Eduardo Fernandez Medina

De: Neil Maiden [cc559@soi.city.ac.uk]
Enviado el: lunes, 15 de agosto de 2005 9:50
Para: g.dobson@lancs.ac.uk; r.lock@lancs.ac.uk; is@comp.lancs.ac.uk; sesmaeil@cs.uwaterloo.ca; nday@cs.uwaterloo.ca; fmavaddat@cs.uwaterloo.ca; carlos.gutierrez@stl.es; Eduardo.FdezMedina@uclm.es; Mario.Piattini@uclm.es; yamamoto.kouji@jp.fujitsu.com; ohashi.kyoko@jp.fujitsu.com; munakata.kazuki@jp.fujitsu.com; r.yamamoto@jp.fujitsu.com; phm@cetic.be; cp@cetic.be; perini@itc.it; susi@itc.it; jm@cs.toronto.edu; sawyer@comp.lancs.ac.uk; hutchinj@comp.lancs.ac.uk; walkerdi@comp.lancs.ac.uk; klaus.schmid@iese.fraunhofer.de; michael.eisenbarth@iese.fraunhofer.de; grund@informatik.uni-kl.de; yws01@ecs.soton.ac.uk; cw2@ecs.soton.ac.uk; lg3 @ecs.soton.ac.uk; gbw@ecs.soton.ac.uk; xzhu@soi.city.ac.uk; saraj@soi.city.ac.uk; kzachos@soi.city.ac.uk; n.a.m.maiden@city.ac.uk; c.b.haley@open.ac.uk; p.sawyer@lancaster.ac.uk; xzhu@soi.city.ac.uk; r.lock@lancs.ac.uk; s.v.jones@city.ac.uk; yws01@ecs.soton.ac.uk; walkerdi@comp.lancs.ac.uk; a.ivanovic@utwente.nl; goluwanifise@yahoo.com; yamamoto.kouji@jp.fujitsu.com; tahara@nii.ac.jp; baresi@elet.polimi.it; segreteria-sra@itc.it; silvestris@itc.it; N.A.M.Maiden@city.ac.uk; ludovic.duchemin@wanadoo.fr; emmanuel.freyd@free.Fr; melofr@wanadoo.fr; guillaume@gogo.fr; gregory.guernevel@9online.fr; eduardo.fdezmedina@uclm.es; ggrau@lsi.upc.edu; Michael.Eisenbarth@iese.fraunhofer.de; sesmaeil@cs.uwaterloo.ca; liaskos@cs.toronto.edu; ozgul_gunduz@yahoo.fr; markcollins@eircom.net; virginiedhouibi@yahoo.fr
Asunto: SOCCER Workshop Timetable

Dear colleagues

Please find attached the agenda for the SOCCER workshop on 30th September 2005.

09.00-09.30 Introduction (Neil Maiden & Luciano Baresi)

09.30-10.30 Service specification

- Quality of Service Requirements Specification Using an Ontology
  Glen Dobson, Russell Lock, and Ian Sommerville
  Computing Department, Lancaster University (UK)

- Faceted Service Specification
  Pete Sawyer, John Hutchinson, James Walkerdine, and Ian Sommerville
  Computing Department, Lancaster University (UK)

10.30-11.00 Coffee break

11.00-12.30 Requirements elicitation
- A CRUD Approach to Requirements Analysis for Service-oriented System Development with Existing Systems Kouji Yamamoto, Kyoko Ohashi, Kazuki Munakata, and Rieko Yamamoto IT Core Laboratory, Fujitsu Laboratories Ltd. (Japan)

- Tropos Design Process for Web Services Anna Perini, Angelo Susi, and John Mylopoulos ITC-IRST, Trento (Italy)

- Web services-based security requirement elicitation C.Gutiérrez, E. Fernández-Medina and M. Piattini Departamento de Informática, Universidad de Castilla-La Mancha, Ciudad Real (Spain)

12.30-14.00 Lunch

14.00-15.30 Challenges and Case studies

- A Scenario and Goal Based Approach for Guaranteeing Quality of Service for Negotiated GRID Service Level Agreements: an Experience Report Philippe Massonet and Christophe Ponsard CETIC research center (Belgium)

- From Requirements Engineering to Knowledge
Engineering: Challenges in Adaptive Systems Klaus Schmid (1), Michael Eisenbarth (1)

Mathias Grund (2)
(1) Fraunhofer IESE (2) AG Software Engineering University of Kaiserslautern (Germany)

