Information and Software Technology xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

The practical application of a process for eliciting and designing security in web service systems

Carlos Gutiérrez [a,b], David G. Rosado [a], Eduardo Fernández-Medina [a,*]

[a] Department of Information Technologies and Systems, ALARCOS Research Group – Institute of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, Spain
[b] Correos Telecom, Madrid, Spain

ARTICLE INFO

Article history:
Received 13 October 2008
Received in revised form 9 March 2009
Accepted 13 May 2009
Available online xxxx

Keywords:
Application information security
Software process
Web Services Security
Design methodology

ABSTRACT

Best practices currently state that the security requirements and security architectures of distributed software-intensive systems should be based on security risk assessments, which have been designed from security patterns, are implemented in security standards and are tool-supported throughout their development life-cycle. Web service-based information systems uphold inter-enterprise relations through the Internet, and this technology has been revealed as the reference solution with which to implement Service-Oriented Architectures. In this paper, we present the application of the Process for Web Service Security (PWSSec), developed by the authors, to a real web service-based case study. The manner in which security in inter-organizational information systems can be analyzed, designed and implemented by applying PWSSec, which combines a risk analysis and management, along with a security architecture and a standard-based approach, is also shown. We additionally present a tool built to provide support to the PWSSec process.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Web services (hereafter WS) is the reference technology used to implement Service-Oriented Architecture (SOA)-based information systems. It has attained such a relevance in both the academic and industrial world that the vision of the Internet is evolving, and is no longer considered to be a mere data repository, but is becoming the underpinning infrastructure in which complex business processes and alliances are deployed [1].

Web service-based specifications, standards, technologies and tools have undergone a spectacular growth in the last few years. This evolution has been mainly motivated by their widespread adoption and promotion by the major vendors in the industry [2]. This overwhelming amount of new business and technological literature, together with the rapid evolution of technology and instability, has caused its target users to show a very reticent attitude towards its practical application in mission-critical software-intensive systems. The main rationale behind this is that the general approach has been based on a bottom-up approach, which lacks a development process with which to establish a general security framework and facilitate the systematic integration of security into all stages of the existing WS-based software development life-cycle. As Vinoski states in [3], until now, the approach followed by the main actors in WS (security) specifications has been driven by self interest, with a greater focus on filling the WS (security) stack for their own (commercial) purposes rather than providing a global architectural (WS-Architecture) solution.

PWSSec (Process for Web Services Security) [4] has been defined to fill this gap. PWSSec is intended to facilitate and guide the development of WS-based security systems in order to permit the easy integration of a complimentary stage comprising security [6] in each of the traditional stages of the construction of this type of systems [5]. The PWSSec process can be used once the WS functional architecture of the system has been built or during the stages in which the architecture is being defined. In both cases, the result will be a pattern-oriented security architecture formed of a set of coordinated security mechanisms that use the WS security standards to address the system security requirements. It is important to stress that PWSSec has been specifically defined to incorporate security in WS-based systems. This is made necessary by the distinguishing features of this technology and by the new security challenges and vulnerabilities it must face. Therefore, if at the beginning of the development it is detected that the system to be implemented will use technologies other than those which are WS-based, PWSSec is not applicable and other approaches and technical standards should be used.

PWSSec helps the developer in the task of incorporating security into Web Services-based systems. It accomplishes the following main stages of secure system development: risk-based security engineering, pattern-oriented security architecture and

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: carlos.gutierrez@correos.es (C. Gutiérrez), david.grosado@uclm.es (D.G. Rosado), eduardo.fdezmedina@uclm.es (E. Fernández-Medina).

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.05.004

standard-centered security design. The first subprocess, WSSecReq (Web Services Security Requirements), applies an application-centered approach which identifies the set of security threats to the system. It thus enables the definition of security requirements from these threats and refines them throughout the construction of a set of inter-related security artifacts: threats arranged in a tree-like structure. The threats are next refined as attack scenarios by specifying misuse cases gathered in security profiles; the impact and risks of attacks are then computed by applying an ISO-compliant 15408 risk analysis methodology. Sound countermeasures in the form of security use cases are identified, and finally security requirements based on a reusable approach are specified. The second subprocess is called WSSecArch (Web Service Security Architecture) and is aimed at allocating the security requirements specified in the WSSecReq subprocess to a WS-based security architecture. This security architecture will be equipped with the necessary security architectural mechanisms to achieve the identified security requirements. A case study demonstrating how this WS-based security architecture can be designed was presented in [6]. The principal objective of the third subprocess, named WSSecTech (Web Services Security Technologies), is to identify the set of WS-based security standards [7] that will implement the architectural security mechanisms identified in the WSSecArch subprocess.

This manuscript shows an actual case study in which PWSSec was applied in its entirety. This case study shows how a security critical use case was engineered from the security standpoint and by applying PWSSec. Although at first glance this may seem a very trivial case study (since only one use case appears), the use case developed covers all aspects of an actual security critical WS-based integration system, from which more than 100,000 lines of code were developed, requiring the joint work of two organizations.

The remainder of the paper is organized as follows: Section 2 presents work related to our approach; Section 3 provides the background to SOA security; in Section 4, the case study developed in this paper is described; Section 5 shows an overview of the PWSSec process; in Section 6, a tool developed by the authors, whose main purpose is to assist developers in their application of the PWSSec process, is briefly presented; the application of the PWSSec's WSSecReq, WSSecArch and WSSecTech subprocesses to the case study is developed in Section 7; in Section 8, lessons learned during the case study are presented; in Section 9 the research method followed to define PWSSec is explained; in Section 10, final conclusions and future research are described, and finally, Appendix I gathers together the set of acronyms used throughout the paper.

2. Related work

With regard to the definition of processes for WS secure system development, we can highlight the extension of the agent and goal-oriented methodology Tropos, defined in [8]. This work describes an adaptation of Tropos which permits the definition of an architecture that covers a certain set of WS-based Quality-of-Service requirements. EFSOC [9] is an event-driven framework for WS-based systems which specifies a security model that can be easily adopted by systems in which the degree of modifiability is very high, thereby requiring the continuous review and updating of authorization policies. In [10], a formal analysis of security-critical service-based software systems is presented and in [11] a formal approach to the construction of service-based systems is described.

In [12] an MDA approach to define a Web Services Security Architecture is explained. This approach does not provide a specific and systematic method with which to produce software security requirements; however, as will be shown later, the PWSSec process does provide software developers with such a method. In [13], Nagaratnam et al. explain a process which considers the business-application life cycle and proposes a policy-driven approach overlaid on a model-driven paradigm for addressing security requirements. This work emphasizes security deployment and administration as complementary activities to be considered in addition to model, design and implementation. Although security engineering and risk analysis and management are mentioned, a formal approach to security requirements engineering is not proposed.

Breu et al. [14,15] define another method for integrating security into WS-based systems. PWSSec is highly aligned with the work of Breu et al., although there are important differences between both approaches: PWSSec defines a detailed security requirement engineering process, permitting its straightforward practical application in real developments, while Breu et al.'s work is a model-driven oriented approach offering a systematic approach for incorporating security at all levels of abstraction of systems modeling (e.g.: business, application, technical level). PWSSec defines software engineering-oriented security architectural views which adhere to the IEEE 1471-2000 standard [16] while the work of Breu et al. provides a security model-based view of the workflow between the integrated organizations.

Fernandez et al. [17,18], have recently proposed a pattern-centered security methodology for the development of SOA systems. This approach is aligned to ours but does not provide specific security artifacts that facilitate the integration of security aspects during the security requirements engineering stage. Neither does it provide traceability between security requirements and security architecture or a security reference architecture from which a WS-based approach can be designed.

None of the previously discussed approaches propose a method similar to that of PWSSec which allows a WS-based secure system to be designed from the business goals, system security goals and functional business and application WS patterns. Moreover, none of these methods offer facilities for the reusability of the generated products thereby guaranteeing the practical applicability of PWSSec. With regard to other tool-supported approaches, no other software engineering-based tools have been identified. In addition, the main purpose of this work is to present the case study and to show how the PWSSec process was supported by our tool. This tool specifically implements the activities, tasks, security artefacts defined in the WSSecReq subprocess.

This paper presents the application of this process to a real case study. The paper mainly focuses on how the WS-specific security requirements have been elicited and how this elicitation has been supported thanks to the tool developed by the authors. The objective of the system-to-be-constructed was the sale of products which were selected by their purchasers through a Web application. Payments were made from the purchasers' bank account which was associated with the bank account of the sales organization. The application of PWSSec to this case study, and its supporting tool, allowed us to both demonstrate its practical applicability and to find new sources for its enhancement.

3. SOA security background

The present-day business environment demands optimized efficiency and rapid adaptability to its continuous changes. New channels, new competitors, new technologies and globalization are forcing companies to learn to see change as the only permanent state of their businesses [19]. Said companies have discovered that the IT function is paramount to attaining this objective, and have in particular discovered that Service Oriented Architectures actually promote the achievement of these goals. There has consequently been an increase in the adoption of Service Oriented Architectures, both in industry and academia, and of its main implementation technology: Web Services technology. The security challenges presented by the Web services approach are highly complex and technologically advanced [20]. In summary, the new security challenges which have arisen from this new paradigm are [21]:

- Risks that appear as a result of the publication on the Internet of a complete and well-documented interface to back office data and company's business logic. One of the main security problems associated with the adoption of WS is derived from the Internet publication of business interfaces through ports HTTP 80 or HTTPS 443 [22].
- Protecting the semantic Web by "ensuring that security is preserved at the semantic level" [23].
- Context-aware and context-based protection at the document level [24,25]. Documents usually have information with different "degrees of sensitivity" which are required to be protected at different levels of security. Access control policies that govern access to the different security parts of the documents and an architecture enforcing these policies currently constitute an extremely important research area in the context of WS security.
- Service trustworthiness. Dynamic discovery and the composition of services imply that a Web service consumer may not know whether the services, individually or in composition, will behave as expected. "Consequently, how to select trustworthy Web services remains a challenge" [26].

Some of its main security objectives are:

- Management of security policies in a large distributed WS environment [27].
- Application-level, end-to-end and just-one-context-security communications. Network topologies require end-to-end security to be maintained in all the intermediaries in the message's path. When data is received and forwarded on by an intermediary beyond the transport layer, both the data integrity and any security information that flows with it may be lost [28].
- Interoperability of the requirements and on-line security elements [29].
- Ability to federate the full information concerning the subjects, thus permitting single sign-on environments and facilitating across-enterprise interoperability [30].
- Maintaining sensitive users' attributes and identity private in trust domains.

Apart from these security challenges, the unstructured and overwhelming number of WS security related literature and approaches make the developers' task of attaining a complete knowledge of all the potential WS security issues, and the standard means to address them, extremely difficult. PWSSec organizes this knowledge and offers a systematic, practical and complete guide through which to put this into practice when developing WS based systems.

4. Bank transfer system case study

This paper presents a case study that was applied to a Web service-based system called WS-BTS (Web service-based Bank Transfer System). The objective was the sale of certain products which were selected by the purchasers through a Web application. Payments were made from the purchaser's bank account which was associated with the bank account of the sales organization.

Fig. 1 shows a general diagram of the system. As we can see, it is a WS-based system and consists of at least two different WS agents: (1) a WS consumer agent, belonging to the sales organization, which will be referred to as WS-BTSConsumer (Web services-based Bank Transfer System Consumer) and (2) the bank service's WS provider agent, which will be referred to as WS-BTSProvider (Web services-based Bank Transfer System Provider). According to W3C Web Services Reference Architecture [31], Web services based systems are composed of Web service agents which interact to fulfill a task, and which can be defined as *"a concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided*". In our case study, these agents interact in order to fulfill a business workflow, whose objective is to assist the final customer during payment and to facilitate the final purchase.

The use case implemented in the case study is called Performing Bank Transfer (PBT), and is shown in Fig. 2.

This use case follows a three-step protocol:

1. When the Web application receives the sales order from the purchaser, which includes the identity of the bank through which s/he wishes to make the payment, it solicits the creation of a security token from the WS-BTSConsumer and sends it via

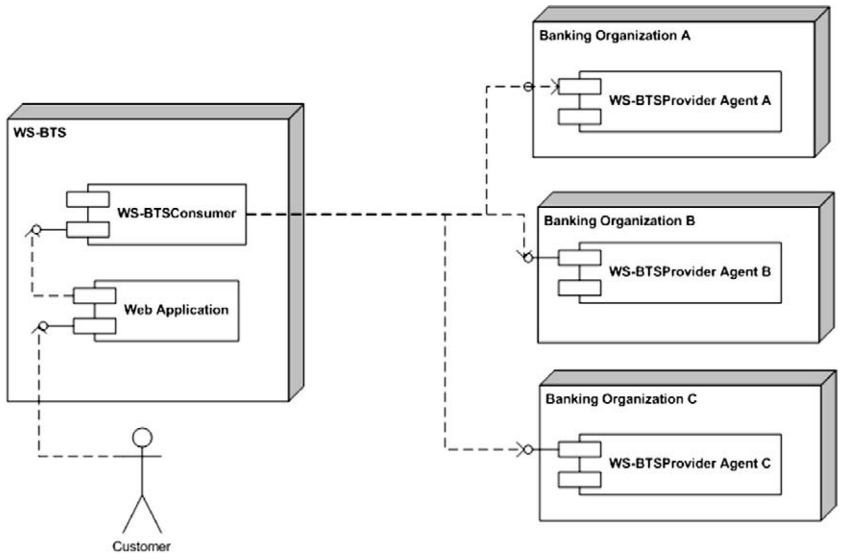


Fig. 1. Overview of the WS-BTS system.

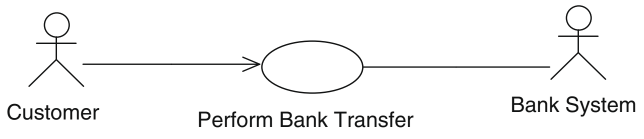


Fig. 2. Use case 'Perform Bank Transfer'.

the Internet to the WS-BTSProvider agent located at the solicited bank organization. The user's browser is then redirected to the bank's Website. It is computationally impossible to duplicate the security token, which is primarily created from product-related information, including information identifying the purchaser in the bank system (there is a network identity federation based on pseudonyms between the Web sales domain and those of different banks [30]) in more than one transaction. The objective of this interaction is to signal to the WS-BTSProvider agent the desire to complete a transfer in order to pay for a certain product. Hereafter, we shall refer to this interaction as 'Token for New Transfer (TNT)'.

2. The banking system of the entity which receives the transfer request from the WS-BTSProvider agent, will then proceed to authenticate the buyer. Once authorized to make the transfer, it will request that the WS-BTSProvider agent re-solicits the token from the WS-BTSConsumer service which will eventually return it. This interaction is denominated as 'Transfer Token Request (TTR)'.
3. Once the token is received and has been verified as being the original token first sent from the Web application, the WS-BTS-Provider agent finalizes the transfer and generates a unique reference number, which cannot be duplicated in other transfers (not only within the same bank but also within other banking organizations), which is then sent to the WS-BTSConsumer. The client verifies this number and sends a confirmation message to complete the transfer. This step is referred to as the 'Reference Number Processing (RNP)' interaction.

Although it may seem at first glance that the use case presented is rather simple, in reality it entailed more than 100,000 lines of code to be developed, more than 30 database tables, more than 50 problem domain classes, and 3 months of development and testing with the first integrated banking organization. This chapter only shows the bird's eye view of the functional solution, but it is important to bear in mind that developing both Web Services Agents and implementing the three-step interaction (and all the performance controls derived from them) yielded not only two low-level classes but also more than 50 problem domain classes (not taking into account those derived from the addition of design patterns to the design model). A Web application, which is not presented in the case study, but which provides the necessary functionality to manage integrated third-parties (banking organizations) and control the integration channel (incident management, payments reconciliations, etc.) of each of them has also to be developed.

The original development of the information system supporting this workflow was designed at the beginning of 2001. At that time, Web service security standards and technologies were still somewhat undeveloped. Despite the fact that a considerable amount of security standards for Web services have been developed and have matured since then, the approach followed has been bottom-up, starting from the standards and moving towards architecture and finally towards requirements. In reality the main security mechanisms are already standardized, but there is still a clear gap between those mechanisms, and the definition of a global security architecture to which they can be allocated and traced to the security requirements of origin.

This lack of experience and maturity led us to opt for a tailor-made solution based on the exchange of XML documents concerning HTTPS (an architectural-style known as REST [32]).

Briefly, the security requirements intended to be addressed by this system were:

- Mutual authentication, permitting identity verification between partners.
- Integrity, allowing the detection of alterations in the exchanged messages.
- Confidentiality, guaranteeing that the exchanged information was only visible to the receiving partner.

The most difficult challenge when facing the design of this system was its low interoperability and integrity, due to its tailor-made nature which lacked an underlying standardized-based infrastructure. Integration processes with bank systems proved to be very costly and required considerable (human) interaction. It was consequentially desirable to equip the system with a standard-based approach requiring minimal interaction between the sales organization and the banking institution. A list of WS-based (security) standards and a specified workflow using a standardized language (e.g.: BPEL), would therefore enable the banking organization to independently develop and test its own WS-BTSProvider agent.

The scope of the application of the PWSSec process to this system was limited in order to secure each interaction between the Web service agents, regardless of whether the whole protocol was secure or not (we assumed that it was). The application of PWSSec to this case study had two main objectives: migrating to a Web service-based technology, and verifying that it was possible to systematically specify a set of WS-based security requirements, a WS-based security architecture and a set of WS-based security standards which would discover and address the main security concerns, define a coordinated security mechanism-based solution, and "standardize" the aforementioned interactions (that is, the system). Thus, there was a clear opportunity to apply this process to a real case study which would serve as a proof-of-concept for the PWSSec process.

5. PWSSEC overview

PWSSec [4] has been created to facilitate and guide the development of WS-based security systems thus permitting the easy integration of a complementary subprocess comprising security into each of the traditional subprocesses for the construction of this kind of systems [5,33].

