use of

visualizations is also acknowledged in several studies [38, 57] as an effective means to inform partners about project status and design decisions that are made. Another facet of requirements agreement is to study the political ecologies in which requirements emerge. To this end, Bergman et al. [4] propose a political ecology model to study the negotiation between project partners and discuss its implication on RE activities.

It is worth noting that other studies have not proposed guidelines for project planning and gap analysis. The main reason for which project planning is not considered in these studies (with the exception of Petersen et al. [54] who argue that "waterfall model" practices are not suitable for large projects) is that RE is recognized as a fundamental phase of the software development process. On the other hand, gap analysis has not been taken into account in other studies since their focus is on projects whose ultimate goal is to deploy a software product rather than develop innovation.

7 Conclusions

In this paper, we presented our experience in eliciting and analyzing requirements in a large-scale research project. In particular, we identified and discuss the challenges to be faced in large-scale requirements engineering. We also showed why none of the existing RE methodologies cover all the activities required by large-scale research projects. To this end, we investigated alternative research proposals and industry best practices and integrated them into our own RE process. We presented the results of the application of this approach to TAS^3 and discussed the lessons we learned.

The TAS^3 project was originally conceived as an architecture-driven project. This approach together with the shallow description of RE activities in the DoW had an impact on the proper execution of the RE activities. First, project partners preferred to focus on their own activities within the project. This was confirmed, for instance, by the resistance of many partners to participate in the RE activities. In addition, the classical flow of engineering projects was ignored.

The carelessness in performing RE activities can lead to severe problems in the project. In particular, we have seen in TAS^3 that the omission of or oblivion toward RE can result in inconsistencies, out-of-scope developments, and, in the worst case, in major revisions to the software design. For instance, there were inconsistencies among the components designed by the different partners and some requirements were neglected in the design of the architecture. These were first discovered when the architecture team mapped the requirements to the architecture. In this case, the architecture team pointed out that some requirements were out of the scope of the architecture. It is worth noting that since the design of the architecture had been completed, accommodating such requirements would have required a (partial) redesign of the architecture. Additional evidence of these problems was provided by the EU Commission who argued that *"after the first year of work, the project still lacks an "integrated" common vision and a clear mission"* and was concerned about *"the lack of a consistent working approach and software engineering methodologies"*.

RE activities can help a project consortium to address these recommendations. Different from other activities that focus on specific workpackages, RE activities span across all workpackages. The RE team has to interact with all project partners and assist them in the elicitation, elaboration and analysis of requirements; in doing that, it can facilitate the communication between partners for integrating their contributions consistently. In addition, the iterative nature of the RE process makes it possible to assess the progress in the project. Therefore, RE activities have the potential of building an integrated common vision among project partners and evaluating project progress.

During the first review, the EU Commission also suggested to the TAS^3 consortium to *"implement best practices for project coordination and management"* and to *"reinforce the capability of the project coordination, in terms of managing the scientific and technical coordination in the project"*. We argue that the technical management of the project can take advantages of RE activities to organize and integrate project activities. In our case, the TAS^3 consortium adopted a new clustering approach for project management that was based on the results of the interaction analysis presented in this paper.

Applying and integrating existing requirements engineering methodologies in a large research project is a socio-technical problem. The selection of methodologies depends on their appropriateness for the RE needs of the project, the expertise of the project partners and the willingness of the partners to participate in the RE process. Our analysis shows that the careful planning of the RE process, the communication and the negotiation of these plans with the project partners, as well as a revision of the plans during the RE process points to the necessity of taking an iterative approach to the RE process during large research projects.

This paper analyzes the potentials and limitations of applying existing methodologies to large research projects. In that sense, it serves as an empirical evaluation of the appropriateness of applying these methods to a project. The lessons learned provide recommendations for applying these methods in the future. Further, in our analysis we also indicated some important future research needs.

Specifically, further studies are needed in developing gap analysis methods to be applied during requirements engineering, methods for evaluating and ensuring quality during RE activities, approaches for inter-domain requirements analysis and an evaluation of the use of graph analysis in collaborative requirements interaction management.