- Towards a Collaborative Orthopaedics Research Environment Y. W. Sim, C. Wang, L. Gilbert, G. B. Wills Electronics and Computer Science, University of Southampton (UK)

15.30-16.00 Coffee break

16.00-17.00 Service discovery

- Specifying Search Queries for Web Service Discovery Shahram Esmaeilsabzali Nancy A. Day Farhad Mavaddat School of Computer Science, University of Waterloo (Canada)

- Applying Patterns in Service Discovery Xiaohong Zhu, Neil Maiden, Sara Jones, Konstantinos Zachos Centre HCI Design, City University (UK)

17.00-17.30 Wrap-up and final discussion (Luciano Baresi and Neil Maiden)

Therefore, each presentation should last 20 minutes, and allow 10 minutes for questions.

The workshop will be held at:
IAE (Institut d'Administration des Entreprises) Paris
21 rue Broca, 75005 Paris
tel : +33 (0)1.53.55.28.00

Access:
Métro : Gobelins, Censier-Daubenton
Bus : 91 - 83 - 27 - 21 - 47
Maps, journey calculators and bus/subway indications can be found on www.ratp.fr, see "international passengers" button on top of the window

Each room has :
- a beamer & whitescreen (PC not provided)
- a whiteboard and color pens
- an overhead projector (no slide or pens)
- internet connections (but cables are not provided)

Hence we anticipate each presentation to be shown from the presenter's laptop. If any presenters have a problem, please contact as soon as possible.

Neil Maiden, Luciano Baresi, Xavier Franch.
--

Professor Neil Maiden
Head of Centre
Centre for HCI Design
City University
Northampton Square
London EC1V OHB

Tel: +44-20-7040-8412
Fax: +44-20-7040-8859
E-Mail: N.A.M.Maiden@city.ac.uk
http://hcid.soi.city.ac.uk/people/Neilmaiden.html

James Robertson and Neil Maiden give their ground-breaking tutorial

Creative Requirements - Invention and its Role in Requirements Engineering

at the RE'05 Conference, Paris, France. August 29 2005.
Come and discover how to invent better software http://crinfo.univ-paris1.fr/RE05/tutorial.html

Web services-based security requirement elicitation

C.Gutiérrez [1], E. Fernández-Medina[2] and M. Piattini[2]
(1) STL, Madrid (SPAIN), carlos.gutierrez@stl.es
(2) Alarcos Research Group. Universidad de Castilla-La Mancha.
Paseo de la Universidad 4, 13071, Ciudad Real. (SPAIN). Tel: 34 926 29 53 00
{Eduardo.FdezMedina, Mario.Piattini}@uclm.es

Abstract

Web services (WS, hereafter) paradigm has attained such a relevance in both the academic and the industry world that the vision of the Internet has evolved from being considered as a mere repository of data to become the underlying infrastructure on which complex business processes and alliances among organizations are deployed. Security is a key aspect if WS are to be generally accepted and adopted. In fact, over the past years, the most important consortiums of the Internet, like IETF, W3C or OASIS, have produced a huge number of WS-based security standards.

Despite this spectacular growth, there does not exist a development process that facilitates the systematic integration of security into all stages of WS-based software development life-cycle. Eventually, this process should guide WS-based software developers in the specification of WS-based security requirements, the design of WS-based security architectures, and the deployment of the most suitable WS security standards. In this article, we will briefly present a process of this type, named PWSSec (Process for Web Services Security), and the artifacts used during the elicitation activity, which pertains to the stage WSSecReq devoted to produce a WS-based security requirement specification.

1. Introduction

Security is a main concern when developing systems whose operational infrastructure is based on a public network such as the Internet.

WS-based systems are based on Internet protocols so security should be one of the main issues to be addressed when designing software based on this paradigm.

A huge number of WS-based security standards have been developed by a numerous set of diverse consortiums. A great effort and a solid background in computational security theory are necessary in order to obtain an in-depth knowledge of all of them. In addition, knowing what specific set of WS standards should be used in a certain WS-based system requires a previous knowledge of the security requirements that the security mechanisms specified in those standards will address.

In consequence, one of the major problems that developers have to deal with is to come up with a complete specification of the WS-based security requirements of their WS-based systems.

In order to solve this problem, we have defined PWSSec (Process for Web Services Security) process. This process consists of 3 main stages. The first stage, named WSSecReq (Web Services Security Requirements) is aimed at producing the above-mentioned WS-based security requirements specification. In particular, its first activity, named elicitation, applies a set of reusable artifacts that guides developers in the task of identifying security requirements from the piece of functionality whose security we want to analyze.