Fig. 3 illustrates the set of subprocesses that PWSSec defines. Each subprocess describes its own inputs, outputs, activities, actors and, in some cases, guides, tools and techniques, thereby complementing, improving and facilitating its practical application.

The main purpose of the WSSecReq (Web Service Security Requirements) process is to produce a specification (or a part of it) of the security requirements of the WS-based system by means of a systematic approach. A more detailed explanation of this subprocess was presented in [4], and its application to the case study is shown later in Section 7.1. This subprocess's main outputs are the System Requirement Specification (SyRS), System Requirement Test Specification (SyTS), Software Requirement Specification (SRS), Software Requirement Test Specification (STS) and Interface Requirement Specification (IRS).

The aim of the WSSecArch (Web Services Security Architecture) subprocess is to allocate the security requirements specified in the previous section to a WS-based security architecture. This security architecture will be equipped with the necessary security architec-

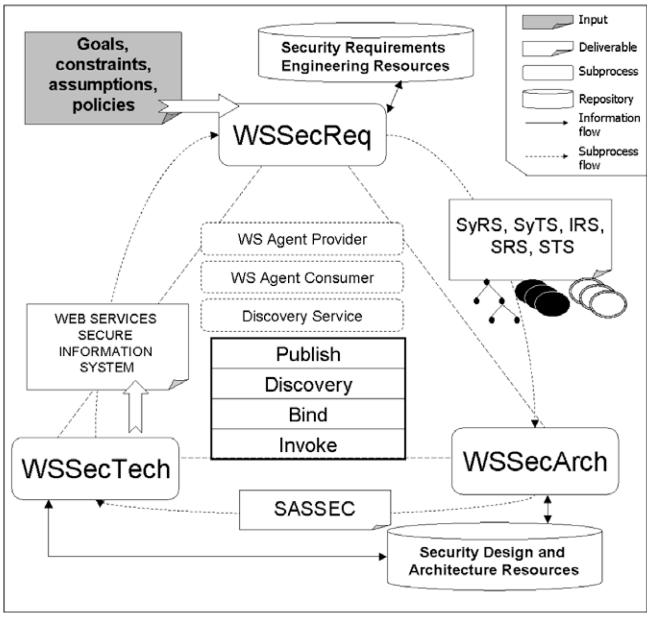

Goals, constraints, assumptions, policies
Security Requirements Engineering Resources
Input
Deliverable
Subprocess
Repository
Information flow
Subprocess flow
WSSecReq
SyRS, SyTS, IRS, SRS, STS
WS Agent Provider
WS Agent Consumer
Discovery Service
WEB SERVICES SECURE INFORMATION SYSTEM
Publish
Discovery
Bind
Invoke
WSSecTech
WSSecArch
SASSEC
Security Design and Architecture Resources

Fig. 3. Subprocesses and main security artifacts of the PWSSec process.

tural mechanisms to achieve the identified security requirements. A case study demonstrating how this WS-based security architecture can be designed was presented in [6], and its application to the case study is shown later in Section 7.2.

The main objective of the WSSecTech (Web Services Security Technologies) subprocess is to identify the set of WS-based security standards that will implement the architectural security mechanisms identified in the previous subprocess, and its application to the case study is shown later in Section 7.3.

As mentioned above, PWSSec subprocesses can easily be integrated into typical SOA development methodologies, as is stated in [5]. Table 1 explains the steps involved in this process, and how PWSSec subprocesses could be integrated.

6. A supporting tool for PWSSEC

The PWSSec process defines a set of numerous security elements (subprocesses, activities, tasks, artifacts and deliverables). Before applying the PWSSec process to the case study, and while PWSSec was being defined, a support tool was developed in parallel.

Since PWSSec is based on UML, an existing CASE tool was extended and we chose Rational Rose 2000. Two main reasons lay behind this decision: Firstly, Rational Rose is a professionally wide-spread and renowned CASE tool and secondly, it provides an easy-to-use and extensibility API (Application Programming Interface) called "Rose Extensibility Interface" (REI).

Extending Rational Rose implied the use of a programming language that allowed us to work with type libraries, since REI is provided by default in the form of the "rationalrose.tlb" component. Although this library can be used by different programming languages, Visual Basic 6 was required if we were to build an add-on. The tool developed was bound to the Rational Rose application in the form of an add-on and was thus valid for Windows 9x/ME / XP operating systems. The REI interface not only allows access to Rational Rose UML model objects and elements, but also to their modification.

The final output of the tool is a *dll ActiveX* library, which is bound by Rational Rose to work on the created UML models. The tool is integrated into Rational Rose through its Extension Manager facility, which is accessible from the menu of the main window *Tools → Edit → Requirement Elicitation*.

Section 7 shows how this tool may have been of assistance in the execution of the tasks defined in the WSSecReq subprocess requirements elicitation activity.

7. Application of PWSSEC

This section explains how each PWSSec subprocess has been applied to the case study previously described. At the outset, we began our work by using a UML model of the system (see Fig. 4).

The UMLPWSSec (UML for PWSSec) tool used this model to initiate the execution of the wizard which guides the developer throughout the tasks that constitute the WSSecReq subprocess elicitation activity.

PWSSec is an iterative and incremental process [34] and, as such, a correct planning of the iterations and increments that should be developed is necessary. In this case study, three iterations were applied, and the development of the first iteration will be described in this paper. The main aim of this phase was to produce the majority of the security requirements and to outline the candidate security architecture (by basically identifying the security architectural patterns and the potential security design patterns).

Table 1
How PWSSec's subprocesses can be integrated into a typical SOA development methodology.

IBM SOA development Step	Description	PWSSec	Description
Domain decomposition	The domain is decomposed into its constituent business architecture consisting of value-chain, business processes, sub-processes, and use cases	WSSecReq	Takes as inputs the service model, the business use cases and system use case model (at this point both the IBM WS business and application pattern have been identified) produced in previous steps and yields as main final output the corresponding set of security requirements
Goal-service model creation	Discovery of business aligned services for the entire organization		
Subsystem analysis	Subsystem analysis refines the business use cases into system use cases that support a given business process		
Service allocation	Ensure that all the services identified have a "home" and that they are all traceable back to business goals and to components	WSSecArch	Once, and while, the services are being identified and the system architecture is being defined, the security-side of the architecture has to specified
Component specification	Develop specifications for each of the components needed		
Structure components and services using runtime patterns	Structure these components and services by using IBM runtime patterns	WSSecTech	WS security mechanisms and WS security standards are identified and specified in conjunction at the same time as technology realization mapping is performed
Technology realization mapping	The choice of how to realize the implementation of the specification is the following a key architectural		

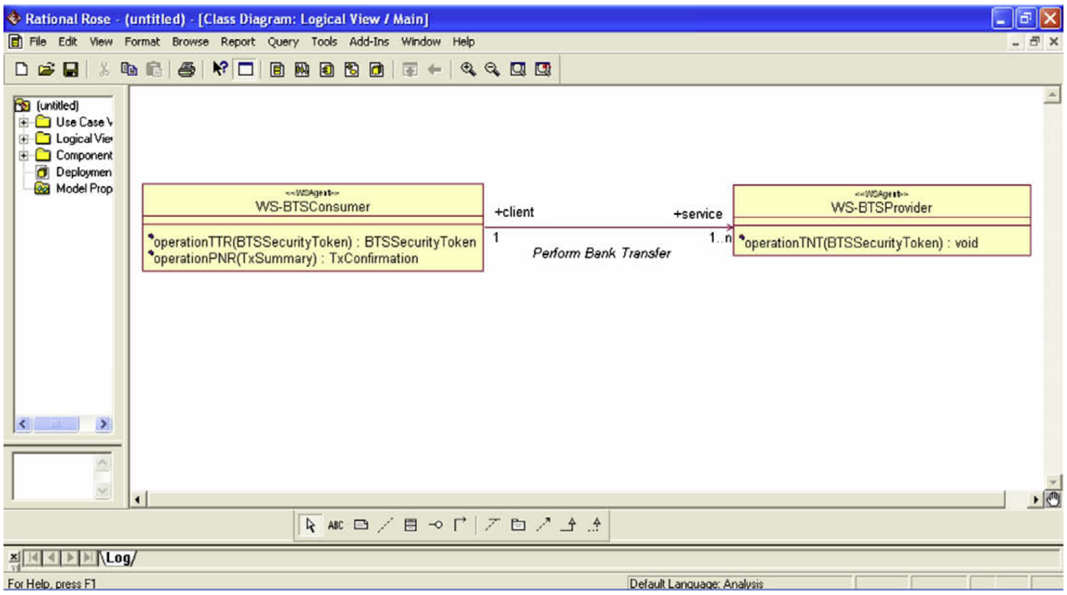

Rational Rose - (untitled) - [Class Diagram: Logical View / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
(untitled)
Use Case V
Logical Vie
Component
Deploymen
Model Prop
<<WSAgent>>
WS-BTSConsumer
operationTTR(BTSSecurityToken) : BTSSecurityToken
operationPNR(TxSummary) : TxConfirmation
+client
1
+service
1..n
Perform Bank Transfer
<<WSAgent>>
WS-BTSProvider
operationTNT(BTSSecurityToken) : void
Log
For Help, press F1
Default Language: Analysis

Fig. 4. Initial UML model of the WS-BTS system.

The second iteration (omitted) was mainly focused on the creation of the final security architecture, along with the application of a set of security design patterns. This second iteration, with regard to the WSSecTech subprocess, was used to define a list of candidate security Web services-based standards. These standards fulfilled the security services identified in the security architecture.

In the third iteration, the security architecture was reviewed, the security design patterns were detailed and the security services that were not provided by the existing security infrastructure were implemented. In addition, the final list of security standards/specifications was refined and these were implemented in the context of the problem.

7.1. WSSECREQ applied to the case study

This section presents all of the activities (see Fig. 6) defined in the WSSecReq subprocess as they were applied to the case study.

As was previously mentioned, this subprocess follows a security risk management approach relying, in this case study, on the security risk management and control methodology named MAGERIT2 [35]. MAGERIT2, is the officially recognized risk management methodology of NATO [36] and OECD [37]. This methodology has been officially adopted by the Spanish public administration, the environment in which this case study was developed.

The inputs of the WSSecReq subprocess are (see Fig. 5):

- **Functional Scope Statement, Assumptions and Restrictions**. In this case, our aim was to elicit the security requirements derived from analyzing security in the TNT, TTR and RNP interactions.
- **Business Goals Statement**. We focused mainly on just one business goal: that of obtaining a standard-based technical solution that would implement the existing workflow and that would be implemented by proprietary vendors' products (versus open source solutions), with the intention that this solution would allow banking organizations to work as independently as possible.
- **Business Security Goals and Policy Statement**. The main security goal was to ensure that the interactions were deployed in a secure form thus ensuring that the parties involved did not face any risks from potential fraudulent actions originating from either internal or external attackers.

7.1.1. WSSecReq – Elicitation Activity

A1.1. Elicitation – T1.1.1: Decide granularity level and identify the fragment of functional software whose security will be analyzed.

Functional scope was provided in the form of a use case model. The use case *Perform Bank Transfer* is shown in Fig. 2. This use case accomplishes the three interactions previously explained in Section 2.

The output of this task was the **System-level Use Case Model,** which included the description of all the use cases and actors involved [38]. The RE (Requirement Engineering) Group was responsible for defining the use case model.

A1.1. Elicitation – T1.1.2: Identify the IBM WS-based business pattern.

In this task, the IBM WS-based business pattern followed by the use case 'Perform Bank Transfer' was identified. This pattern's definition is stored in the *E&A Repository – IBM's WS-based Patterns' Catalogue*. The business pattern that fitted the use case is called Extended Enterprise (also known as the Business-to-Business pattern) [5]. We used the template specified in the repository to document the application of this pattern in our system. This task's output is shown in Table 2.

The instantiation of the business-level threat tree shown in Fig. 9 was developed in task T1.1.4, and its description was completed through the use of steps 1 and 2 of the tool's wizard, as is shown in Figs. 7a and 7b.

The description of the application of the business pattern shown in Table 2 is one of the two outputs defined in task T1.1.2. The second output is the **Extended Enterprise Security Policy** [39]. In this case study, a template of this policy was defined and was then customized according to the agreement reached with each banking organization, and developed jointly between the Expert Domain Group and the Security Group. Both groups were formed of members of the sales organization and the first banking organization integrated into the system.

A1.1. Elicitation – T1.1.3: Identify the IBM WS-based application pattern.

In PWSSec, a relationship is established between every IBM WS-based business pattern and one or more IBM WS-based application patterns [5].

The *E&A Repository – IBM's WS-based Patterns' Catalogue* was again used to identify the most suitable WS-based application pat-

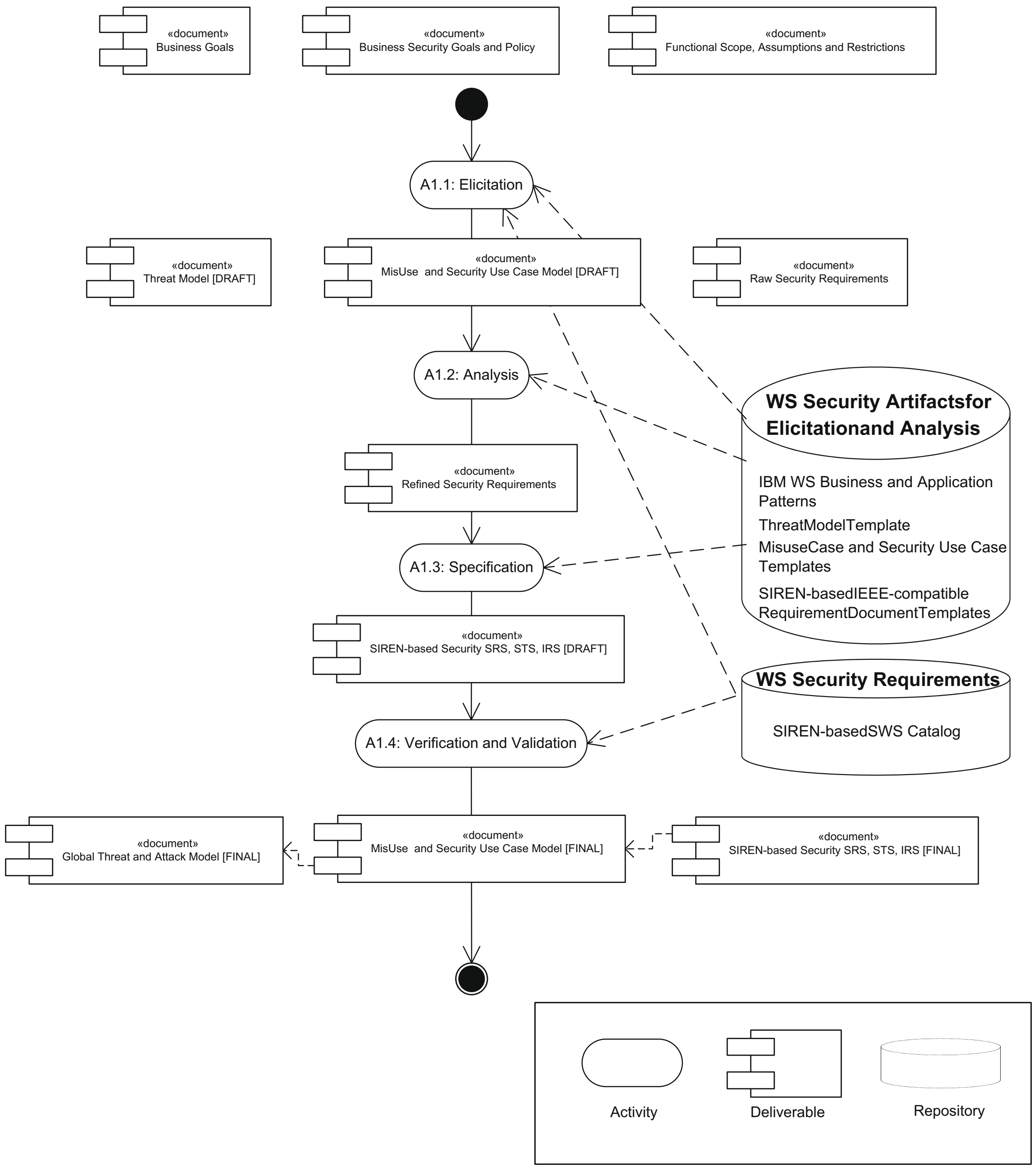

«document» Business Goals
«document» Business Security Goals and Policy
«document» Functional Scope, Assumptions and Restrictions
A1.1: Elicitation
«document» Threat Model [DRAFT]
«document» MisUse and Security Use Case Model [DRAFT]
«document» Raw Security Requirements
A1.2: Analysis
WS Security Artifactsfor Elicitationand Analysis
IBM WS Business and Application Patterns
ThreatModelTemplate
MisuseCase and Security Use Case Templates
SIREN-basedIEEE-compatible RequirementDocumentTemplates
«document» Refined Security Requirements
A1.3: Specification
«document» SIREN-based Security SRS, STS, IRS [DRAFT]
WS Security Requirements
SIREN-basedSWS Catalog
A1.4: Verification and Validation
«document» Global Threat and Attack Model [FINAL]
«document» MisUse and Security Use Case Model [FINAL]
«document» SIREN-based Security SRS, STS, IRS [FINAL]
Activity
Deliverable
Repository

Fig. 5. Main activities and security artifacts of the WSSecReq subprocess.

tern. Once all the potential application patterns within the scope of the Extended Enterprise business pattern had been analyzed, it was clear that the Exposed Direct Connection application pattern was that which best fitted the context of the problem.

The indicated WS-based application pattern was documented as the output of this task, as can be seen in Table 3. This table shows the report generated once the information from this pattern was introduced in step 4 of the UMLPWSSec tool's wizard (see Fig. 8).

We have limited the analysis presented in this paper to the TNT interaction, owing to space limitations.

A1.1. Elicitation – T1.1.4: Identify possible threats at the business-level.

The main objective of this task is to use the business pattern description produced in task T1.1.2 to define the set of business-level threats that apply to the business process supported by the system under development. The resulting output tree is shown in Fig. 9. Showing the business-level risk analysis is not within the scope of this article.