We can conclude that RE is a critical phase for the successful achievement of project objectives. This phase should identify the problems that the developed system is supposed to solve, bringing together the needs of all project partners. RE can provide the basis upon which an integrated common vision of the project within the consortium can be established and, in general, bestow a large and distributed project with a central reference point. However, a critical factor for the success of RE activities is to engage all project partners in their execution. In large-scale research projects, the pressure of producing tangible results is immense. As long as RE activities are not seen as a tangible outcome in a project, they are likely to be neglected in favor of project activities with tangible (software) products. Therefore, greater emphasis should be given to framing RE activities and the results from these activities as a desirable and valuable outcome of large research projects.

Acknowledgments The authors thank the TAS^3 partners and in particular Gilles Montagnon for their valuable and critical comments. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n 216287 (TAS3 -Trusted Architecture for Securely Shared Services).

References

1. Alves C, Finkelstein A (2002) Challenges in cots decision-making: a goal-driven requirements engineering perspective. In: Proceedings of 14th international conference on Software Engineering and Knowledge Engineering. ACM, New York, pp 789–794
2. Andriole SJ (1989) Storyboard prototyping: a new approach to user requirements analysis. QED Information Sciences Inc., Wellesley
3. Asnar Y, Moretti R, Sebastianis M, Zannone N (2008) Risk as dependability metrics for the evaluation of business solutions: a model-driven approach. In: Proceedings of the 3rd international conference on availability, reliability and security. IEEE Computer Society, Washington, pp 1240–1248
4. Bergman M, King JL, Lyytinen K (2002) Large-scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering. Requir Eng 7(3):152–171
5. Blomberg J, Giacomi J, Mosher A, Swenton-Wall P (1993) Ethnographic field methods and their relation to design. In: Participatory design: principles and practices. Erlbaum, London, pp 123–155
6. Breaux TD, Antón AI (2008) Analyzing regulatory rules for privacy and security requirements. IEEE Trans Softw Eng 34(1):5–20
7. Bresciani P, Giorgini P, Giunchiglia F, Mylopoulos J, Perini A (2004) TROPOS: an agent-oriented software development methodology. Auton Agents Multi-Agent Syst 8(3):203–236
8. Caplan S (1990) Using focus group methodology for ergonomic design. Ergonomics 33(5):527–533
9. Carlshamre P, Sandahl K, Lindvall M, Regnell B, Dag JN (2001) An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of the 5th IEEE international symposium on requirements engineering. IEEE Computer Society, Washington, pp 84–93
10. Chen D, Doumeingts G (2003) European initiatives to develop interoperability of enterprise european initiatives to develop interoperability of enterprise applications-basic concepts, framework and roadmap. Annu Rev Control 27:153–162
11. Coughlan J, Lycett M, Macredie RD (2003) Communication issues in requirements elicitation: a content analysis of stakeholder experiences. Inf Softw Technol 45(8):525–537
12. Dahlstedt Å, Persson A (2005) Requirements interdependencies: state of the art and future challenges. In: Engineering and managing software requirements. Springer, New York, pp 95–116
13. Damian DE (2007) Stakeholders in global requirements engineering: lessons learned from practice. IEEE Softw 24(2):21–27
14. Damian DE, Zowghi D (2003) Requirements engineering challenges in multi-software development organizations. Requir Eng 8(3):149–160
15. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20:3–50
16. Davis A, Dieste O, Hickey A, Juristo N, Moreno AM (2006) Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review. In: Proceedings of the 14th IEEE international requirements engineering conference. IEEE Computer Society, Washington, pp 176–185
17. Decker B, Ras E, Rech J, Jaubert P, Rieth M (2007) Wiki-based stakeholder participation in requirements engineering. IEEE Softw 24(2):28–35
18. Elahi G, Yu E, Zannone N (2010) A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. Requir Eng 15(1):41–62
19. Fabian B, Gürses SF, Heisel M, Santen T, Schmidt H (2010) A comparison of security requirements engineering methods. Requir Eng 15(1):7–40
20. Fitz-Gibbon CT (1990) Performance indicators. Multilingual Matters, Clevedon
21. Fowler FJ, Mangione TW (1990) Standardized survey interviewing: minimizing interviewer-related error. Applied social research vol 18. Sage, Newbury Park
22. Garlan D, Schmerl B (2006) Architecture-driven modelling and analysis. In: Proceedings of the 11th Australian workshop on safety critical systems and software. Australian Computer Society Inc, Darlinghurst, pp 3–17
23. Gervasi V, Nuseibeh B (2002) Lightweight validation of natural language requirements. Softw Pract Exp 32:113–133
24. Giorgini P, Massacci F, Mylopoulos J, Zannone N (2006) Requirements engineering for trust management: model, methodology, and reasoning. Int J Inf Secur 5(4):257–274
25. Goguen JA, Linde C (1993) Techniques for requirements elicitation. In: Proceedings of IEEE international symposium on requirements engineering. IEEE Computer Society, New York, pp 152–164