The main purpose of this article is to describe this set of artifacts showing how they can be used in a coordinated way to specify, in a systematic way, the security requirements of a certain WS-based system.

The rest of the article is organized as follows: in section 2, an overview of PWSSec process will be presented; in section 3, we will offer a complete description of the mentioned artifacts; and in section 4, conclusions as well as future research will be proposed.

2. PWSSec - Process for WS Security

PWSSec [8] has been created to facilitate and orientate the development of WS-based security systems so that a complementary stage comprising security [4] could be easily integrated into each one of the traditional stages for the construction of this kind of systems [3].

Figure 1 illustrates the stages in which PWSSec is structured into.

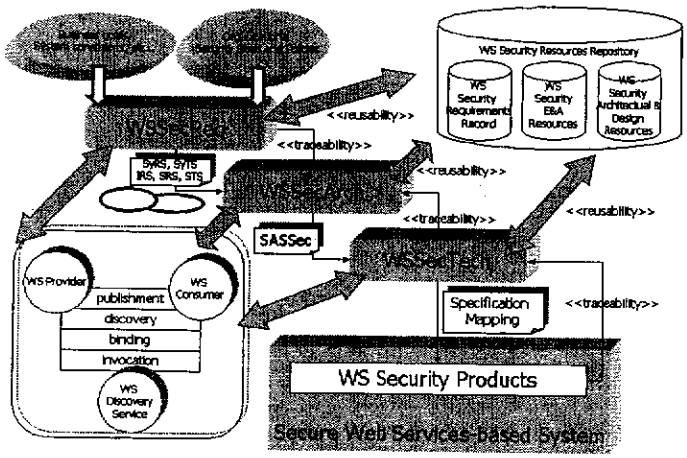

Figure 1. Stages and products in the PWSSec development process.

Each one of the stages defined in PWSSec describes its inputs, outputs, activities, actors and, in some cases, there are also guides, tools and techniques which complement, improve and facilitate the set of activities developed within these stages.

The WSSecReq stage's main purpose is to produce, by means of a systematic approach, a specification (or a part of it) of the security requirements of the WS-based system.

The WSSecArch stage is aimed at allocating the security requirements specified in the previous section to a WS-based security architecture. This security architecture will be equipped with the necessary security architectural mechanisms to achieve the considered security requirements.

The WSSecTech stage's main objective is to identify the set of WS-based security standards that will implement the architectural security mechanisms identified in the previous stage.

3. Elicitation in WSSecReq

In this section, we will explain all the security artifacts which the elicitation activity of the WSSecReq stage is based on.

Sometimes, we will present concrete examples where these artifacts are applied in practice. The examples of concrete artifacts shown here are based on the classical use case 'Place Order'. In this use case, a WS-based system of a retailer organization (primary actor) and a WS-based system of its supplier organization (secondary actor) [3] participate. This use case consists of one request/reply message interaction between the WS-based systems of both organizations. When the WS-based retailer system detects that any of its products is out-of-stock, it sends a request (and it gets blocked until a response is received) of stock replenishment to the WS-based system of the supplier organization.

3.1 WSSecReq overview

Two main principles have been considered in the definition of WSSecReq: reusability and traceability. Product reusability is achieved by defining two repositories: i) *WS Security E&A Resources*, that contains all the abstract artifacts being used during the elicitation activity (we will explain these artifacts in the following section); ii) *WS Security Requirements Record* that contains a set of WS-specific security requirement templates that can be applied to WS-based systems within diverse domains [14]. Both repositories are constantly being brought up-to-date.

On the other hand, product traceability is addressed by means of a coordinated and reasoned use of a set of security artifacts. These artifacts and their application will be detailed in the following section.

The input of the WSSecReq stage consists of:

1. A specification of the piece of software functionality whose security will be analyzed. WSSecReq treats security analysis as a micro-process which is performed at each level of abstraction and for each increment [1]. As we are dealing with WS-specific security requirements, the core artifacts will mainly belong to the system and to the application architecture level of abstraction, i.e. WS and their interactions. However, as we will see later on, WSSecReq may also be used to analyze security at higher levels of abstraction, , for instance, the business level one. For example, WSSeqReq stage's input could be specified at a low