The output of this task was the **Business Threat Model** containing the description of the set of threats identified in the branches of the tree shown in Fig. 9. The graphic representation of the Business Threat Model is based on the OMG Quality-of-Service UML Profile [40].

A1.1. Elicitation – T1.1.5: Identify possible threats at the application-level.

As previously stated in task T1.1.2, we have defined an Application-level Threat Abstract Tree. This tree refines the business threat tree shown in Fig. 9. During task T1.1.5, the UMLPWSSec tool was used to specify the application-level threat tree (see Fig. 10).

Fig. 10 shows the part of the application-level threat tree that extends branch 1.1. The set of threats that must be considered towards the WS agents (WS-BTSConsumer and WS-BTSProvider) involved in the WS-PBT system are arranged beneath this branch.

ARTICLE IN PRESS

PWSSec Process
Sub-process P1 – **WSSecReq**
Activity A1.1: Elicitation
Task T1.1.1: Decide granularity level and identify the fragment of functional software whose security will be analyzed.
Task T1.1.2: Identify the IBM WS-based business pattern.
Task T1.1.3: Identify the IBM WS-based application pattern.
Task T1.1.4: Identify possible threats at the business-level.
Task T1.1.5: Identify possible threats at the application-level.
Task T1.1.6: Relate business and application-level threats.
Task T1.1.7: Identify and evaluate threats.
Task T1.1.8: Identify the type of attackers and their possible types of attack.
Task T1.1.9: Assessment of attack impact.
Task T1.1.10: Estimate and prioritize security risks.
Task T1.1.11: Determine the behaviour the system should have for each attack.
Task T1.1.12: Specify security requirements.
Activity A1.2: Analysis
Task T1.2.1: Identify conflicts due to composition or integration scenarios.
Task T1.2.2: Remove redundant and refine ambiguous requirements.
Task T1.2.3: Classify elicited security requirements.
Task T1.2.4: Identify inclusion/exclusion relationships among the requirements.
Task T1.2.5: Update the E&A Repository.
Activity A1.3: Specification
Task T1.3.1: Instantiate the templates of the security requirement specification documentation.
Task T1.3.2: Arrange set of security requirement specifications adhered to SIREN approach.
Activity A1.4: Verification and Validation
Task T1.4.1: Internal Verification.
Task T1.4.2: External Validation.

Fig. 6. Activities and tasks of WSSecReq subprocess.

Threats to the network are arranged under branch 1.2 according to the network zones previously identified. The analysis of potential threats to the Business Rules (arranged under branch 1.3) was omitted, and was considered to be an open future research line. Fig. 11, shows branch 1.4, under which the set of threats to be studied are deployed.

The output of this task was the **Application-level Threat Model** which accomplished:

- An application-level threat tree instantiated in the WS-PBT system.
- A UML QoS-based threats and assets model. In this case, the sources for identifying the different threats under branch 1.4.x.1 were those defined by the WS-I Consortium [41] and the threat catalogue defined by the Common Criteria compliant security risk analysis and management methodology of the Spanish public administration (MAGERIT2).

Fig. 12 presents the UML QoS [40] representation of the threats to all messages involved in the TNT interaction, and generated by the UMLPWSSec tool from the tree shown in Fig. 11.

In the TNT interaction only the TNT message exists. In this paper, three threats will be analyzed at greater length: TNT Message Modification (branch 1.4.1.1.3), TNT Message Interception (branch 1.4.1.1.2) and TNT Message Counterfeit (branch 1.4.1.1.4).

A1.1. Elicitation – T1.1.6: Relate business and application-level threats.

The aim of this task was to combine the business-level and application-level threat trees in a single threat model. A general vision of the threats to be considered within the scope of the system was thus obtained. The resulting tree is shown in Fig. 13. This task's output was the **Global Threat Model** which included the complete business/application-level threat tree; a description of each threat; and its related UML QoS-based representation.

A1.1. Elicitation – T1.1.7: Identify and evaluate threats.

Tasks T1.1.2 and T1.1.3 allowed us to identify the set of assets of the system and tasks T1.1.4, T1.1.5 and T1.1.6 in order to arrange the set of potential threats to those assets in a tree-like form.

In function of the WS-based systems, specific types of assets were classified as follows:

- Service: WS-PBT.
- WS Agent: WS-BTSConsumer and WS-BTSProvider.
- Messages: TNT, TTR, TTR reply, RNP, RNP reply.
- Underlying Supporting Services (BD, LDAP, other web Services, etc.): in the case study of this system, a database system on the side of the sales organization was identified (BD-BTS).

In the following step, we created the Asset Dependency Matrix. Assets were easily identified by using the Asset Dependency Matrix template stored in *E&A Repository – MAGERIT2 Profile* (see Section 8 for more information about the MAGERIT2 profile). This matrix (see Table 4) and its related graph (see Fig. 14) show the identified security dependencies among assets presented in the application-level threat tree.

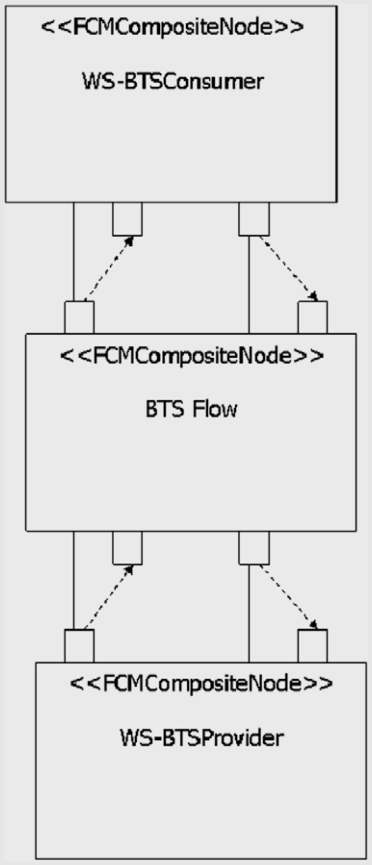
The Asset Dependency Matrix allowed us to define the dependencies among the different assets, thus facilitating the later calculation of the accumulated impact.

The rationale behind the dependencies identified in this Asset Dependency Matrix is the following:

- The security of the WS-BTS systems depends fundamentally on the messages exchanged in their interactions. These messages are sent by applications which, in the context of WS-based systems, are called WS provider/consumer agents.
- Messages depend on the level of security of the underlying network to which they are sent.

Messages, understood as data, depend on the WS-BTSProvider and WS-BTSConsumer agents since these agents generate and consume them, respectively. If the integrity of WS-BTSConsumer or WS-BTSProvider is compromised, the messages they send and receive may also be compromised. In this case, the principle of any security risk and analysis methodology, which states that data (messages) security, must depend on the applications (WS agents) that process them, is applied.

Table 2
Instantiation of the WS-based Extended Enterprise business pattern.

Project: *project code* (*Pddd* fictitious value will be supposed)	
Product: *product code* (*Pppp* fictitious value will be supposed)	
Extended enterprise IBM WS business pattern	
ID	Pppp-EE4WS-1
Business process	Integral process for performing internet bank transfers
Brief description	Business process that accomplishes the performing of online Internet bank transfers, the conciliation and the exchange of information concerning the transfers and payments made
Origin entity	Sales organization
Recipient entity	Banking organization
Network	Internet
Formal description of the business process	 <<FCMCompositeNode>> WS-BTSConsumer <<FCMCompositeNode>> BTS Flow <<FCMCompositeNode>> WS-BTSProvider Description created by using the OMG's UML profile denominated as Flow Control Model. This business pattern defines the following elements: • *Business Entities* that participate in the business case, depicted as elements with stereotype *FCMCompositeNode* • *Network* over which the use case will be performed. This element is not presented in the diagram but it is implicit if the *FCMDataLink* and *FCMControlLink* elements in the diagram are considered to be developed over the Internet • *Business Rules* are presented through the central element with stereotype *FCMCompositeNode* which, in this case, gathers the set of business rules established for the completion of the bank transfer among the systems • *Interactions*, represented through the *FCMControlLink* and *FCMDataLink*
Service quality level clauses	
References to security requirements	
References to reliability requirements	
Abstract threat tree	
Threat tree template ID	A3Bus-EE
Threat tree ID:	Pppp-A3Bus-EE-1
The threat/attack tree can be found in Fig. 9	

The WS-BTSConsumer agent depends on the security of the supporting database system. If a threat to this database system became a real attack, it may cause direct harm to the WS-BTSConsumer agent. As it can be observed in both Table 5 and in the threat tree described in Figs. 10 and 11, we proceeded to identify the most likely threats to the WS-BTS system. This task was trivial since it was simply necessary to complete the threat trees by incorporating two new metrics: threat frequency estimation and asset degradation ratio (see scale in Table 6).

Threat frequency was introduced in task T1.1.8, in which all types of attacks per threat were defined, and the most likely was

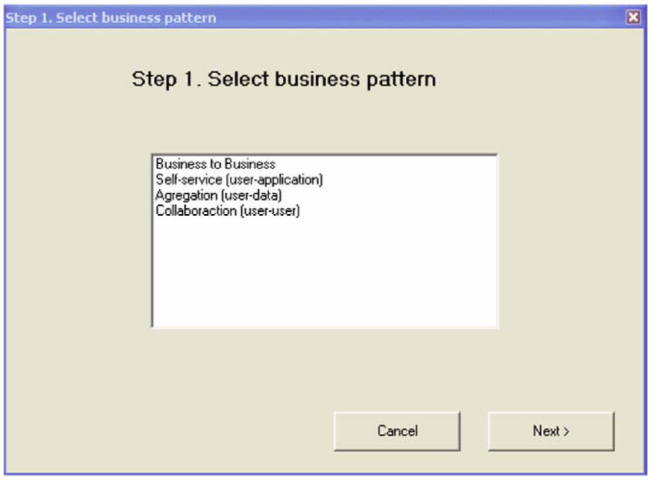

Step 1. Select business pattern
Step 1. Select business pattern
Business to Business
Self-service (user-application)
Agregation (user-data)
Collaboraction (user-user)
Cancel
Next >

Fig. 7a. Screenshots of step 1 of the UMLPWSSec tool.

obtained. The asset degradation ratio was fulfilled in task T1.1.9, in which the impact of threats was calculated. Table 7 shows the Risk Map of the assets identified at the application level. The security dimensions for which a degradation value must be assigned in a further task are marked.

This task's main output was the **Assessed Global Threat Model** which was obtained as a result of adding the analysis performed on the Global Threat Model.

A1.1. Elicitation – T1.1.8: Identify the type of attackers and their possible types of attack.

In the case study at hand, the attack profile known as *Attack Profile of WS-based Application Pattern WS Exposed Direct Connection*, in line with [42], was defined and included in the *Repository E&A – Security Profiles*.

TNT interaction is one-way and does not guarantee message reliability (there is no guarantee that the provider agent will receive the message from the consumer agent) and is performed in a network which is not controlled by any of the involved entities. The variants defined for this profile are detailed below:

- Web Services Provider Entity and Web Services Consumer Entity; in our case, the banking organization and the retail organization selling the product respectively.
- Web Services Provider Agent and Web Services Consumer Agent; in our case, the agents offering the Web services WS-BTS-Provider and WS-BTSConsumer respectively.
- Name of the consumed operation executed. In this case, it is the TNT operation that fulfills the message exchange pattern defined in WSDL (Web Services Description Language), which is called a one-way operation.

The abstract misuse cases defined and included in this profile are the following:

- Attack on the integrity of the SOAP message, which refines the threat represented by threat tree branches 1.4.1.1.1, 1.4.1.1.3. 1.4.1.1.4, 1.4.1.1.5 and 1.4.1.1.6 of the Exposed Direct Connection application pattern shown in Fig. 11. Refining a threat means expressing the set of attacks (represented in PWSSec as misuse cases) from which threats materialize. This attack implies an attack on the message integrity as a result of altering those parts which have a semantic message value.
- Attack on SOAP message authentication, which refines branches 1.4.1.1.4 and 1.4.1.1.5 of the threat tree shown in Fig. 11.
- Attack on SOAP message confidentiality, which refines branches 1.4.1.1.2 and 1.4.1.1.7 of the threat tree shown in Fig. 11.
- Attack on SOAP message replay, which refines branch 1.4.1.1.8 of the threat tree shown in Fig. 11.
- Flood Attack, which refines branch 1.4.1.1.8 of the threat tree shown in Fig. 11.
- Attack on the user identity of the SOAP message, which refines branch 1.4.1.1.1 of the threat tree shown in Fig. 11.

The possible attackers – the primary actors of the proposed misuse cases – are:

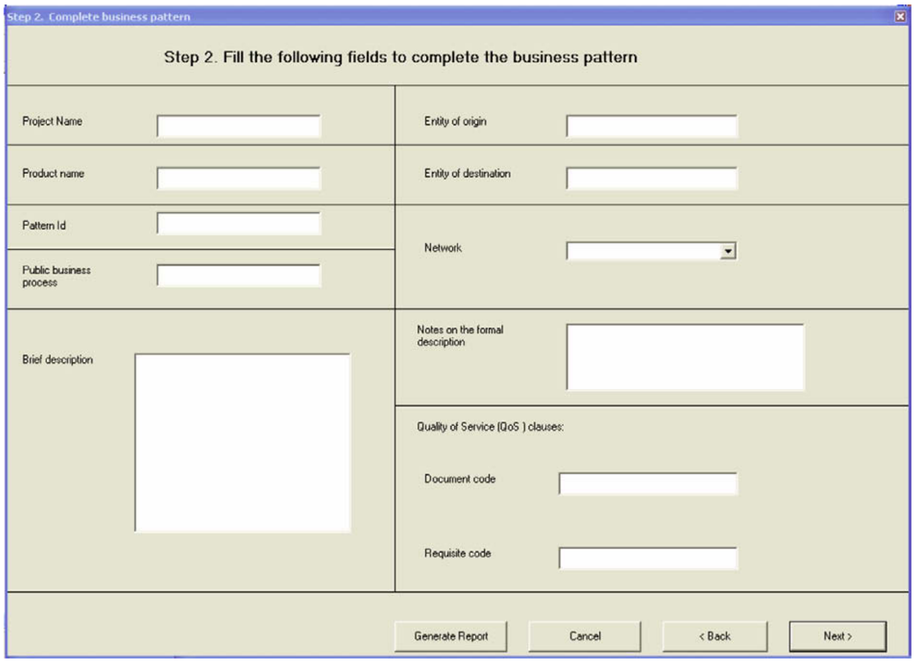


Fig. 7b. Screenshots of step 2 of the UMLPWSSec tool.

Table 3
Instantiation of the Exposed Direct Connection WS-based application pattern,

Project: Pddd				
Product: Pppp				
WS Application Pattern WS Extended Enterprise :: Exposed Direct Connection				
BUSINESS PATTERN CARD ID				Pppp-EE4WS-1
ID				Pppp-EE4WS-1-CED-1
Description				
N.A.				
Functional scope				
'Perform Bank Transfer' use case, which is part of the business process called 'Integral Process for Performing Bank Transfers over the Internet'				
Involved systems				
WS-BTSConsumer system	Type	■ Consumer ☐ Provider ☐ Discovery	Owner entity	Sales Organization
	Objective	To request bank transfers on behalf of potential customers who wish to pay for their products by means of their Internet bank accounts.		
WS-BTSProvider system	Type	☐ Consumer ■ Provider ☐ Discovery	Owner entity	Banking Organization
	Objective	To provide the entry point to the bank transfer core business process of banking organizations		
Connection				
TCP/IP + HTTPS + SOAP				
Connection Rules				
N.A.				
Network Zones				
Name: Public – Internet (ZE)				
Security Level: Low				
Owner: –				
Name: Private – LAN – Banking Organizations (ZS Banking Organization)				
Security Level: High				
Owner: Banking Organization				
Name: Private – LAN – Sales Organization (ZS Sales Organization)				
Security Level: High				
Owner: Sales Organization				
Interactions				
Interaction: TNT	Type of interaction	■ One-way request ☐ Request/response ☐ Notify ☐ Notify/response		
	Pattern variation	Message		
	Origin System	WS-BTSConsumer		
	Recipient System	WS-BTSProvider		
	Information	Header	Message reliability, security	
		Body	Token	
		Attachments	–	
	Connection Rules	–		
	Connection	TCP/IP + HTTPS + SOAP		
	Network Zones	ZS – Sales Organization, ZE – Internet, ZS – Banking Organization		
Representation				
Abstract Threat Tree				
Threat Tree Template ID				A3Ap-CED
Threat Tree ID: A3Bus-EE				A3Ap-CED-1
See Fig. 10				

- **Intermediary WS Agent:** The SOAP architecture upon which WS-based systems are based contains the figure of intermediary SOAP nodes which are able to process the messages during their transit. It is possible and permissible that a sender agent does not know of the existence of this type of intermediaries on the path of the messages it sends.
- **External Attacker:** An attacker located on the Internet who is able to perform some of the aforementioned attacks. There is a high risk that this type of attacker will appear, as the Internet is unpredictable and uncontrollable.

Among these misuse cases, we will describe that known as *Attack to [message name] SOAP message integrity*. This abstract misuse case is shown in Table 8.

The UMLPWSSec was used to instantiate this abstract misuse case, and the concrete misuse case shown in Table 9 was obtained.

Figs. 15 and 16 show the screenshots of the UMLPWSSec tool which allows developers to both manage the defined misuse cases and to instantiate abstract misuse cases (in this case that called *Attack on [message name] SOAP message integrity*).

As in task T1.1.6, we complemented the text description of the misuse cases with the notation provided by the UML QoS profile. The Global Threats Model was thus completed with attack forms and their frequency in order to obtain the **Global Threat and Attack Model**.

A1.1. Elicitation – Task T1.1.9: Assessment of attack impact. In this task, we performed the assessment of the attack impact with the aim of obtaining the complete Risk Map presented in Table 10 as a tangible result. The final version of the Risk Map establishes the asset degradation value caused by the fulfillment of threats in each of the dimensions. This Risk Map was, moreover, completed with an additional column indicating the accumulated impact for lower "assets" and the propagated impact for upper "assets".