26. Good M (1989) Seven experiences with contextual field research. SIGCHI Bull 20(4):25–32
27. Guarda P, Zannone N (2009) Towards the development of privacy-aware systems. Inf Softw Technol 51(2):337–350
28. Gürses S, Jahnke JH, Obry C, Onabajo A, Santen T, Price M (2005) Eliciting confidentiality requirements in practice. In: Proceedings of the 2005 conference of the centre for advanced studies on collaborative research. IBM Press, Canada, pp 101–116
29. Gürses S, Zannone N (2009) Requirements assessment report. Research report D1.2, TAS^3 consortium
30. Hofmann HF, Lehner F (2001) Requirements engineering as a success factor in software projects. IEEE Softw 18(4):58–66
31. Hull E, Jackson K, Dick J (2005) Requirements engineering, 2nd edn. Springer, New York
32. Isensee S, Rudd JR, Heck M (1995) Art of rapid prototyping: user interface design for windows and OS/2. International Thomson Computer Press, New York
33. ISO (2008) Gap analysis tool. ISO 9001:2008
34. Jackson M (2001) Problem frames: analysing and structuring software development problems. Addison Wesley, New York
35. Jureta IJ, Mylopoulos J, Faulkner S (2009) A core ontology for requirements. Appl Ontol 4(3–4):169–244
36. Jürjens J (2004) Secure systems development with UML. Springer, New York
37. Kiyavitskaya N, Zannone N (2008) Requirements model generation to support requirements elicitation: the secure tropos experience. Autom Softw Eng 15(2):149–173
38. Konrad S, Gall M (2008) Requirements engineering in the development of large-scale systems. In: Proceedings of the 16th IEEE international requirements engineering conference. IEEE Computer Society, New York, pp 217–222
39. Kontio J, Chen SF, Limperos K, Tesoriero R, Caldiera G, Deutsch M (1995) A cots selection method and experiences of its use. In: Proceedings of 20th annual software engineering workshop
40. van Lamsweerde A (2004) Elaborating security requirements by construction of intentional anti-models. In: Proceedings of the 26th international conference on software engineering. IEEE Computer Society, New York, pp 148–157
41. Liu L, Yu E, Mylopoulos J (2003) Security and privacy requirements analysis within a social setting. In: Proceedings of the 11th IEEE international conference on requirements engineering. IEEE Computer Society, New York, pp 151–161
42. Lund MS, Solhaug B, Stølen K (2010) Model-driven risk analysis: the CORAS approach. Springer, New York
43. Marczak S, Damian D, Stege U, Schröter A (2008) Information brokers in requirement-dependency social networks. In: Proceedings of the 16th IEEE international requirements engineering conference. IEEE Computer Society, New York, pp 53–62
44. Massacci F, Mylopoulos J, Zannone N (2010) Security requirements engineering: the SI* modeling language and the secure tropos methodology. In: Advances in intelligent information systems. Springer, New York, pp 147–174
45. Massacci F, Prest M, Zannone N (2005) Using a security requirements engineering methodology in practice: the compliance with the Italian data protection legislation. Comput Stand Interfaces 27(5):445–455
46. Massacci F, Zannone N (2011) Detecting conflicts between functional and security requirements with secure tropos: John Rusnak and the Allied Irish Bank. In: Social modeling for requirements engineering. MIT Press, Cambridge
47. Maulsby D, Greenberg S, Mander R (1993) Prototyping an intelligent agent through Wizard of Oz. In: Proceedings of the IFIP TC13 international conference on human-computer interaction. ACM, Switzerland, pp 277–284