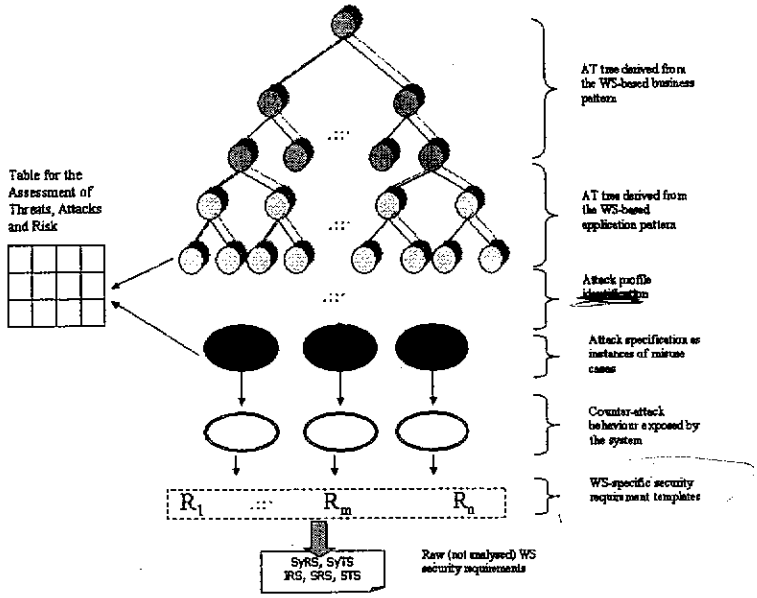

Figure 2. Coordination of products in the elicitation activity of the WSSecReq stage.

level of abstraction (application architecture) when it is composed of a (small) set of WS and operations which are within the scope of the current iteration (in this case the "security enhanced core artifacts" will be the WS and their interactions). On the other hand, it could be specified as a set of high-level functional requirements that describe how a group of responsibilities should be addressed by the WS-based system.

2. The business and security goals defined for the system, as well as the part of the organizational security policy that, in our opinion, can have an impact on the system design.

This stage defines four activities [8]: elicitation, analysis, specification, verification and validation.

In this article, we will focus on the *elicitation* activity and, in particular, on the artifacts involved. The activity of elicitation will be supported by a detailed study of the security of each WS identified and considered in the current iteration.

The activity of elicitation combines concepts derived from the risk analysis and management methodologies (in particular the process known as Operationally Critical Attack, Asset, and Vulnerability Evaluation ^{SM} (OCTAVE) [6]) with techniques that enable security requirements reusability [7, 14].

During this activity, several artifacts are considered and coordinated so that both elicitation and traceability of the security requirements can be facilitated and provided.

3.2 Traceability in WSSecReq

Traceability in WSSecReq has to do with the elicitation activity. This activity specifies a set of tasks that will produce a set of security requirements to be applied to the piece of software functionality under analysis (SuA).

In this section, we will explain the security artifacts involved in this activity and how they are linked together to obtain full traceability between the WS, whose security is under analysis, and the elicited security requirements. The main artifacts, and the steps in which they take part, are shown in Figure 2.

3.2.1. WS-based business and application patterns.

In [3], a catalog of WS-based business, integration, application, composite and runtime patterns are presented. This catalog of WS-based patterns offers us a complete pattern-based design solution for WS-based system design. In our work, we use these patterns as a reference for identifying the set of potential threats that should be taken into account during the elicitation stage. Basically, these patterns define a set of elements, and their interactions. Thus, threats on these elements and interactions are studied and considered from the very beginning.

First of all, the WS-based *business* patterns underlying the design of the functionality whose security is under analysis are identified and instantiated for the specific system. Notice that if we have already made an analysis at the business level, we may not need to identify and instantiate this type of WS-based

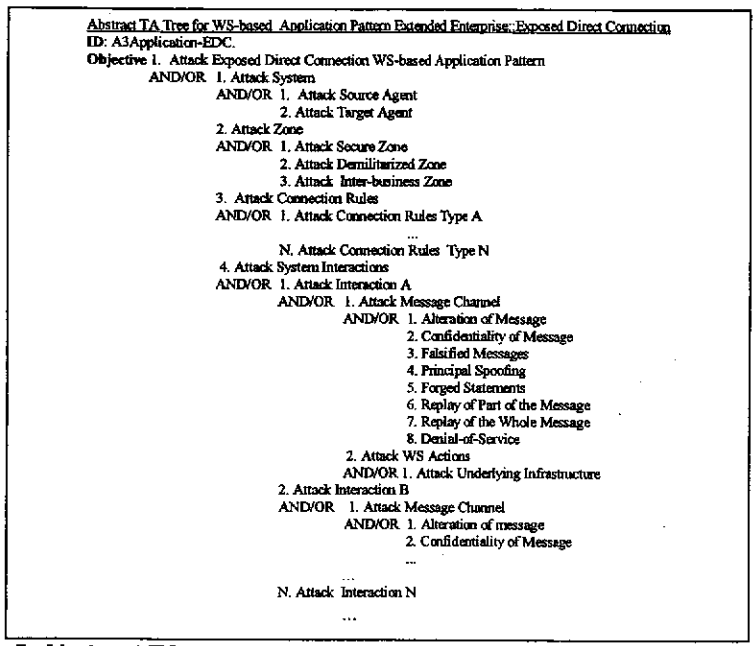

Figure 5. Abstract TA tree associated with the WS-based application pattern named Exposed Direct Connection.

pattern. In this section, we will assume that we have identified a WS-based business pattern.