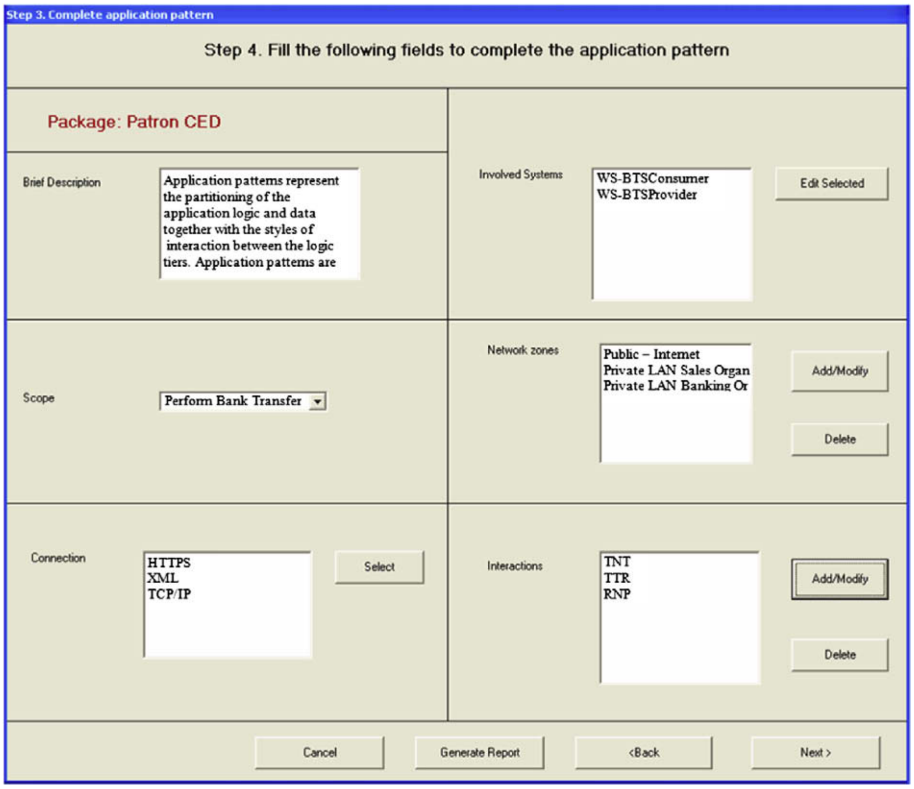
When an asset is the victim of a threat, part of its value is degraded. We can intuitively refer to an "asset degradation percentage" which may cause between 0% and 100% of the value to be lost. "*d*" stands for a real value of between 0.0 (0% degradation) and 1.0 (100% degradation) [24]. Two types of impact can be defined: accumulated impact and propagated impact.

The accumulated impact of a threat to a "lower" asset is the measurement of what that threat implies to that asset, in other words, the loss of its accumulated value. The accumulated value is calculated on the "lower" assets which have no value in themselves but adopt their value from the "upper" assets which have their own values. If an asset has an accumulated value in a certain dimension "*vd*" and is degraded by a "*d*" percentage, the value of the impact will be:

$$I = round(vd * d).$$

When an "upper" asset A depends on a "lower" asset B, threats to B have an effect on A. If asset B has suffered a *d* degradation level, the same will happen to asset A, in other words, its own value will be degraded in *d*.

As a result of this task, we will obtain the Valued Threat and Attack Model in which the Risk Map of the system will have been partially completed. In Table 10, we show the Risk map for WS-BTSConsumer and TNT message assets. In each cell of the WS-BTSConsumer asset, three values are presented: the asset's accumulated value, the degradation percentage and the value of the accumulated impact. For the TNT message asset, we show the asset's own value and the propagated impact. The WS-BTS-Consumer asset is a "lower" asset, so its value and impact are therefore accumulated. On the other hand, the TNT message asset is an "upper" asset and therefore has its own value and propagated impact. The propagated impact is materialized so that each dimension inherits the greater degradation degree from the assets on which they depend. Table 10 therefore shows that for all threats, and within the Integrity dimension, the impact inherits a 100% degradation percentage since threat 1.1.1.1.1.8 to the WS-BTSConsumer asset has a greater degree of degradation in the Integrity security dimension. This table also includes the frequency of each threat according to the Misuse Cases obtained from each of them in Task 1.1.9. We have used the frequency scale shown in Table 11.


Step 3. Complete application pattern

Step 4. Fill the following fields to complete the application pattern

Package: Patron CED

Brief Description: Application patterns represent the partitioning of the application logic and data together with the styles of interaction between the logic tiers. Application patterns are

Involved Systems: WS-BTSConsumer, WS-BTSProvider

Scope: Perform Bank Transfer

Network zones: Public – Internet, Private LAN Sales Organ, Private LAN Banking Or

Connection: HTTPS, XML, TCP/IP

Interactions: TNT, TTR, RNP

Fig. 8. Screenshot of step 4 of the UMLPWSSec tool.

Threat/Attack Tree related to the WS-based business pattern Extended Enterprise
ID: A3Bus-EE-1
Goal: 1. Cause harm in the execution of the business process ‘Integral Process for Performing Internet Third-party Bank Tranfers’, Business Use Case ‘Perform Bank Transfer’

- 1.1 Attack Business Entities
  - 1.1.1 Attack Business Entity ‘Sales Organization’
  - 1.1.2 Attack Business Entity ‘Banking Organization’
- 1.2 Attack Network Infrastructure
- 1.3 Attack Business Rules
- 1.4 Attack Business Interactions
  - 1.4.1. Attack Service PBT (Perform Bank Transfer)

Fig. 9. Threat/attack tree associated with the extended enterprise WS-based business pattern.

A1.1. Elicitation – Task T1.1.10: Estimate and prioritize the security risks.

Finally, the risk was estimated and prioritized by completing the Global Threat and Attack Model through the assignment of the risk value. The risk value is a function of impact and frequency.

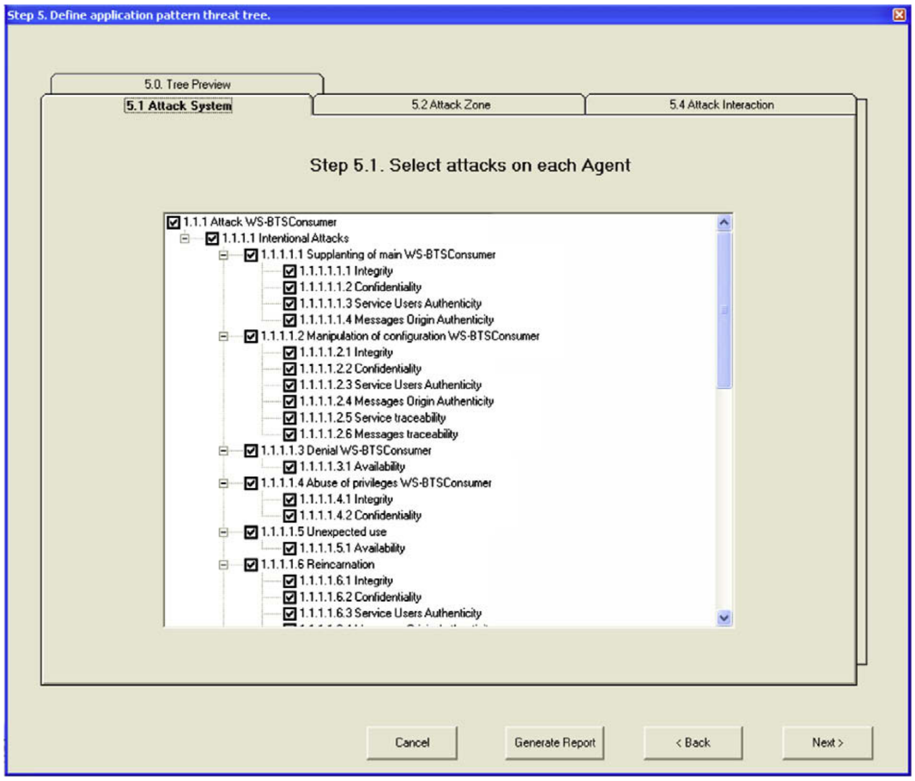
At first, the frequency was assigned in such a way that the risk for each asset and threat was the corresponding value assigned to the impact. The value of the frequency was modified according to the state of the real production system. Thus, it was possible to assign realistic historical values to the frequencies of each of the threats. Table 12 presents a partial view of the Risk Map showing risks calculated for each security dimension of the WS-BTSConsumer and TNT message assets.

We initially considered a medium frequency for all threats, and the risk value computed was therefore the same as the impact value. Owing to the nature of the security events in the system, the frequency of threats was modified in order to additionally modify the risk values. For example, if threat 1.4.1.1.1.1 takes place monthly, its frequency will change to Monthly Frequency (MF) and consequently, the risk should increase to High (H).

A1.1. Elicitation – Task T1.1.11: Determine the behavior the system should have for each attack.

The abstract misuse case *Attack on [message name] SOAP message integrity*, shown in Table 8, holds an association with the abstract security use case which defines the sequence of steps that the system should perform to prevent, detect or react to the misuse case. This abstract security use case, called *Protect [message name] SOAP Message Integrity*, included within the *Attack Profile of WS-based Application Pattern WS Exposed Direct Connection* and stored in the *Repository E&A – Security Profiles*, was instantiated through the UMLPWSSec tool. As Fig. 17 shows, the UMLPWSSec tool also supports the management of security use cases. This figure shows a screenshot of the UMLPWSSec tool, which allows the instantiation of abstract security use cases and traces their relationship with their corresponding misuse cases. In step 8, the tool allows the developer to instantiate one or more abstract security use cases whose type depends on the type of security subfactor and its related misuse cases. Traceability can thus be continued from misuse cases to security use cases and, as we shall now see, up to security requirements.

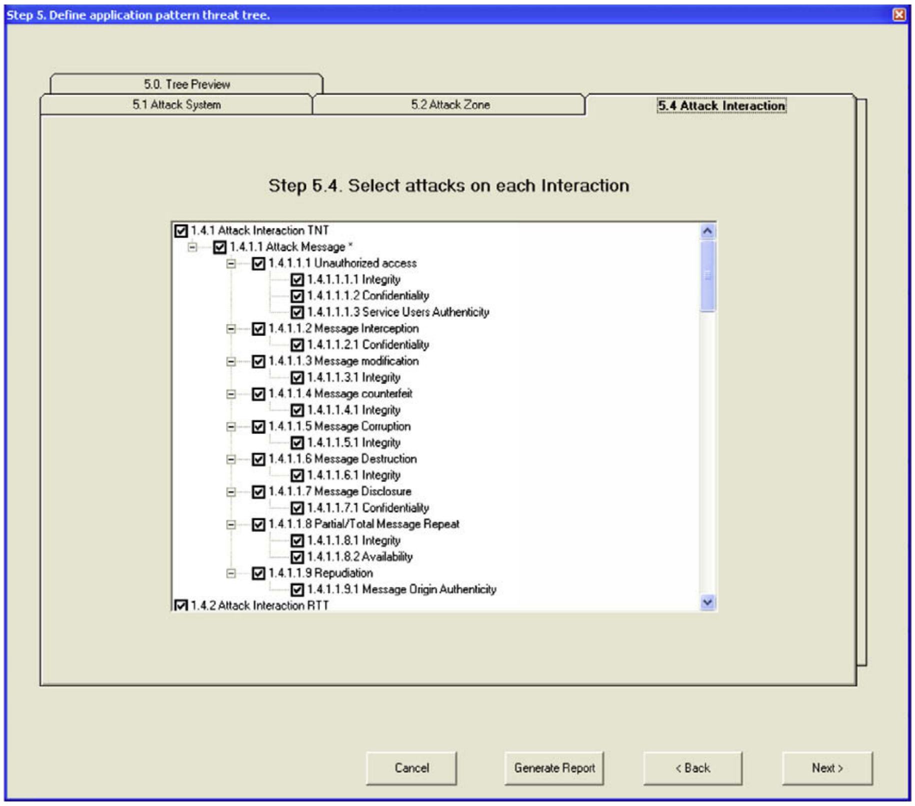
This abstract security use case demonstrates how the WS agent recipient, upon recognizing the message, throws it out and exe-



Step 5.1. Select attacks on each Agent

- ☑ 1.1.1 Attack WS-BTSConsumer
  - ☑ 1.1.1.1 Intentional Attacks
    - ☑ 1.1.1.1.1 Supplanting of main WS-BTSConsumer
      - ☑ 1.1.1.1.1.1 Integrity
      - ☑ 1.1.1.1.1.2 Confidentiality
      - ☑ 1.1.1.1.1.3 Service Users Authenticity
      - ☑ 1.1.1.1.1.4 Messages Origin Authenticity
    - ☑ 1.1.1.1.2 Manipulation of configuration WS-BTSConsumer
      - ☑ 1.1.1.1.2.1 Integrity
      - ☑ 1.1.1.1.2.2 Confidentiality
      - ☑ 1.1.1.1.2.3 Service Users Authenticity
      - ☑ 1.1.1.1.2.4 Messages Origin Authenticity
      - ☑ 1.1.1.1.2.5 Service traceability
      - ☑ 1.1.1.1.2.6 Messages traceability
    - ☑ 1.1.1.1.3 Denial WS-BTSConsumer
      - ☑ 1.1.1.1.3.1 Availability
    - ☑ 1.1.1.1.4 Abuse of privileges WS-BTSConsumer
      - ☑ 1.1.1.1.4.1 Integrity
      - ☑ 1.1.1.1.4.2 Confidentiality
    - ☑ 1.1.1.1.5 Unexpected use
      - ☑ 1.1.1.1.5.1 Availability
    - ☑ 1.1.1.1.6 Reincarnation
      - ☑ 1.1.1.1.6.1 Integrity
      - ☑ 1.1.1.1.6.2 Confidentiality
      - ☑ 1.1.1.1.6.3 Service Users Authenticity

Fig. 10. WS-based application-level threat/attack tree – view of threats to the WS agents.



Step 5.4. Select attacks on each Interaction

- ☑ 1.4.1 Attack Interaction TNT
  - ☑ 1.4.1.1 Attack Message *
    - ☑ 1.4.1.1.1 Unauthorized access
      - ☑ 1.4.1.1.1.1 Integrity
      - ☑ 1.4.1.1.1.2 Confidentiality
      - ☑ 1.4.1.1.1.3 Service Users Authenticity
    - ☑ 1.4.1.1.2 Message Interception
      - ☑ 1.4.1.1.2.1 Confidentiality
    - ☑ 1.4.1.1.3 Message modification
      - ☑ 1.4.1.1.3.1 Integrity
    - ☑ 1.4.1.1.4 Message counterfeit
      - ☑ 1.4.1.1.4.1 Integrity
    - ☑ 1.4.1.1.5 Message Corruption
      - ☑ 1.4.1.1.5.1 Integrity
    - ☑ 1.4.1.1.6 Message Destruction
      - ☑ 1.4.1.1.6.1 Integrity
    - ☑ 1.4.1.1.7 Message Disclosure
      - ☑ 1.4.1.1.7.1 Confidentiality
    - ☑ 1.4.1.1.8 Partial/Total Message Repeat
      - ☑ 1.4.1.1.8.1 Integrity
      - ☑ 1.4.1.1.8.2 Availability
    - ☑ 1.4.1.1.9 Repudiation
      - ☑ 1.4.1.1.9.1 Message Origin Authenticity
- ☑ 1.4.2 Attack Interaction RTT

Fig. 11. WS-based application-level threat/attack tree – view of threats to the WS interactions.

cutes a series of operations (logging, alert emission, etc.). The recipient WS agent throws the message out without notifying the sender WS agent because this abstract use case is associated with the application pattern for WS identified in task T1.1.3. An example of

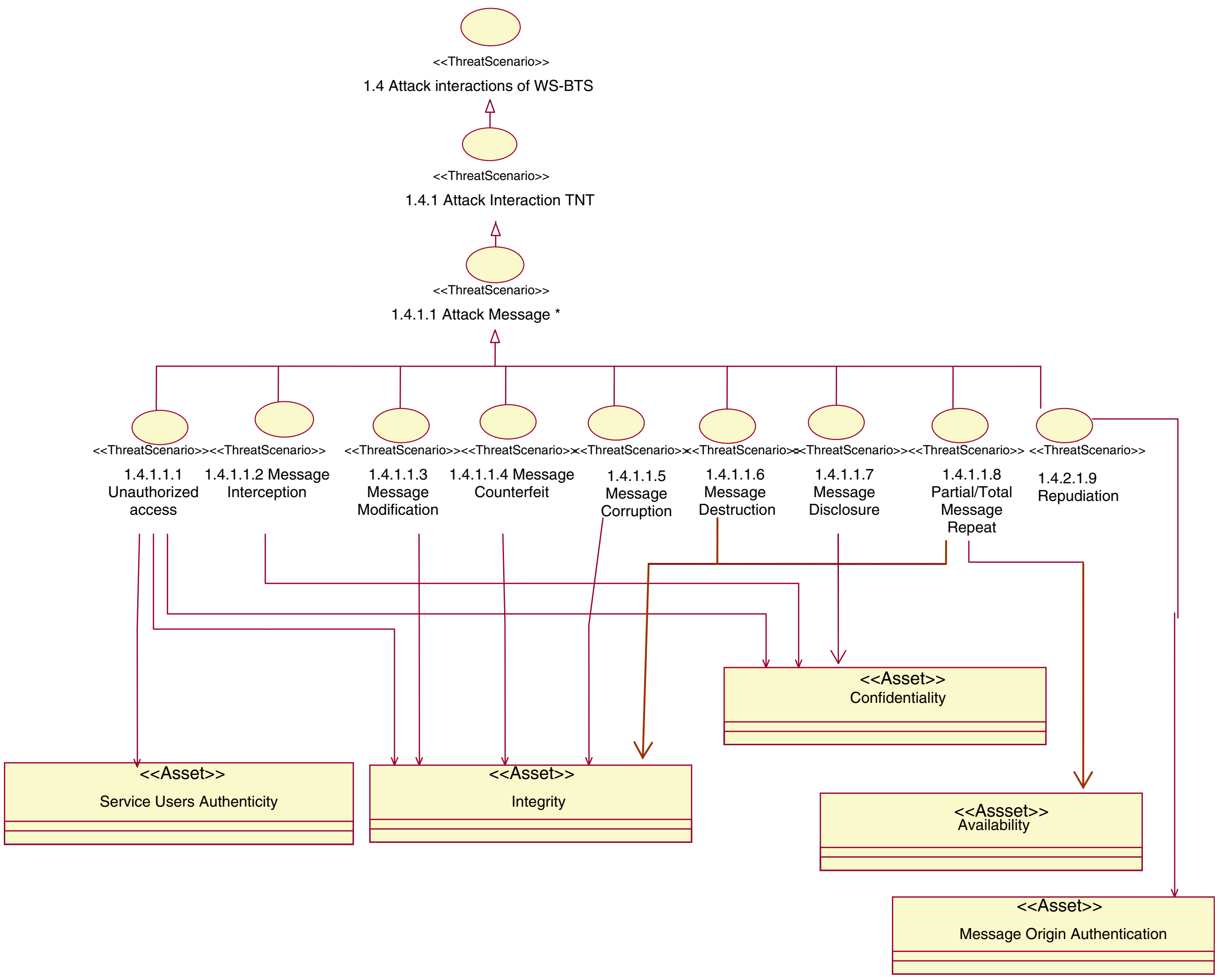

<<ThreatScenario>>
1.4 Attack interactions of WS-BTS
<<ThreatScenario>>
1.4.1 Attack Interaction TNT
<<ThreatScenario>>
1.4.1.1 Attack Message *
<<ThreatScenario>>
1.4.1.1.1 Unauthorized access
<<ThreatScenario>>
1.4.1.1.2 Message Interception
<<ThreatScenario>>
1.4.1.1.3 Message Modification
<<ThreatScenario>>
1.4.1.1.4 Message Counterfeit
<<ThreatScenario>>
1.4.1.1.5 Message Corruption
<<ThreatScenario>>
1.4.1.1.6 Message Destruction
<<ThreatScenario>>
1.4.1.1.7 Message Disclosure
<<ThreatScenario>>
1.4.1.1.8 Partial/Total Message Repeat
<<ThreatScenario>>
1.4.2.1.9 Repudiation
<<Asset>> Confidentiality
<<Asset>> Service Users Authenticity
<<Asset>> Integrity
<<Assset>> Availability
<<Asset>> Message Origin Authentication

Fig. 12. UML QoS-based threat/attack tree.