48. Montagnon G (2009) Design requirements. Research report D1.4, TAS^3 consortium
49. Nielsen, J, Mack, RL (eds) (1994) Usability inspection methods. Wiley, New York
50. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the conference on the future of software engineering. ACM, New York, pp 35–46
51. Nuseibeh B, Kramer J, Finkelstein A (1994) A framework for expressing the relationships between multiple views in requirements specification. IEEE Trans Softw Eng 20(10):760–773
52. Object Management Group (2009) Unified modeling language (UML). http://www.omg.org/spec/UML/2.2/
53. Parmenter D (2010) Key performance indicators (KPI): developing, implementing, and using winning KPIs, 2nd edn. Wiley, New York
54. Petersen K, Wohlin C, Baca D (2009) The waterfall model in large-scale development. In: Proceedings of the 10th international conference on product-focused software process improvement, lecture notes in business information processing, vol 32. Springer, New York, pp 386–400
55. Potts C, Takahashi K, Antón AI (1994) Inquiry-based requirements analysis. IEEE Softw 11(2):21–32
56. Przybilski M (2006) Requirements elicitation in international research projects. In: Proceedings of the 12th Americas conference on information systems
57. Regnell B, Svensson RB, Wnuk K (2008) Can we beat the complexity of very large-scale requirements engineering? In: Proceedings of the 14th international conference on requirements engineering: foundation for software quality, LNCS 5025. Springer, New York, pp 123–128
58. Reinhard T, Meier S, Stoiber R, Cramer C, Glinz M (2008) Tool support for the navigation in graphical models. In: Proceedings of the 30th international conference on software engineering. ACM, New York, pp 823–826
59. Rettig M (1994) Prototyping for tiny fingers. Commun ACM 37(4):21–27
60. Robertson S, Robertson J (2006) Mastering the requirements process. 2nd edn. Addison-Wesley Professional, New York
61. Robinson WN, Pawlowski SD, Volkov V (2003) Requirements interaction management. ACM Comput Surv 35(2):132–190
62. Sampaio do Prado Leite JC, Hadad GDS, Doorn JH, Kaplan GN (2000) A scenario construction process. Requir Eng 5(1):38–61
63. Sindre G, Opdahl AL (2005) Eliciting security requirements with misuse cases. Requir Eng 10(1):34–44
64. Sommerville I, Sawyer P (1997) Viewpoints: principles, problems and a practical approach to requirements engineering. Ann Softw Eng 3:101–130
65. Toro AD, Jiménez BB, Cortés AR, Bonilla MT (1999) A requirements elicitation approach based in templates and patterns. In: Proceedings of workshop em engenharia de requisitos, pp 17–29
66. Wiegers KE (2006) More about software requirements: thorny issues and practical advice, 2nd edn. Microsoft Press, Redmond
67. Wnuk K, Regnell B, Karlsson L (2009) What happened to our features? Visualization and understanding of scope change dynamics in a large-scale industrial setting. In: Proceedings of the 17th IEEE international requirements engineering. IEEE Computer Society, New York, pp 89–98
68. Wnuk K, Regnell B, Schrewelius C (2009) Architecting and coordinating thousands of requirements—an industrial case study. In: Proceedings of the 15th international working conference on requirements engineering: foundation for software quality, LNCS 5512. Springer, New York, pp 118–123
69. Zave P (1997) Classification of research efforts in requirements engineering. ACM Comput Surv 29(4):315–321
70. Zave P, Jackson M (1997) Four dark corners of requirements engineering. ACM Trans Softw Eng Methodol 6(1):1–30