For every WS-based business pattern, a WS-based *application* pattern can be selected. If we have not specified a WS-based business pattern, we may identify the WS-based application pattern straight forward from the functional software architecture. Then, the WS-based application pattern which the SuA is based on is selected and put into the context of the system. The identification of the WS-based application pattern assumes that there exists a functional architecture where, at least, a set of core WS and interactions have already been defined. For each WS-based business and WS-based application pattern, we have defined an Abstract Threat/Attack (TA) tree that shows how the elements and their interactions - as defined by the WS-based patterns - are threatened.

Our concept of threat and attack is based on the Internet Glossary (RFC 2828) [11].

3.2.2 Abstract and concrete TA trees. We have adapted the security attack trees, as defined in [9, 10], to the context of security WS-based systems. For every WS-based business and application pattern, we have established a relationship with an Abstract TA tree. Thus, once both WS-based patterns have been identified and instantiated for the part of the SuA of the

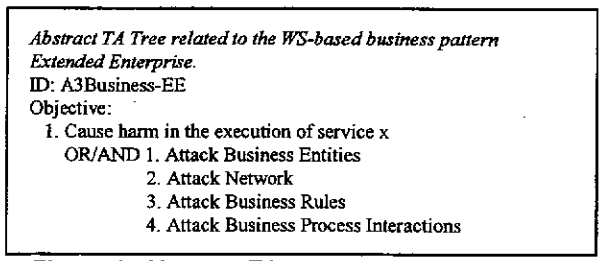

Figure 3. Abstract TA tree associated with the WS-based business pattern named Extended Enterprise.

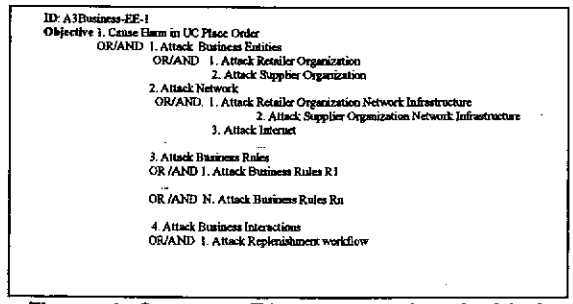

Figure 4. Concrete TA tree associated with the WS-based business pattern named Extended Enterprise.