<u>Abstract TA Tree related to the WS-based business/appli cation pattern Extended Enterprise/Exposed Direct Connection</u>
ID: A3-EE-CED-1
Goal: 1. Cause harm in the execution of the business process ‘Integral Process for Performing Internet Third-party Bank Tranfers’, Business Use Case ‘Perform Bank Transfer’
1.1 Attack Business Entities
1.1.1 Attack Business Entity ‘Sales Organization'
1.1.1.1 Attack WS-BTSConsumer
1.1.1.1.1 Suplanting of Principal WS-BTSConsumer
1.1.1.1.2 Manipulation of configuration WS-BTSConsumer
1.1.1.1.3 Denial of Service WS-BTSConsumer
1.1.1.1.4 Abuse of privileges WS-BTSConsumer
1.1.1.1.5 Unexpected use
1.1.1.1.6 Messaging re-routing
1.1.1.1.7 Sequence alteration
1.2 Attack network
1.3 Attack Business Rules
1.4 Attack Interactions
1.4.1. *Attack Service PBT (Perform Bank Transfer)*
1.4.1.1 Attack Interaction TNT
…

Fig. 13. Business/application threat/attack tree of the use case ‘Perform Bank Transfer’.

instances performed on this abstract security use case is shown in Table 13.

A1.1. Elicitation - Task T1.1.12: Specify security requirements.

Subtask T1.1.12.1: Select the security requirement templates from the repository.

In this subtask, it was necessary to retrieve the security requirement templates associated with each abstract security case previously instantiated from the *Repository E&A – Security for Web Services Catalogue* (presented in [43]).

With regard to the abstract security case *Protect [message] Message Integrity*, two security requirement templates were defined according to the side of the system (consumer or provider) from which the security requirement was written. Both security requirement templates are presented in Fig. 18.

Table 4
Asset dependency table.

	WS-PBT	TNT message	TTR message	TTR response message	RNP message	RNP response message	WS-BTSConsumer	WS-BTSProvider	BD BTS
WS-PBT		*	*	*	*	*	*	*	*
TNT message							*		*
TTR message								*	
TTR Response message							*		*
RNP message								*	
RNP response message							*		*
WS-BTSConsumer									*
WS-BTSProvider									
BD BTS									

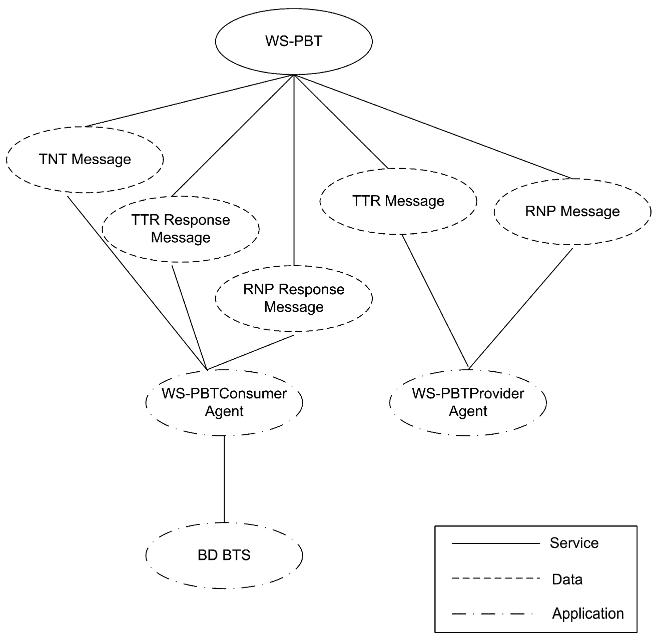


Fig. 14. Asset dependency graph of the WS-BTS system.

Fig. 19 shows how the instantiation of these templates through the UMLPWSSec tool allows traceability to be graphically reflected, from the WS-based application pattern and its logical components, passing through the potential attacks on them (misuse cases) and the types of countermeasures that the system must provide (security use cases) to obtain the security requirements that must be addressed.

Subtask T1.1.12.2: Select security criteria.

The security criterion established for writing the security requirements was determined in terms of the minimum number of transmissions to be performed. The security subfactor associated with each requirement will be guaranteed in these transmissions.

Subtask T1.1.12.3: Select security metrics.

The security metrics defined to write the security requirements were determined for measurement in percentage units and to reflect the degree of presence of the security criterion selected in the previous step.

Subtask T1.1.12.4: Establish the minimum accepted level for each security metric.

The minimum accepted level established was 99.99%.

Subtask T1.1.12.5: Security requirement specification.

This task consisted of instantiating each of the security requirements templates from all of the variables fixed in the previous subtasks (see Fig. 20).

7.1.2. WSSecReq – analysis activity

A1.2. Analysis – Task T1.2.1: Identify composition or integration conflicts.

The execution of this task was performed through peer reviews with the stakeholders. Conflict was found in neither composition nor integration since the services defined within the scope of the system (WS-BTSProvider and WS-BTSConsumer) had the sole purpose of taking part in the BTS business process and in no other.

A1.2. Analysis – Task T1.2.2: Elimination of redundancies and ambiguities.

During the review meeting held to review the resulting security requirements, we found certain imprecision with regard to the way in which the sales organization used banking business domain terms. Furthermore, we reviewed the fact that each security

Table 5
Asset Table Assessment.

Asset	Security dimensions						
	Integrity (I)	Confidentiality (C)	Availability (A)	Service user authentication (SUA)	Message source authentication (MSA)	Service traceability (ST)	Data/message traceability (DMT)
WS-PBT			M	H		M	
TNT message	M	L	M	H	M	M	L
TTR message	L	L	M	H	M	M	M
TTR message reply	VL	L	M	H	L	M	L
RNP message	H	M	M	H	M	M	M
RNP message reply	M	M	M	H	M	M	M
WS-BTSConsumer	M	M	M	H	M	M	M
WS-BTSProvider	H	M	M	H	M	M	M
BD BTS			M	H		M	

ARTICLE IN PRESS

Table 6
Scale of values used to assess the assets.

Value	Description
VL	Very low
L	Low
M	Medium
H	High
VH	Very high

requirement was written from the point of view of the two participant agents to allow them to be independently verified.

A1.2. Analysis – Task T1.2.3: Classification of the security requirements according to security topic.

The classification of the set of elicited security requirements allowed us to decide whether the requirement was application-level or business-level. These applications were, moreover, classified into system, software, interface, priority, criticality and viability requirements.

A1.2. Analysis – Task T1.2.4: Identification of inclusion/exclusion traceability relationships.

We identified inclusion relationships between the security requirements which referred to the same element to be protected, associated with the integrity security requirement type, and those related to the authenticity of the data origin since both quality subfactors are dependent on one another.

A1.2. Analysis – Task T1.2.5: Update the SWS Catalogue.

As this was the first time that this process had been applied, we incorporated the set of new security requirements templates (defined during the application of the PWSSec process to this case study) into the SWS Repository (see Fig. 5).

A1.2. Analysis – Task T1.2.6: Prioritize security requirements.

Requirements prioritization was performed according to the Risk Map produced during the elicitation activity. Thanks to the fact that each security requirement is defined in relation to one single element (e.g. WS agent or a determined SOAP message), it was easy to connect each security requirement, through its referred element, to its associated risks. The output of this task allowed us to obtain an assessment of the risk associated with each security requirement.

7.1.3. *WSSecReq – specification activity*

A1.3. Specification – Task T1.3.1: Instantiate the templates of the security requirement specification documentation.

During the first iteration, we used the *Repository E&A –Software Requirement Specification Template* security artifact to obtain the documentation conforming to the IEEE 1998-830 standard of software requirements specification [26]. Due to the relatively small size of the system, it was not necessary to instantiate the *Repository E&A – Interface Specification Template*.

The *Repository E&A-Software Testing Specification Template* security artifact was additionally used to draw up the security validation case test document for the requirements specified from the outset. The security requirements were individually instantiated according to the security pattern proposed in task T1.3.1 and the result of the integrity and confidentiality security subfactor for TNT SOAP message is shown in Tables 14 and 15.

A1.3. Specification – Task T1.3.2: Arrange set of security requirement specifications adhering to SIREN approach.

During the application of this task, the requirements documentation was structured into the hierarchy established by the SIREN requirements reuse process [44].

Table 7
WS-BTS system's Risk Map.

Asset	Threat	F	Security dimensions						
			I	C	A	SUA	MSA	ST	DMT
WS-BTSConsumer	1.1.1.1.1.1	0	X	X		X	X		
	1.1.1.1.1.2	0	X	X		X	X	X	X
	1.1.1.1.1.3	0			X				
	1.1.1.1.1.4	0	X	X					
	1.1.1.1.1.5	0			X				
	1.1.1.1.1.6	0	X	X		X	X		
	1.1.1.1.1.7	0	X						
	1.1.1.1.1.8	0	X	X		X			
WS-BTSProvider	1.1.1.1.2.1	0	X	X		X	X		
	1.1.1.1.2.2	0	X	X		X	X	X	X
	1.1.1.1.2.3	0			X				
	1.1.1.1.2.4	0	X	X					
	1.1.1.1.2.5	0			X				
	1.1.1.1.2.6	0	X	X		X			
	1.1.1.1.2.7	0	X						
	1.1.1.1.2.8	0	X	X		X			
TNT message	1.4.1.1.1	0	X	X		X			
	1.4.1.1.2	0							
	1.4.1.1.3	0	X						
	1.4.1.1.4	0	X						
	1.4.1.1.5	0	X						
	1.4.1.1.6	0	X						
	1.4.1.1.7	0		X					
	1.4.1.1.8	0	X		X				
	1.4.1.1.9	0					X		
BD BTS	1.4.1.2.1	0	X	X		X			
	1.4.1.2.2	0		X					
	1.4.1.2.3	0	X						
	1.4.1.2.4	0	X						
	1.4.1.2.5	0	X						
	1.4.1.2.6	0	X						
	1.4.1.2.7	0		X					

ARTICLE IN PRESS

Table 8
Template of misuse case 'Attack to SOAP message integrity'.

Misuse Case Name: Attack to [message name] SOAP message integrity		
ID	*ID*	
Abstract Misuse Case ID	*CMUA-1-1-1*	
Related Use Cases	[Identifiers]	
Related threats	A3Ap-CED – 1.4.1.1.1, 1.4.1.1.3. 1.4.1.1.5 y 1.4.1.1.6	
Summary	The attacker type accesses to the message (message name) exchanged between the [consumer/provider/discovery] agent [agent name] and the [consumer/provider/discovery] agent [agent name] and [modifies/eliminates/inserts [part]] the message at the [transport/SOAP] level located at the [heading/body/annex] with the purpose of [objective]	
Security dimensions	Integrity Message origin authentication	
Frequency	[VERY HIGH FREQUENCY | HIGH FREQUENCY | MEDIUM FREQUENCY | LOW FREQUENCY | VERY LOW FREQUENCY] [INCIDENTS PER YEAR]	
Sophistication	[VERY HIGH | HIGH | MEDIUM | LOW | VERY LOW]	
Priority	[VERY HIGH | HIGH | MEDIUM | LOW | VERY LOW]	
Deployment environment	[Select security zones identified in the application pattern]	
Attacker	[Attacker type]	
Entry point	[Network| Host | Application]	
Preconditions	(1) The attacker has physical access to the message (2) The attacker has a clear knowledge of the structure and meaning of the message	
Assumptions		
Interactions of Consumer Agent	Interactions of Attacker	Interactions of WS Agent Provider
The consumer agent sends the message [message name]		
	The attacker [attacker type] intercepts it	
	The attacker [attacker type] identifies the part to be modified and [eliminates, replaces or aggregates] information	
	The attacker forwards the message to the WS Agent Provider	
		The provider agent receives the message and processes it erroneously in accordance with the altered semantics.
Postconditions	Threat in worst case	(1) The system will be in an erroneous state with regard to the original intentions of the consumer agent [consumer agent name] (2) The system's register over which the provider agent [provider agent name] is executed will show that the received petition was that with the altered semantics
	Desired prevention guarantee	
	Desired detection guarantee	
	Desired recovery guarantee	

SIREN is a method for requirements engineering based on requirements reuse which includes a general purpose process model that can be applied to a wide variety of domains and profiles, such as, security, personal data protection law and e-commerce. The reusable requirements are organized in catalogues according to the domain or profile to which they belong. The structure of each catalogue is a hierarchy of documents templates conforming to the IEEE requirements specification standards. In [43] we presented the SWS (Security for Web Services) catalogue which contains requirements templates that can be reused between projects.

In this case, we completed the WS-BTS subsystem software requirements specification document and its corresponding Software Testing Specification document with the defined security requirements. As the WS-BTS workflow is, from the viewpoint of the Organization selling the products (IPSS – Internet Product Selling System), part of a larger system in charge of selling the products, other requirements specification documents for other subsystems already existed. We therefore defined a new set of particular specifications for the system. Fig. 21 depicts the location of software requirements specifications (and their corresponding testing specification) of the WS-BTS subsystem in the context of IPSS (those surrounded by the red square). We also show the specifications defined for the VxI Web application which allows purchasers to select the products.

7.1.4. WSSecReq – verification and validation activity

A1.4: Verification and validation – Task T1.4.1: Internal verification.

The RE Group carried out an inspection of the specified security requirements to check that they were not ambiguous, that no conflict existed, that the inclusion/exclusion traceability relationships were correct and so on.

A1.4: Verification and validation - Task T1.4.2: External validation.

This task basically consisted of holding review meetings with the actors involved in this subprocess in order to verify that the elicited security requirements fulfilled their interests.

The main output generated as a result of the application of the WSSecReq subprocess was the Security Requirement Specification Documents Bundle.

7.2. Subprocess P2 – WSSECARCH

This section summarizes how each of the WSSecArch subprocess activities was developed without giving details of the application of each of their tasks. For each activity, we will indicate the actors that took part, what their purpose was, what input and output products were used and obtained respectively, and how the activity was performed.

The input to this subprocess is accomplished by using the output of the WSSecReq subprocess: the Global Threat and Attack Model, the Misuse and Security Use Case Model, and the SIREN-based Security Set of Requirement Specifications.

7.2.1. Identification of security patterns

The set of security requirements established as a basis through which to create the initial architecture of the system was com-

Table 9
Attack on TNT SOAP Message Integrity misuse case.

Misuse Case Name: Attack to TNT SOAP Message Integrity			
ID	*CMU-1-1-1-1*		
Abstract Misuse Case Source ID	*CMUA-1-1-1*		
Related Use Cases	UC-1		
Related threats	A3Ap-CED – 1.4.1.1.1, 1.4.1.1.3. 1.4.1.1.5 y 1.4.1.1.6		
Summary	The external attacker type accesses the TNT message exchanged between the WS-BTSConsumer consumer agent and the WS-BTSProvider provider agent and modifies the part of the SOAP message containing the bank account number of the beneficiary of the transfer located in the SOAP body with the purpose of altering its meaning by changing the account number into that belonging to the attacker.		
Security dimensions	Integrity Message origin authentication		
Frequency	Medium frequency		
Sophistication	High		
Priority	High		
Deployment environment	ZE – Internet		
Attacker	External attacker		
Entry point	Network		
Preconditions	(1) The attacker has physical access to the message (2) The attacker has a clear knowledge of the structure and semantic of the message		
Assumptions			
Interactions of consumer agent	Interactions of attacker	Interactions of provider agent	
The WS-BTSConsumer consumer agent sends the TNT message			
	The external attacker intercepts it The external attacker modifies the bank account number The external attacker resends the TNT message to the WS-BTSProvider provider agent		
		The WS-BTSProvider provider agent receives the message and processes it erroneously in accordance with the altered semantics, in other words, it establishes that the consumer agent wishes to perform a transfer from the customer account to the account modified by the external attacker	
Postconditions	Threat in worst case	(1) The system will be in an erroneous state with regard to the original intentions of the WS-BTSConsumer consumer agent (2) The system's register over which the provider agent WS-BTSProvider is executed will show that the message requested which was received was that with the altered semantics	
	Desired prevention guarantee	(1) Robust material/ cryptographic algorithms (2) Application of reduced time windows that decrease to a minimum the potential validity period of the attack	
	Desired detection guarantee	(1) Precise knowledge of altered information	
	Desired recovery guarantee	–	

posed of those related to integrity, message origin authentication and confidentiality.

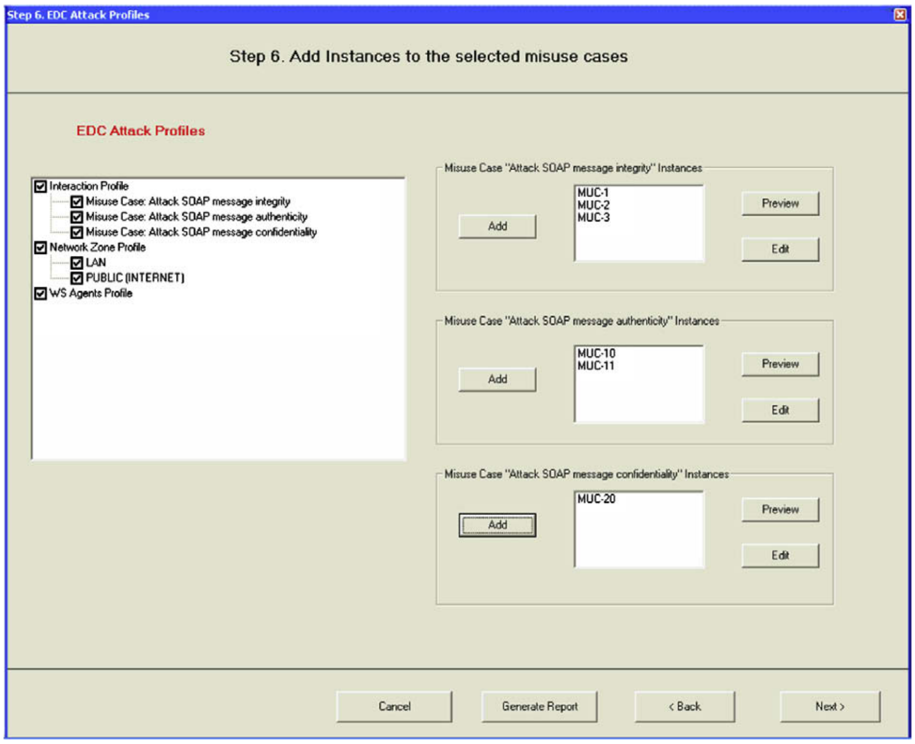
The *WSSecArch – repository of architecture patterns/security design* for WS was used to identify the security pattern. In this repository, various architecture security patterns are organized according to the type of security requirements they address. The security requirements considered in the current iteration can be solved by applying the Quality of Protection (QoP) pattern which defines a generic WS-based security service guaranteeing message confidentiality, authentication and integrity [45]. This pattern defines a basic service which applies and verifies the security mechanisms necessary to guarantee the integrity, authenticity of origin and confidentiality of messages. The security pattern selection process is security requirement-oriented. Within the aforementioned repository each architectural security pattern is related to one or more security requirements and, in turn, every security architectural pattern (e.g.: QoP pattern) has one or more associated security design patterns (e.g.: Single Sign-on delegation, WS security message inspector) which can implemented through one or more WS-based security standards (e.g.: WS-Security + XML Encryption + XML Digital Signature) [45].

Therefore, as a result of applying this activity, the architect produced a description conforming to the IEEE 1471-2000 [16] of the QoP security architecture pattern which would be included in the initial specification of the security architecture (SASSec – Architecture Security Patterns section). This description, which is particularized to the context of the system, describes the QoP application pattern from different points of view. The complete description is composed of a set of views including:

- View of misuse cases and security use cases solved by the QoP architecture pattern. This view also shows which security requirements are being solved. As part of this view, security requirements are documented in XML document form and misuse and security use cases are mentioned and considered.
- View of logical security, which shows how a potential customer can use the security service offered by the QoP pattern. This logical view distributes the logical security components throughout different classes of security services. In the case of the QoP pattern, this view is formed of only one security service class in the manner of a security interface.
- View of security components, which describes the existing runtime entities that take part in the achievement of the different interactions (TNT, TTR and RNP).

7.2.2. Security architecture design

The security architecture design consequently gave place to a draft of the security architecture that fulfilled the security requirements selected in the previous section. To sum up, we instantiated the most relevant logic components of the reference architecture for WS described in [6].


Step 6. EDC Attack Profiles

Step 6. Add Instances to the selected misuse cases

EDC Attack Profiles

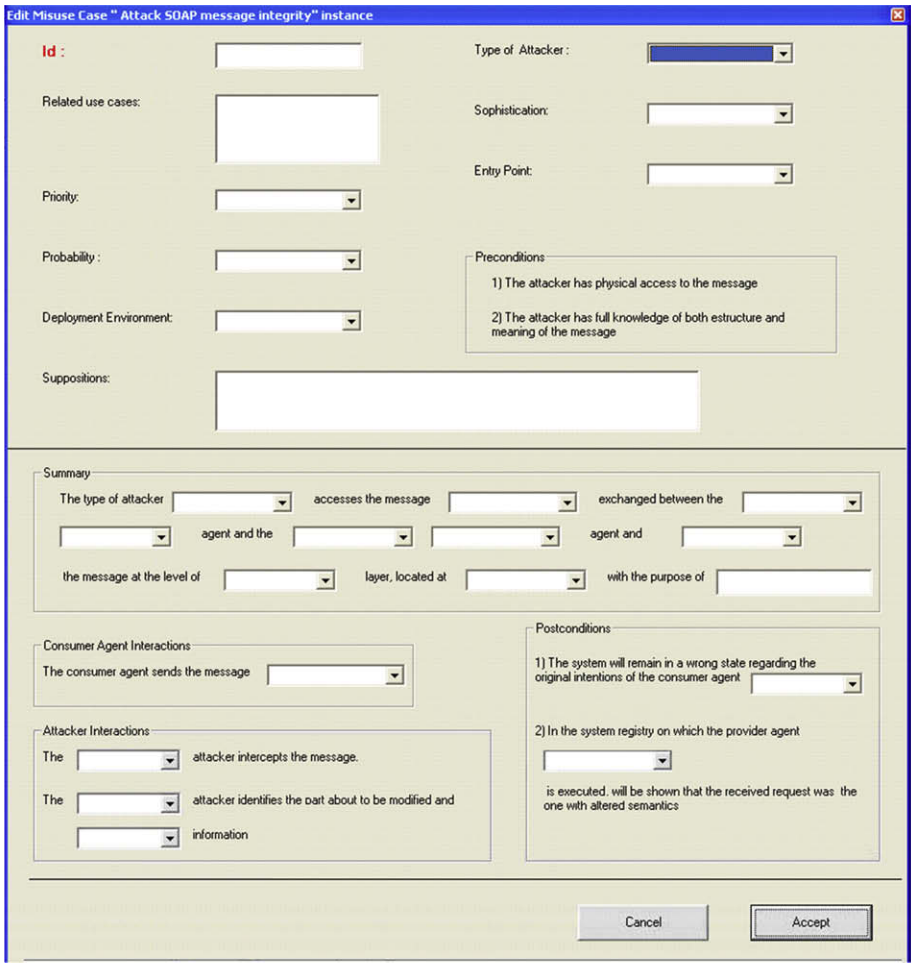
- [✓] Interaction Profile
  - [✓] Misuse Case: Attack SOAP message integrity
  - [✓] Misuse Case: Attack SOAP message authenticity
  - [✓] Misuse Case: Attack SOAP message confidentiality
- [✓] Network Zone Profile
  - [✓] LAN
  - [✓] PUBLIC (INTERNET)
- [✓] WS Agents Profile

Misuse Case "Attack SOAP message integrity" Instances: MUC-1, MUC-2, MUC-3

Misuse Case "Attack SOAP message authenticity" Instances: MUC-10, MUC-11

Misuse Case "Attack SOAP message confidentiality" Instances: MUC-20

Fig. 15. Misuse cases instantiation with UMLPWSSec.


Edit Misuse Case " Attack SOAP message integrity" instance

Id :

Related use cases:

Priority:

Probability :

Deployment Environment:

Suppositions:

Type of Attacker :

Sophistication:

Entry Point:

Preconditions

1) The attacker has physical access to the message

2) The attacker has full knowledge of both estructure and meaning of the message

Summary

The type of attacker [ ] accesses the message [ ] exchanged between the [ ] [ ] agent and the [ ] [ ] agent and [ ] the message at the level of [ ] layer, located at [ ] with the purpose of [ ]

Consumer Agent Interactions

The consumer agent sends the message [ ]

Attacker Interactions

The [ ] attacker intercepts the message.

The [ ] attacker identifies the part about to be modified and [ ] information

Postconditions

1) The system will remain in a wrong state regarding the original intentions of the consumer agent [ ]

2) In the system registry on which the provider agent [ ] is executed, will be shown that the received request was the one with altered semantics

Fig. 16. 'Attack on SOAP Message Integrity' misuse case specification with UMLPWSSec.

ARTICLE IN PRESS

Table 10
Map Risk view showing the WS-BTSConsumer and TNT message assets.

Asset	Threat	F	Security dimensions						
			I	C	D	A_S	A_M	T_S	T_M
WS-BTSConsumer	1.1.1.1.1.1	AF	M, 50%, L	L, 50%, VL		H, 100%, H	M, 100%, M		
	1.1.1.1.1.2	AF	M, 60%, L	L, 5%, VL		H, 10%, VL	M, 10%, VL	M , 0% , VL	M, 0%, VL
	1.1.1.1.1.3	AF			M, 10%, VL				
	1.1.1.1.1.4	AF	M, 0%, VL	L, 0%, VL					
	1.1.1.1.1.5	AF			M, 0%, VL				
	1.1.1.1.1.6	AF	M, 10%, VL	L, 5%, VL		H, 5%, VL	M, 5%, VL		
	1.1.1.1.1.7	AF	M, 0%, VL						
	1.1.1.1.1.8	AF	M, 100%, M	L, 10%, VL		H, 60%, M			
TNT message	1.4.1.1.1.1	AF	M, 100%, L	L , 50%, VL		H, 60%, M			
	1.4.1.1.1.2	AF							
	1.4.1.1.1.3	AF							
	1.4.1.1.1.4	AF	M, 100%, M						
	1.4.1.1.1.5	AF	M, 100%, M						
	1.4.1.1.1.6	AF	M, 100%, M						
	1.4.1.1.1.7	AF		L, 50%, VL					
	1.4.1.1.1.8	AF	M, 100%, M		M, 10%, VL				
	1.4.1.1.1.9	AF					M, 100%, M		

Table 11
Frequency scale used in [55].

Value	Meaning
DF	Daily frequency
MF	Monthly frequency
AF	Annual frequency
SF	Every few years frequency

- **Critical Security Zone** (see Table 16): During the design of the architecture, we agreed that all software systems which need any kind of interaction with the transaction legacy system of the Organization selling the Products should be hosted in their own security area/domain. The security policy of the security area gave place to the security requirements shown in Tables 17 and 18 as a result of factoring the security requirements common to WS-BTSConsumer and WS-BTSProvider agents.
- Security kernel (see Table 19): A master security kernel for the security zone SecZone-1-1 was defined. Each security kernel manages a set of security services, derived from the application of a certain set of architecture patterns, and whose purpose is to help a potential set of business WS. Each WSSecKern will support one or more security services implemented through one or more concrete security mechanisms in the manner of standards or specifications that will be defined in the WSSecTech subprocess.
- Security services (see Table 20): Various security services were defined within the SecZone-1-1 Security Zone and under the management of the WSSecKern-1-1-M-1 Security Kernel. Among these Security Services, we can highlight that which was derived from the application of the QoP security architecture pattern summarized in Table 20 (see Fig. 22).

7.2.3. Security policy review

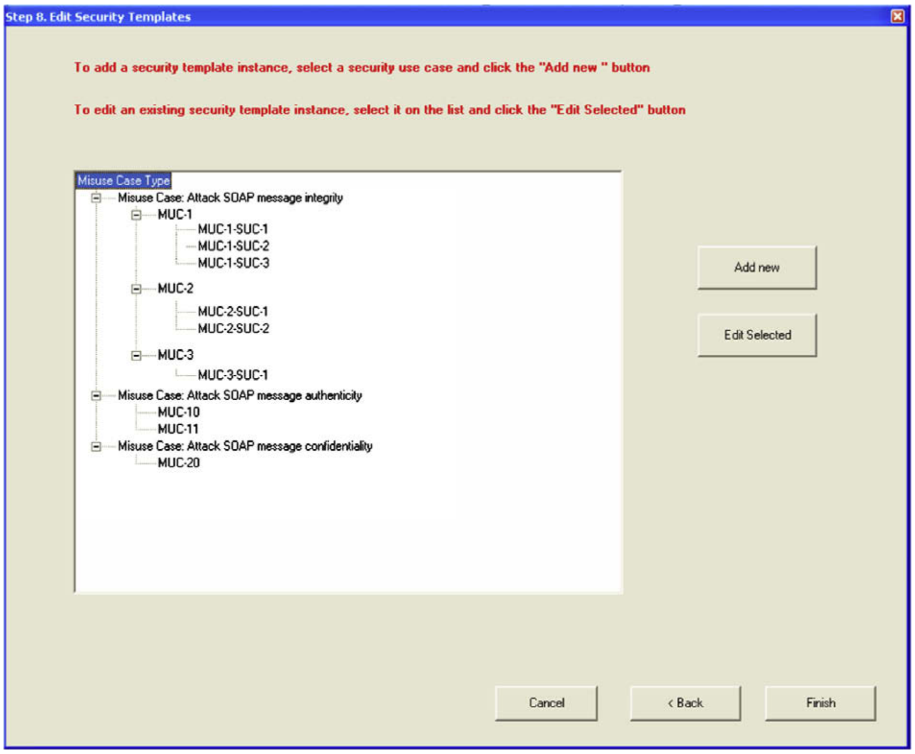
The input to this activity was the SASSec initial document which included the initial security architecture of the system. The logical security elements defined in this architecture were then used to create each of their security policies.

During this activity, the WS-based systems architect and the group responsible for security specified the policies of the various security elements found during the two previous activities. These security policies were:

- Security policy at the security requirement level. This policy groups the set of security requirements covered by the security architecture in such a way that these requirements can be

Table 12
Risk Map view of the WS-BTSConsumer and TNT message assets.

Asset	Threat	F	Security dimensions						
			I	C	A	SUA	MSA	ST	DMT
WS-BTSConsumer	1.1.1.1.1.1	AF	L	VL		H	M		
	1.1.1.1.1.2	AF	L	VL		VL	VL	VL	VL
	1.1.1.1.1.3	AF			VL				
	1.1.1.1.1.4	AF	VL	VL					
	1.1.1.1.1.5	AF			VL				
	1.1.1.1.1.6	AF	VL	VL		VL	VL		
	1.1.1.1.1.7	AF	VL						
	1.1.1.1.1.8	AF	M	VL		M			
TNT message	1.4.1.1.1.1	AF	L	VL		M			
	1.4.1.1.1.2	AF							
	1.4.1.1.1.3	AF							
	1.4.1.1.1.4	AF	M						
	1.4.1.1.1.5	AF	M						
	1.4.1.1.1.6	AF	M						
	1.4.1.1.1.7	AF		VL					
	1.4.1.1.1.8	AF	M		VL				
	1.4.1.1.1.9	AF					M		



Step 8. Edit Security Templates

To add a security template instance, select a security use case and click the "Add new " button

To edit an existing security template instance, select it on the list and click the "Edit Selected" button

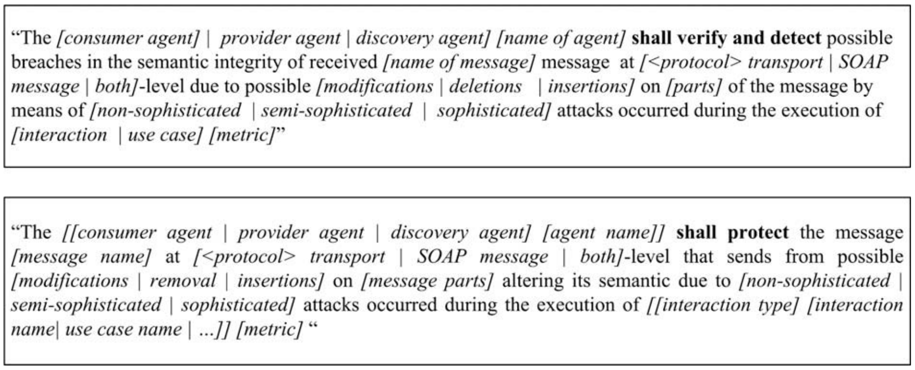
Misuse Case Type

- Misuse Case: Attack SOAP message integrity
  - MUC-1
    - MUC-1-SUC-1
    - MUC-1-SUC-2
    - MUC-1-SUC-3
  - MUC-2
    - MUC-2-SUC-1
    - MUC-2-SUC-2
  - MUC-3
    - MUC-3-SUC-1
- Misuse Case: Attack SOAP message authenticity
  - MUC-10
  - MUC-11
- Misuse Case: Attack SOAP message confidentiality
  - MUC-20

Fig. 17. Instantiation of abstract security use with UMLPWSSec.

Table 13
Security Use Case 'Protect TNT SOAP Message Integrity'.