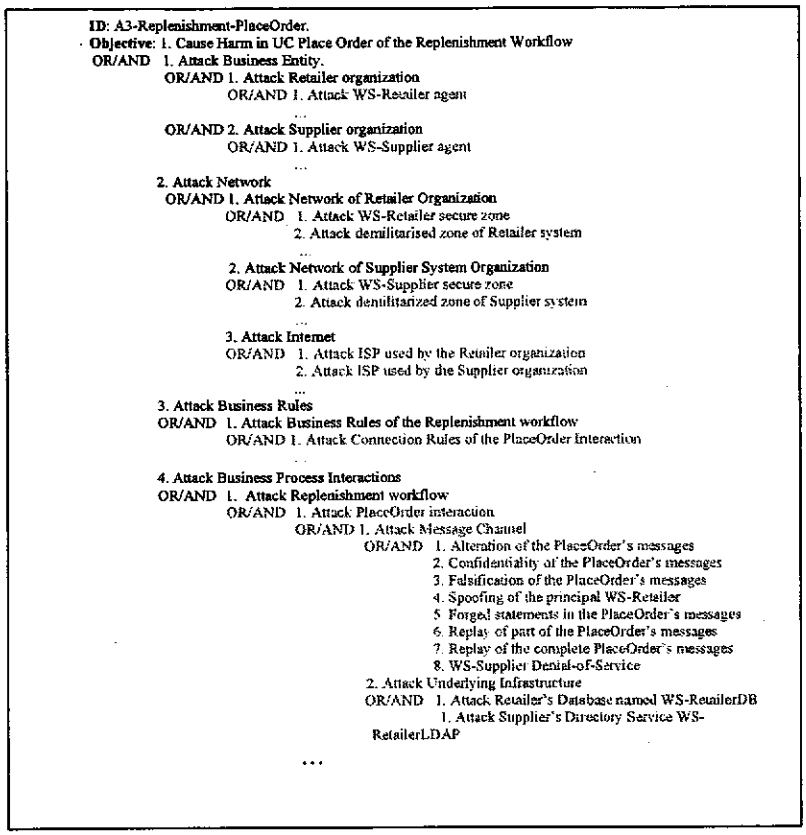

Figure 6. Concrete TA tree resulted after combining the business-level TA tree and the application-level TA tree.

current iteration, a tree-like structured set of threats at both, the business and the application level, is systematically obtained. Firstly, the abstract TA tree associated with the WS-based business pattern will be instantiated.

In consequence, a concrete business-level TA that is specific for the current iteration's SuA is defined. In Figure 3, an example of the abstract TA tree associated with the WS-based business pattern named Extended Enterprise is shown. In Figure 4, an example of instantiation of the mentioned abstract TA is depicted. The same process will be performed for the WS-based application pattern so that a concrete application-level TA tree can be produced. In Figure 5, the abstract TA tree associated with the WS-based application pattern named *Extended Enterprise::Exposed Direct Connection* is shown.

Branch 1.4, known as *Attack System*, will refine branch 1.1 *Attack Business Entities* of the A3Business-EE, 1.2 will refine branch 1.2 of the A3Business-EE, and so forth. Likewise, branch 1.4 *Attack Business Process Interactions* of the A3Business-EE will refine branch 1.4 Attack *System Interactions* The set of threats which appear under branches 1.4.x.1 have been taken from [15]. Once both concrete TA trees have been developed, they will be combined to obtain a single TA tree that groups the set of threats to be considered within the selected fragment of functionality.

Figure 6 shows us an example of the resulting TA tree once both TA trees, business and application level trees have been combined. Combining both trees is optional, being possible to maintain and refine both trees in a separate way (e.g. when the combined TA tree is too big to be handled).

We should highlight the fact that, thanks to this adaptation of the attack trees from [9, 10] and the relationship established with the WS-based business

Table 1. Attack profiles associated with the WS-based application pattern Exposed Direct Connection.

Business Pattern	Application Pattern	Element	Variation	Attack Profile	ID
Extended Enterprise	Exposed Direct Connection	Interaction	Message-based Variation	*WS Message-based Interaction with no Acknowledgement*	AAP-1-1
			Invocation-based Variation	*WS Message-based Interaction with Acknowledgement*	AAP-1-2

and application patterns, we are making it possible not only to consider the aspects of security of the interactions of the WS security agents themselves but also to take into account, although without this being the main objective of this specific work, possible attacks on the provider and consumer organizations, on the network services (e.g.: attacks on the Internet Service Providers of any of the participating business entities) or the infrastructure in use, along with other elements at the organizational level.

3.2.3 Attack Identification. The next step will consist of refining the leaf-nodes of the TA tree, i.e. further specification of the threats by means of concrete attacks. The threats themselves are of no significance if there are not attacks which may bring them to fruition. It is the right time, then, to identify the set of possible attacks which could occur, for each of the threats identified. To do this, use will be made of the concept of Attack Profile described in [9]. The attack patterns set out in this work seem not particularly formal, as compared to the misuse cases in [13]. As both artifacts have the same aim, i.e. to define the sequences of steps which state the achievement of successful attacks on the system, we have opted to employ the second type when defining the attack profiles. Basically, an attack profile contains a set of abstract misuse cases that

Table 2. Abstract misuse case 'Attack on the Semantic Content of the SOAP'.

Name of Abstract Misuse Case: Attack on the Semantic Content of the SOAP [message | interaction] [message | interaction name]		
ID: *AMUC-1-1-1*		
PROBABILITY *[HIGH][MEDIUM][LOW]*		
Summary: the attacker type *[attacker type]* gains access to the *[message | interaction] [name]* exchanged by the *[consumer | provider | discovery]* agent *[agent name]* and the *[consumer | provider | discovery]* agent *[agent name]* and *[modifies | deletes | inserts [part]*]* of the message at the *[transport | SOAP]*-level situated in the *[header | body | attachment]* with the object of *[objective]*.		
Preconditions: 1) The attacker has physical access to the message. 2) The attacker has clear knowledge of the structure and meaning of the message.		
Interactions of the Consumer Agent	**Interactions of the Misuser**	**Interactions of the Provider Agent**
The Consumer Agent *[agent name]* sends the message *[name of message]*		
	The attacker *[type of attacker] [name of attacker]* intercepts it	
	The attacker *[type of attacker] [name of attacker]* identifies the part to modify and *[deletes | replaces | adds]* information	
	The attacker *[type of attacker] [name of attacker]* forwards the message to the Provider Agent *[agent name]*	
		The Provider Agent *[agent name]* receives the message*[name of message]* and processes it erroneously due to the altered semantic content.
Postconditions 1) The system will remain in a state of error with respect to the original intentions of the Consumer Agent *[name of consumer agent]*. 2) In the register of the system in which the Provider Agent *[name of provider agent]* was executed the request received with an altered semantic content will be reflected.		