Security Use Case Name: Protect TNT SOAP Message Integrity				
ID	MUC-1-SUC-1			
Related misuse cases	MUC-1			
Related security requirement templates	SRT-1-1, SRT-1-2			
Preconditions	(1) External attackers have physical access to message (2) External attackers have full-knowledge of the part of the message they intend to alter			
Attacker interactions	System requirements			
	Interactions of sender WS agent	Actions of sender WS agent	Interactions of recipient WS agent	Actions of recipient WS agent
	The sender WS-BTSConsumer consumer agent constructs a TNT SOAP message and sends it to the recipient WS agent.	The sender WS-BTSConsumer consumer agent must try to ensure that the modifications that could affect the TNT message during its transit are obviously detected by the recipient WS-BTSProvider provider agent.		
The external attacker intercepts the TNT message and modifies the part corresponding to the benificiary's bank account number at the level of the SOAP message layer.				
			The recipient WS-BTSProvider provider agent receives the corrupt TNT message	
				The receptor WS-BTSProvider provider agent receives the TNT message that was altered and discards it. It also sends an alarm to the Monitoring System and a critical message to the register of the system
Postconditions	(1) The WS-BTSProvider agent will have sent an alarm to the monitoring system and will have sent a critical message to the register of the system as a consequence of having detected that the message was altered in transit			



"The *[consumer agent]* | *provider agent* | *discovery agent] [name of agent]* **shall verify and detect** possible breaches in the semantic integrity of received *[name of message]* message at *[<protocol> transport* | *SOAP message* | *both]*-level due to possible *[modifications* | *deletions* | *insertions]* on *[parts]* of the message by means of *[non-sophisticated* | *semi-sophisticated* | *sophisticated]* attacks occurred during the execution of *[interaction* | *use case] [metric]*"

"The *[[consumer agent* | *provider agent* | *discovery agent] [agent name]]* **shall protect** the message *[message name]* at *[<protocol> transport* | *SOAP message* | *both]*-level that sends from possible *[modifications* | *removal* | *insertions]* on *[message parts]* altering its semantic due to *[non-sophisticated* | *semi-sophisticated* | *sophisticated]* attacks occurred during the execution of *[[interaction type] [interaction name*| *use case name* | *...]] [metric]* "

Fig. 18. WS security requirement templates SRT-1-1 and SRT 1-2.

referenced by other security policies with a more specific purpose.

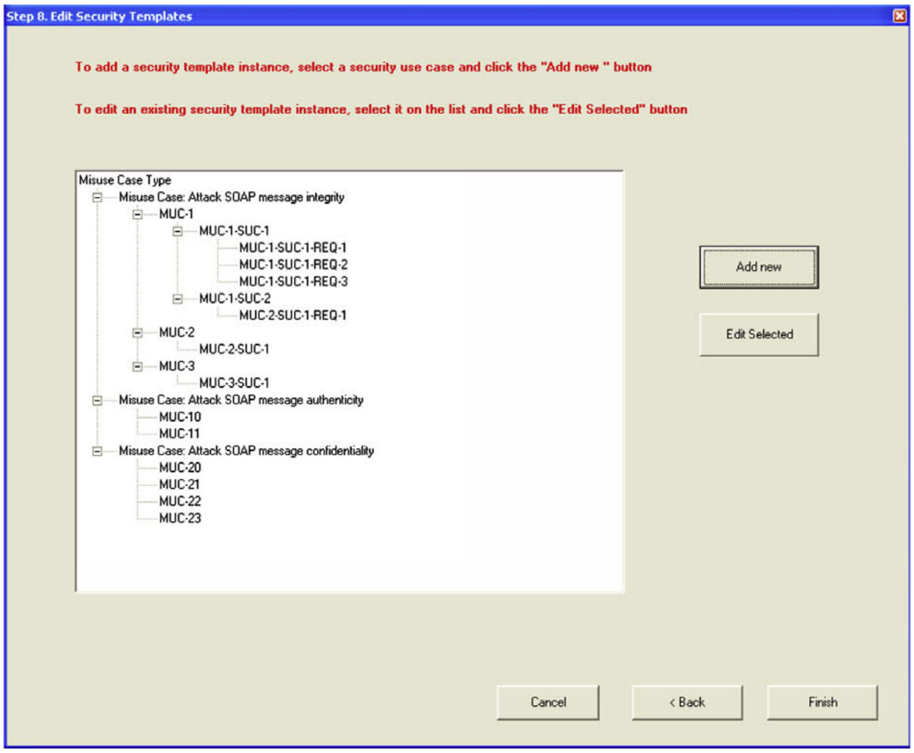
- Security Policy at the level of SecSrv-QoP-1 security service. This security policy, described through version 1.1. of the WS-Policy framework [46], defines the types of security requirements covered by the service, together with the mechanisms used to solve them and the permitted configurations.
- Security Policy at the level of the WSSecKern-1-1-M-1 security kernel. This security policy, described through version 1.1 of the WS-Policy framework, indicates which security requirements are covered by the security kernel, what services are used to solve them, what security services are associated with it and what configuration it allows for the use of the mechanisms that implement those security services.
- Security Policy of the WS-BTSConsumer agent. This describes the security configuration, described through version 1.1 of the WS-Policy framework, required for the input and output messages of the WS-BTSConsumer agent.
- Security Policy of the WS-BTSProvider. This describes the security configuration, described through version 1.1 of the WS-Policy framework, required for the input and output messages of the WS-BTSProvider agent.

The WSE (Web Services Enhancement) Policy Advisor Microsoft Tool [47] was also used to audit the security defined for each of the WS-BTSConsumer and WS-BTSProvider agents' security policies.

This activity's output was the SASSec document which was completed with a description of each security policy, in its normalized version, and was defined to include the obtained security audit report.

7.2.4. A2.4. Security architecture specification

The input of this activity was the SASSec document in its draft version. The WS-based systems architect created an initial version of the SASSec security architecture document in which all the information explained in detail in the previous points was col-

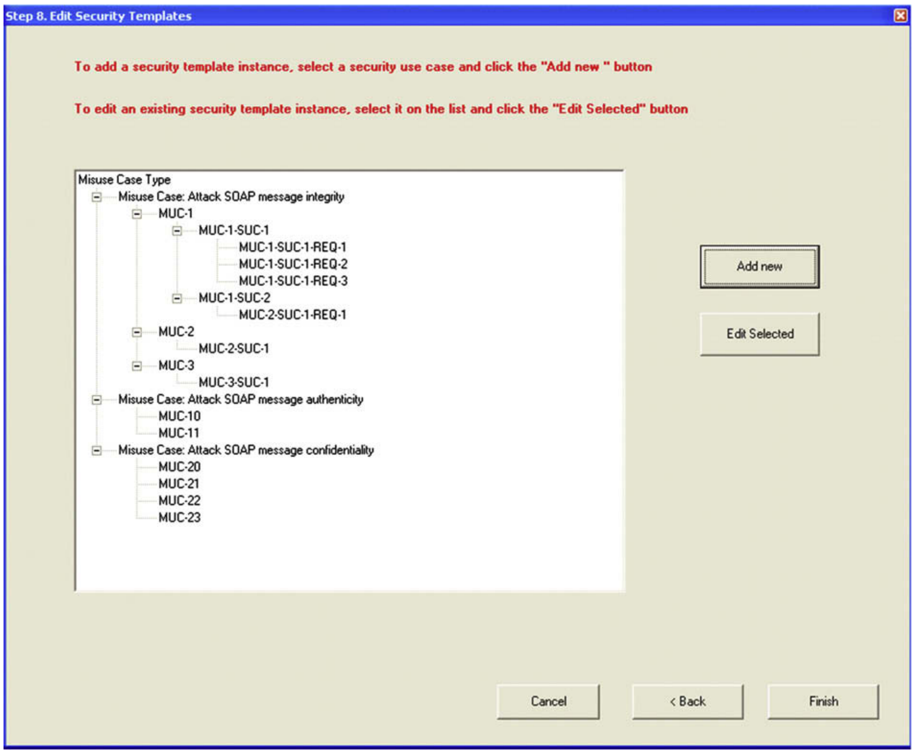


To add a security template instance, select a security use case and click the "Add new " button

To edit an existing security template instance, select it on the list and click the "Edit Selected" button

```
Misuse Case Type
  Misuse Case: Attack SOAP message integrity
    MUC-1
      MUC-1-SUC-1
        MUC-1-SUC-1-REQ-1
        MUC-1-SUC-1-REQ-2
        MUC-1-SUC-1-REQ-3
      MUC-1-SUC-2
        MUC-2-SUC-1-REQ-1
    MUC-2
      MUC-2-SUC-1
    MUC-3
      MUC-3-SUC-1
  Misuse Case: Attack SOAP message authenticity
    MUC-10
    MUC-11
  Misuse Case: Attack SOAP message confidentiality
    MUC-20
    MUC-21
    MUC-22
    MUC-23
```

Fig. 19. Security requirement traceability with UMLPWSSec.



To add a security template instance, select a security use case and click the "Add new " button

To edit an existing security template instance, select it on the list and click the "Edit Selected" button

```
Misuse Case Type
⊟ Misuse Case: Attack SOAP message integrity
    ⊟ MUC-1
        ⊟ MUC-1-SUC-1
            MUC-1-SUC-1-REQ-1
            MUC-1-SUC-1-REQ-2
            MUC-1-SUC-1-REQ-3
        ⊟ MUC-1-SUC-2
            MUC-2-SUC-1-REQ-1
    ⊟ MUC-2
        MUC-2-SUC-1
    ⊟ MUC-3
        MUC-3-SUC-1
⊟ Misuse Case: Attack SOAP message authenticity
    MUC-10
    MUC-11
⊟ Misuse Case: Attack SOAP message confidentiality
    MUC-20
    MUC-21
    MUC-22
    MUC-23
```

Fig. 20. Integrity security requirements.

Table 14
Security requirement specification related to the Protect SOAP TNT Message Integrity written from the WS Consumer Agent's perspective.

Id	MUC-1-SUC-1-REQ-1
Quality factor	Security
Quality subfactor	Integrity
Abstraction level	Application
Security use case	MUC-1-SUC-1
Protected asset	TNT Message
Priority	High
Criticality	High
Feasibility	High
Risk	M (see Table 11)
Source	Sales organization
Includes	SRQ-14, SRQ-15
Excludes	–
See Fig. 21	

lected in a summarized version. This document follows the basic sections of the functional architecture specification document conforming to 1471-2000, and aggregates the new actors, viewpoints, views and security view packages. The output of this activity was therefore the initial, although not yet validated, version of the SASSec document.

7.2.5. A2.5. Security architecture validation

The security architecture specification document (SASSec) was used as input to carry out the validation process of the security architecture through review meetings held by those involved. These people validated that the security architecture was in accordance with their needs, and also that it could be integrated into the existing functional architecture without incidence. It was furthermore verified that such architecture was not in conflict with any other non-functional requirement type.

Table 15
Security requirement specification related to the TNT Message confidentiality written from the WS Consumer Agent's perspective.

Id	MUC-20-SUC-1-REQ-1
Quality factor	Security
Quality subfactor	Confidentiality
Abstraction level	Application
Security use case	MUC-20-SUC-1
Protected asset	TNT message
Priority	High
Criticality	High
Feasibility	High
Risk	M (see Table 11)
Source	Sales organization
Includes	–
Excludes	–
The WS-BTSConsumer consumer agent will protect the SOAP TNT Message at the HTTP transport layer and SOAP message-level that it transmits, guaranteeing that both the beneficiary's bank account number and the cost of the purchase, can only be seen by the WS-BTSProvider consumer agent, thus resisting sophisticated attacks during the execution of the TNT interaction of the use case Perform Bank Transfer of the BTS business process in 99.9% of the transmissions that it performs	

7.3. Subprocess P3 – WSSECTECH

In this first iteration, the sole purpose of applying this subprocess was that of obtaining an initial candidate list of security standards to be used to implement the security architecture drafted in the previous section.

The set of types of security requirements solved by the security architecture (integrity, authenticity of data origin and confidentiality) was the input. The necessary security standards were then drawn from the decision table (based on [20,48] and our previous work in [45]), which relates types of security requirements to standards (see Table A-3). The set of security standards for WS used is shown in Table 21 and corresponds to the standards necessary for

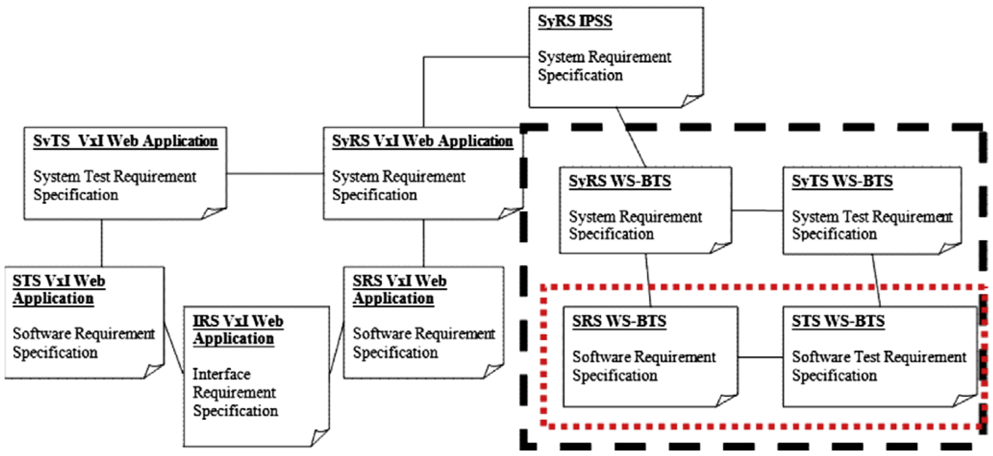

SyRS IPSS
System Requirement Specification
SvTS VxI Web Application
System Test Requirement Specification
SyRS VxI Web Application
System Requirement Specification
SyRS WS-BTS
System Requirement Specification
SvTS WS-BTS
System Test Requirement Specification
STS VxI Web Application
Software Requirement Specification
IRS VxI Web Application
Interface Requirement Specification
SRS VxI Web Application
Software Requirement Specification
SRS WS-BTS
Software Requirement Specification
STS WS-BTS
Software Test Requirement Specification

Fig. 21. Partial hierarchy of IPSS system's security requirement documentation.

Table 16
Critical Security Zone description.

Security zone	
ID	SecZone-1-1
Name	Critical Security Zone
Description	Security zone which hosts those services that must interact with the Central System to be able to fulfil the services they offer
Criticality	High
Security policy	Omitted due to reasons of privacy
ID master WSSecKern	WSSecKern-1-1-M-1
WS consumers/providers agents	WS-BTSConsumer
Security requirements addressed	SRQ-12, SRQ-13, SRQ-14, SRQ-15
Types of addressed security requirements	Integrity message origin authentication confidentiality message repetition availability

Table 17
Security requirement specification related to the integrity of the incoming messages of Security zone SecZone1-1.

Id	SRQ-33
Quality factor	Security
Quality subfactor	Integrity
Abstraction level	Application
Security use case	–
Protected asset	Security zone SecZone1-1
Priority	High
Criticality	High
Feasibility	High
Risk	–
Source	Sales organization
Includes	SRQ-12, SRQ-14
Excludes	–
100% of the incoming SOAP messages to the SecZone1-1 Security Zone must incorporate the security mechanisms necessary for guaranteeing the integrity of the information that they contain at the HTTPS transport layer and SOAP message-level	

Table 18
Security requirement specification related to the message authenticity of the incoming messages of Security Zone SecZone1-1.

Id	SRQ-34
Quality factor	Security
Quality subfactor	Integrity
Abstraction level	Application
Security use case	–
Protected asset	Security zone SecZone1-1
Priority	High
Criticality	High
Feasibility	High
Risk	High
Source	Sales organization
Includes	SRQ-15
Excludes	–
100% of the incoming SOAP messages in the SecZone1-1 security zone must incorporate the security mechanisms necessary for verifying the identity of their origin at the SOAP message-level	

the OASIS Web Services Security: SOAP Message Security 1.0 standard.

8. Lessons learned

Some of the most relevant lessons learned during the application of PWSSec in the case study presented in this paper are:

– Applying a security process such as PWSSec implies a strong and solid back up from the top management of the organization. Otherwise, its application will have a very high risk of failing. The scope of implementing PWSSec within an organization should be undertaken gradually, establishing a clear and well-defined scope (e.g. a low-risk mid-size IS in the first stages of its adoption).
– Given the size of PWSSec, it should be tailored, and the breadth and depth of its application should be thoroughly analyzed, previous to its real application. With the relatively small scope of the case study, we had to develop more than 200 security intermediate artifacts to complete all of the PWSSec subprocesses. In consequence, we assume that in systems with a broader scope, a careful analysis of the level of implementation of our approach

ARTICLE IN PRESS

Table 19
Description of the master security kernel hosted in the Critical Security Zone.

Security kernel	
ID	WSSecKern-1-1-M-1
Name	WSSecKern-CriticalSecurityZone
Type	Master
Description	Master security kernel for the security zone
Security zone ID	SecZone-1-1
IDs SecSrv	SecSrv-QoP-1
WS consumers/providers agents	WS-BTSConsumer
Security requirements addressed	SRQ-12, SRQ-13, SRQ-14, SRQ-15
Types of addressed security requirements	Integrity message origin authentication confidentiality message repetition availability

Table 20
Quality-of-Protection Security Service description.

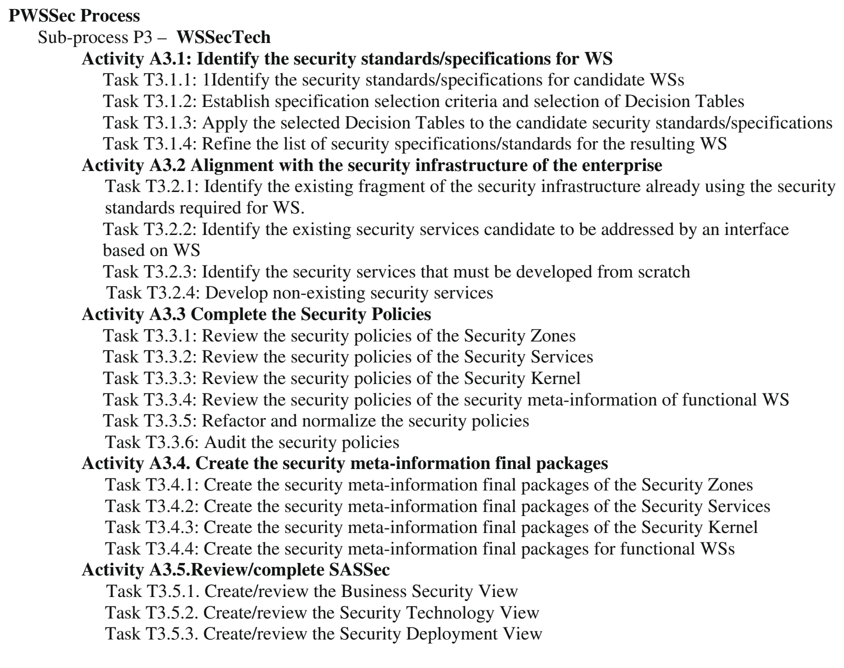
Security service	
ID	SecSrV-QoP-1
Name	Quality-of-protection security
Description	This security service's main purpose is to provide with 'quality of protection' to SOAP messages
Security policy	–
IDs WSSecKern	WSSecKern-1-1-M-1
Types of addressed security requirements	Integrity message origin authentication confidentiality

should be performed to ensure that the potential benefits outweigh the costs of applying it.

– Before applying WSSecReq subprocess to the present case study, templates of the threat trees associated with IBM WS-based application/business patterns were based solely on the threats defined by WS-I [41]. When applying the aforementioned case study we realized that it was possible to take advantage of the threat catalogue defined in MAGERIT2. We therefore arranged the threats in the MAGERIT2 catalogue in a tree-like form (as shown in Figs. 10 and 11) and then associated them with the corresponding IBM WS-based business/application pattern thereby obtaining a more formal and complete expression of threat trees.
– Adapting MAGERIT2 to WSSecReq subprocess was a smooth task given the high-abstract level in which WSSecReq activities have been defined. A priori, adapting other risk and analysis methodologies (e.g.: OCTAVE [49].) should be an easy task.
– When applying WSSecReq to the present case study we learned how WS-based systems can easily be broken down into their main assets, how these can be classified within the context of a security risk analysis and what dependencies can be defined among them (see Fig. 14). We believe that this effort can be fully reutilized in future security risk analyses of WS-based systems.
– As a lesson learned from applying WSSecReq to the case study, we incorporated a new artifact (*MAGERIT2* Profile) into the *E&A Repository*. While adapting *MAGERIT2* to WSSecReq security risk activities we identified, defined and incorporated the concept of Profile for Risk Analysis and Management as an extension and realization mechanism of risk-oriented tasks defined within the WSSecReq subprocess. Each Profile for Risk Analysis and Management defines how to apply certain risk analysis and management methodology in the context of the PWSSec process and, in particular, of its WSSecReq subprocess. Ongoing work to define a new Profile mapping risk-oriented tasks to the OCTAVE security risk is currently taking place. A profile must contain:

Table 21
Decision table of WS security standards.

Security requirement type	WS security standard
Confidentiality Message authentication Integrity	• W3C XML Exclusive Canonicalization • W3C XML Digital Signature • W3C XML Encryption • OASIS Web Service Security: SOAP Message Security 1.1 • OASIS Web Services Security: username token profile 1.1
Interoperability	WS-I Basic Security Profile 1.0
Security token management	OASIS WS-Trust Language 1.1



PWSSec Process
Sub-process P3 – **WSSecTech**
Activity A3.1: Identify the security standards/specifications for WS
Task T3.1.1: 1Identify the security standards/specifications for candidate WSs
Task T3.1.2: Establish specification selection criteria and selection of Decision Tables
Task T3.1.3: Apply the selected Decision Tables to the candidate security standards/specifications
Task T3.1.4: Refine the list of security specifications/standards for the resulting WS
Activity A3.2 Alignment with the security infrastructure of the enterprise
Task T3.2.1: Identify the existing fragment of the security infrastructure already using the security standards required for WS.
Task T3.2.2: Identify the existing security services candidate to be addressed by an interface based on WS
Task T3.2.3: Identify the security services that must be developed from scratch
Task T3.2.4: Develop non-existing security services
Activity A3.3 Complete the Security Policies
Task T3.3.1: Review the security policies of the Security Zones
Task T3.3.2: Review the security policies of the Security Services
Task T3.3.3: Review the security policies of the Security Kernel
Task T3.3.4: Review the security policies of the security meta-information of functional WS
Task T3.3.5: Refactor and normalize the security policies
Task T3.3.6: Audit the security policies
Activity A3.4. Create the security meta-information final packages
Task T3.4.1: Create the security meta-information final packages of the Security Zones
Task T3.4.2: Create the security meta-information final packages of the Security Services
Task T3.4.3: Create the security meta-information final packages of the Security Kernel
Task T3.4.4: Create the security meta-information final packages for functional WSs
Activity A3.5.Review/complete SASSec
Task T3.5.1. Create/review the Business Security View
Task T3.5.2. Create/review the Security Technology View
Task T3.5.3. Create/review the Security Deployment View

Fig. 22. Activities and tasks of WSSecTech subprocess.

ARTICLE IN PRESS

- Activity Matching Table: This table should reflect how activities from a particular risk and analysis management methodology match with those defined in the PWSSec process.
- Template of Threat Trees associated with every IBM WS-based application/business pattern.
- A method for assessing assets.
- A method for assessing threat impact and calculating risk.
- A method for assessing established countermeasures.

9. Research method

The research method used to define PWSSec was the Action-Research method. This method represents the manner in which groups of people prepare the conditions which are necessary to learn from their own experiences, and make these experiences available to others [50]. The Action-Research method has two aims: to generate benefits for the research "client" whilst simultaneously generating the relevant research knowledge [51]. Action-Research is oriented towards the production of new knowledge which is useful in practice, and which is obtained through the exchange of and/or search for solutions to real situations affecting a group of professionals [52].

Within the field of Information Systems, the research client is usually an organization to which the researcher gives services such as consultations or assistance in changing or developing software in exchange for access to data of interest to the research project and, in many cases, funding [51].

A research process using Action-Research is made up of groups of activities which form a characteristic cycle. Padak and Padak [53] identify the following steps, which should be followed in any research using this method:

(1) Planning: Identify the relevant questions, which will guide the research, which must be directly related to the object under investigation and to which answers can be found. This activity seeks alternative paths, lines to follow or the strengthening of something that already exists. The result is a clear definition of other problems or situations that must be dealt with.
(2) Action. Careful, deliberate and controlled variation of practice. A simulation or test of the solution is carried out. This is the moment at which the researcher acts on the reality actually.
(3) Observation. Collecting information and data, documenting what happens. This information may come from almost anywhere (bibliographies, measures, test results, observations, interviews, documents, etc.). This is also known as 'evaluation'.
(4) Share and analyze the results with other interested parties in order to pose new relevant questions and, as Wadsworth [54] adds, "to develop deeper understandings and more useful and more powerful theory about the matters we are researching, in order to produce new knowledge which can inform improved action or practice".

In the case of PWSSec, the Action-Research method has been applied in its participative variant, which is to say that the critical reference group (for whom the research in being carried out and which participates in the research process) puts into practice the recommendations made by the researcher, and shares the effects and results with her.

10. Conclusions

Security is a crucial aspect if WS-based systems are to be the 'de facto' solution for inter- and intra- integrating heterogeneous systems. For this to become a reality, a software engineering-based, security engineering-centered global approach must be defined. This approach should provide developers with all the activities, tasks, tools, security artifacts and organizational structures necessary to design a secure WS-based solution.

This issue has been addressed through the definition of the PWSSec process, which accomplishes security requirements engineering and the security architectural and design of WS-based systems.

In this paper, we have focused our discussion on a case study in which PWSSec was applied to a real system for the first time, and we have also presented a tool which supports the activities with regard to security requirements engineering. As a main consequence of applying PWSSec to the case study, we discovered that it is crucial to analyze the level of implementation of each of the subprocesses, artifacts and deliverables involved before their application. This level of implementation must balance the benefits of applying PWSSec and its direct costs. The scalability of our approach will be guaranteed in two ways: (i) by feeding the repositories specified in the subprocesses as PWSSec is applied in more case studies; (ii) by using the UMLPWSSec tool, thus providing automatic support when applying the subprocesses, and their activities, and when creating the different security artifacts required by PWSSec.

We are currently working to improve this process by applying it in more case studies and equipping the UMLPWSSec tool with all the security artifacts defined within the E&A Repository. We are also complementing PWSSec by defining the corresponding subprocesses in order to cover test deployment and IT governance disciplines.

Acknowledgements

This research is part of the ESFINGE (TIN2006-15175-C05-05) Project financed by the Spanish Ministry of Education and Science, and of the QUASIMODO (PAC08-0157-0668) Project, financed by the FEDER and the Regional Science and Technology Ministry of Castilla-La Mancha (Spain).

Appendix 1

CASE	Computer Aided Software Engineering
IPSS	Internet Product Selling System
IRS	Interface Requirement Specification
PWSSec	Process for Web Services Security
RNP	Reference Number Processing
SASSEC	Architecture Security Patterns section
SECZONE	Security Zone
SOA	Service Oriented Architecture
SRS	Software Requirement Specification
STS	Software Test Specification
SyRS	System Requirement Specification
SyTS	System Test Specification
TTR	Transfer Token Request
UML	Unified Modeling Language
UML QoS	UML Quality of Service
UMLPWSSec	UML for PWSSec
WS-BTS	Web Services Bank Transfer System
WSSECARCH	Web Services Security Architecture
WSSECREQ	Web Services Security Requirements
WSSECTECH	Web Services Security Technologies

References

[1] J. Zhang, Trustworthy web services: actions for now, IEEE IT Pro 7 (1) (2005) 32–36.

[2] C. Gutiérrez, E. Fernández-Medina, M. Piattini, A survey of web services security, in: Workshop on Internet Communications Security 2004 (WICS 2004), in Conjunction with the 2004 International Conference on Computational Science and Its Applications (ICCSA 2004), Springer-Verlag, Assisi (PG), Italy, 2004.
[3] S. Vinoski, WS-NonexistentStandards, IEEE Internet Comput. 8 (6) (2004) 94–96.
[4] C. Gutiérrez, E. Fernández-Medina, M. Piattini, PWSSec: Process for Web Services Security, in: IEEE International Conference on Web Services 2006, Chicago, USA, 2006.
[5] M. Endrei et al., Patterns: Service-Oriented Architecture and Web Services, 2004, p. 345.
[6] C. Gutiérrez, E. Fernández-Medina, M. Piattini, Web services enterprise security architecture: a case study, in: ACM Workshop on Security on Web Services, ACM Press, Fairfax, Virginia, USA, 2005.
[7] D. Remy, J. Rosenberg, Securing web services with WS-Security: demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption, SAMS, 2004, p. 408.
[8] M. Aiello, P. Giorgini, Applying the tropos methodology for analysing web services requirements and reasoning about qualities of services, UPGRADE 5(4) (2004) 20–26.
[9] K. Leune, M. Papazaglou, Specification and querying of security constraints in the EFSOC framework, in: International Conference on Service Oriented Computing, Willem-Jan van den Heuvel, New York City, USA.
[10] M. Deubler et al., Sound development of secure service-based systems, in: 2nd International Conference on Service Oriented Computing (ICSOC'04), ACM Press, New York, USA, 2004.
[11] M. Deubler et al., Towards a model-based and incremental development process for service-based systems, in: The IASTED Conference on Software Engineering (IASTED SE 2004), Innsbruck, Austria, 2004.
[12] Y. Nakamura et al., Model-driven security based on a web services security architecture, in: IEEE International Conference on Services Computing (SCC'05), IEEE Computer Society, Orlando, Florida, USA, 2005.
[13] N. Nagaratnam et al., Business-driven application security: from modeling to managing secure applications, IBM Syst. J. 44 (4) (2005) 847–867.
[14] M. Breu et al., Web service engineering – advancing a new software engineering, in: 5th International Conference on Web Engineering (ICWE'05), Springer, Sydney, Australia, 2005.
[15] M. Hafner, R. Breu, Security Engineering for Service-Oriented Architectures, Springer, 2009. p. 248.
[16] IEEE, IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems – Description, 2000.
[17] E.B. Fernandez, P. Cholmondeley, O. Zimmermann, Extending a secure system development methodology to SOA, in: DEXA '07: Proceedings of the 18th International Conference on Database and Expert, IEEE Computer Society, Regensburg, Germany, 2007.
[18] N.A. Delessy, E.B. Fernandez, A pattern-driven security process for SOA applications, in: SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing, ACM press, Fortaleza, Ceara, Brazil, 2008.
[19] A. Buecker et al., Understanding SOA Security Design and Implementation, in: IBM Redbooks, IBM, 2007.
[20] A. Singhal, T. Winograd, K. Scarfone, Guide to secure web services, in: Recommendations of the National Institute of Standards and Technology, National Institute of Standards and Technology (NIST), 2007.
[21] C. Gutierrez, M. Piattini, E. Fernandez-Medina, Web services security, in: E. Ferrari, B. Thuraisingham (Eds.), Web and Information Security, IRM Press, 2005, p. 318.
[22] M. O'Neill et al., Web Services Security, McGraw-Hill, Berkeley, California, 2003.
[23] E. Ferrari, B. Thuraisingham, Security and privacy for web databases and services, in: Extending Database Technology (EDBT'04), Springer-Verlag, Heraklion, Crete, Greece, 2004.
[24] R. Bhatti et al., XML-Based Specification for Web Services Document Security, IEEE Comput. 37 (4) (2004) 41–49.
[25] E. Bertino, S. Castano, E. Ferrari, On specifying security policies for web documents with an XML-based language, in: Sixth ACM Symposium on Access Control Models and Technologies (SACMAT'01), ACM Press, Chantilly, Virginia, USA, 2001.
[26] L.-J. Zhang, J. Zhang, J.-Y. Chung, An Approach to help select trustworthy web services, in: E-Commerce Technology for Dynamic Business, IEEE International Conference on CEC-East'04, Beijing, China, 2004.
[27] S. Chang, Q. Chen, M. Hsu, Managing security policy in large distributed web services environment, in: 27th Annual International Computer Software and Applications Conference (COMPSAC'03), Dallas, Texas, 2003.
[28] IBM and Microsoft, Security in a Web Services World: A Proposed Architecture and Roadmap, 2002.
[29] WS-I, Basic Security Profile Version 1.0, Working Group Draft, 2004.
[30] LibertyAllianceProject, Liberty ID-FF Architecture, Overview, v1.2, 2003.
[31] W3C, Web Services Architecture, 2004.
[32] R.T. Fielding, Architectural styles and the design of network-based software architectures, in: Software Research Group, University of California, Irvine, 2000.
[33] T. Erl, SOA Principles of Service Design, Prentice Hall/PearsonPTR, 2007, p. 573.
[34] B.W. Boehm, A spiral model of software development and enhancement, IEEE Comput. (1988) 61–72.
[35] F. López et al., MAGERIT – Versión 2, Metodologías de Análisis y Gestión de Riesgos de los Sistemas de Información, I-Método, Ministerio de Administraciones Públicas, Madrid, 2005, p. 154.
[36] CCN-CERT, Últimos avances en ciberseguridad (9th NATO cyberdefense workshop), Revista auditoria y Seguridad, 2008 (23), pp. 70–71.
[37] OECD, The promotion of a culture of security for information systems and networks in OECD countries, Organisation for Economic Co-operation and Development, 2005.
[38] P. Krutchen, The 4 + 1 view model of software architecture, IEEE Softw. (1995) 42–50.
[39] S.H. Houmb, J. Jürjens, Developing secure networked web-based systems using model-based risk assessment and UMLSec, in: 10th Asian–Pacific Software Engineering Conference (APSEC'03), IEEE Computer Society, Chiangmai, Thailand, 2003.
[40] OMG, UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, 2004.
[41] WS-I, Security Challenges, Threats and Countermeasures, Versión 1.0, 2005, WS-I.
[42] A.P. Moore, R.J. Ellison, R.C. Linger, Attack modelling for information security and survivability, in: Survivable Systems, Software Engineering Institute, 2001.
[43] C. Gutiérrez et al., Security requirements for web services based on SIREN, in: Symposium on Requirements Engineering for Information Security, Paris, France, 2005.
[44] A. Toval et al., Requirements reuse for improving information systems security: a practitioner's approach, Require. Eng. J. 6 (4) (2001) 205–219.
[45] G. Rosado et al., Security patterns and security requirements for web services, Internet Res. 16 (5) (2006).
[46] VeriSign et al., Web Services Policy Framework (WS-Policy), 2004.
[47] K. Bhargavan et al., An advisor for web services security policies, in: 2005 Workshop on Secure Web Services, ACM Press, Fairfax, VA, USA, 2005, pp. 1–9.
[48] H.L. Levinson, L. O'Brien, Acquiring evolving technologies: web services standards, in: Acquisition Support Program, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2006, p. 64.
[49] C.J. Alberts et al., OCTAVE Framework, Version 1.0, in: Networked Systems Survivability Program, Carnegie Mellon, SEI, 1999, p. 84.
[50] R. McTaggart, Principles of participatory action research, Adult Educ. Quart. 41 (3) (1991).
[51] N. Kock, F. Lau, Information systems action research: serving two demanding masters, Inform. Technol. People (special issue on Action Research in Information Systems) 14(1) (2001) 6–11.
[52] D. Avison et al., Action research, Commun. ACM 42 (1) (1999) 94–97.
[53] N. Padak, G. Padak, Guidelines for Planning Action Research Projects, 1994.
[54] Y. Wadsworth, What is participatory action research?, Action Res Int. 2 (1998).
[55] F. López et al., MAGERIT – Versión 2, Metodologías de Análisis y Gestión de Riesgos de los Sistemas de Información, III -Guía de Técnicas, Ministerio de Administraciones Públicas, Madrid, 2005, p. 154.