apply to a reference model defined within the profile. Therefore, interactions in every WS-based application pattern have one attack profile related. Every WS-based application pattern has one or more attack profiles related to it which state the potential attacks that could be targeted at them. For instance, for the WS-based application pattern *Exposed Direct Connection*, the set of attack profiles exposed in Table 2 has been defined. Every attack profile gathers a set of abstract misuse cases that focuses on a particular element defined within the reference model specified for the questioned profile. In Table 1, both attack profiles are interaction-centered, i.e. the attacks they contain are focused on exploiting any vulnerability that may be deduced from the analysis of the messages exchanged within the interaction and from the nature of the interaction itself (eg: synchronous vs. asynchronous, message exchange pattern in use, etc.). Other attack profiles, which are connection rules-centered or zone-centered, have been specified.

In our example, the PlaceOrder interaction follows a request-reply message exchange pattern. An uncontrolled network, i.e. the Internet, is the context that should be assumed for it. These are the variants specified for this profile:

- WS Provider and WS Consumer Organizations. In this study, these are the Supplier and the Retailer Organization, respectively.
- WS Provider Agent and WS Consumer Agent. In our case, these are the WS-Supplier and WS-Retailer agents, respectively.
- The name of the operation to be performed, here known as PlaceOrder operation, which is in line with the message exchange patterns defined in WSDL (Web Services Description Language) [2] classified as a request/reply operation.
- A set of misuse cases has been defined as a result of analyzing the threats enumerated in branch 1.4.x.1.y (interaction attacks) of the A3Application-EDC.

Misuse cases specify the possible attack scenarios that materialize the threats which they are associated with. We have abstract misuse cases and concrete misuse cases. As mentioned above, the former are grouped into attack profiles, while the latter are instances of the former and set out the sequence of steps for a given attack. Two abstract misuse cases described in this profile are listed below: i) **Misuse Case of Attack on Semantic content SOAP (AMUC-1-1-1),** which refines the threat represented by branch 1.4.x.1.1 (Alteration of the Message) of the A3Application-EDC in the WS-based application pattern *Exposed Direct Connection* (see Figure 5); ii) **misuse Case of Attack on the Authenticity of the SOAP Message (AMUC-1-1-2),** which refines branches 1.4.x.1.3, 1.4.x.1.4 and 1.4.x.1.5 of the A3Application-EDC (see Figure 5).

Finally, the possible attackers, primary actors in the stated abstract misuse cases, are (extracted from the attack profile): i) Malicious WS-Provider Agent: the WS-Provider agent may not behave as expected and perform illicit activities such as revealing the identity of buyers for its own benefit (selling this information, creating buyer profiles to personalize offers, etc.); ii) intermediary WS Agent: in the SOAP architecture, which Web-based services systems are founded on, the figure of the intermediary SOAP nodes appears. These nodes can process messages while traveling along their path; iii) External Attacker: this is an attacker who has the ability to perpetrate all the attacks we have pointed out from the Internet. The risk from this type of attacker is extremely high, due to how unpredictable and uncontrollable the Internet is.

In Table 2, an example of an abstract misuse case is presented. As it can be seen, it is highly parameterized; therefore it is not application-specific and can be reused in different systems.

3.2.4 Specification of System Security Behaviour.

Every abstract misuse case holds a relationship with one or more security use cases [5, 12]. Security use cases define a sequence of steps which allow the system to prevent, detect or react to each of the attacks which take place in the form of an instance of the misuse cases they are associated with.

3.2.5 Specification of Security Requirements. Each abstract security use case will have associated with it one or more templates of WS-based security requirements, which should be instantiated in order to obtain the final security requirements. In Figure 7, an example of a WS-based security requirement template is shown. This template is associated with the abstract security use case presented in Table 2.

The steps that should be followed when instantiating the WS-specific security template are explained in detail in [8].

> *"The [consumer agent | provider agent | discovery agent] [name of agent] **must protect** the message [name of message] at the level of [transport <protocol>| SOAP message | both] which is transmitting possible [modifications | deletions | insertions] in [parts of the message] which alter its semantic content due to attacks which are [non-sophisticated | semi-sophisticated | sophisticated] during the execution of [interaction | use case] with this given [metric]".*

Figure 7. WS-specific security requirement template.

4. Conclusions and future research

Security is a crucial aspect if WS-based systems are to be the 'de facto' solution for inter- and intra-integrating heterogeneous systems [16].

In this article, we have presented an overview of the PWSSec process. Then, we have focused our discussion on the reusable artifacts used during the elicitation activity of the WSSecReq. The stated application of these artifacts enables developers to perform a systematic approach that will produce a complete WS-based security requirement specification. In addition, all these artifacts used during elicitation expose associations among them that provide full traceability. This traceability lets us know what security requirements have been derived from which fragment of functionality and vice-versa. This traceability connects the fragment of software functionality whose security is under analysis with the set of security requirements elicited through a set of security artifacts (e.g. threat attack trees, misuse cases, security use cases, etc.).

Some of the research lines we are currently working on are listed below:

- To define and refine TA trees at the business level in order to obtain a complete security vision of the problem. This analysis is producing new business-level TA trees, attack profiles, misuse business cases, security business cases and business security requirements templates.
- To analyze the potential relationships that may exist between branches defined within and between TA trees defined at different abstraction levels (e.g. business, application, etc.).
- To define a formal meta-model for the artifacts in order to make it possible not only to create a repository of reusable artifacts but also to provide tool-based support to developers during the activity of elicitation.
- To incorporate threat and attack trees as a result of taking into account the WS-based Runtime patterns. From the abstraction point of view, WS-based runtime patterns refine WS-based application patterns.

5. Acknowledgments

This research is part of the following projects: MESSENGER (PCC-03-003-1) financed by the "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" (Spain) and CALIPO (TIC2003-07804-C05-03) granted by the "Dirección General de Investigación del Ministerio de Ciencia y Tecnología" (Spain).

6. References

[1] R. Breu, K. Burger, M. Hafner, J. Jürjens, G. Popp, V. Lotz, and G. Wimmel, "Key Issues of a Formally Based Process Model for Security Engineering", Proc. 16th International Conference on Software and Systems Engineering and their Applications (ICSSEA'03), 2003.

[2] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "W3C Web Services Description Language (WSDL) 1.1 - W3C Note 15 March 2001", 2001.

[3] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, *Patterns: Service-Oriented Architecture and Web Services*, 1st ed, 2004.

[4] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, *Patterns: Services Oriented Architectures and Web Services*, 2004.

[5] D. G. Firesmith, "Security Use Cases", *Journal of Object Technology*, vol. 2, pp. 53-64, 2003.

[6] D. G. Firesmith, "Engineering Security Requirements", *Journal of Object Technology*, vol. 2, pp. 53-68, 2003.

[7] D. G. Firesmith, "Specifying Reusable Security Requirements", *Journal of Object Technology*, vol. 3, pp. 61-75, 2004.

[8] C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "PWSSec: Process for Web Services Security", Proc. IEEE International Conference on Web Services 2005, Orlando, Florida, USA, 2005.

[9] A. P. Moore, R. J. Ellison, and R. C. Linger, "Attack Modelling for Information Security and Survivability", Software Engineering Institute 2001.

[10] B. Schneier, "Attack Trees: Modeling Security Threats", *Dr. Dobb's Journal*, 1999.

[11] R. Shirey, "Internet Security Glossary (RFC 2828)", 2000.

[12] G. Sindre, D. G. Firesmith, and A. L. Opdhal, "A Reuse-Based Approach to Determining Security Requirements", Proc. 9th International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'03), Klangenfurt, Velden, Austria, 2003.

[13] G. Sindre and A. L. Opdahl, "Eliciting Security Requirements with Misuse Cases", *Requirements Engineering Journal*, vol. 10, pp. 34-44, 2005.

[14] A. Toval, J. Nicolás, B. Moros, and F. García, "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach", *Requirements Engineering Journal*, vol. 6, pp. 205-219, 2001.

[15] WS-I, "Security Challenges, Threats and Countermeasures", 2005.

[16] J. Zhang, "Trustworthy Web Services: Actions for Now", *IEEE IT Pro*, vol. 7, pp. 32-36, 2005.