MODEL AND IMPLEMENTATION OF SAFETY CASE CORES

Kateryna NETKACHOVA

Abstract: The paper introduces a general concept of Safety Case Core, which is an extension of the Safety Case methodology. The definition of the safety case core is provided, the scope, principles and structure of a core are outlined, and a general set-theoretical model is presented. An approach to tracking and managing vulnerability information, assessing security and reliability characteristics of ready-made software components is discussed. To demonstrate the practical relevance and applicability of the proposed approach, a safety case core for assessing off-the-shelf components is developed. The database schema of the developed safety case core, modelled using the entity-relationship diagram, is presented; the important design and implementation details and techniques are outlined. The integration of the core with ASCE software tool as a plug-in and implementation as a web service for off-the-shelf component assessment are presented.

Keywords: Safety case core, security, vulnerability, OTS component assessment, ASCE plug-in, web service.

Introduction

Safety plays a crucial role in the modern society. Assuring safe operation is one of the vitally important tasks faced by system developers and experts. The concept of Safety Case has been evolving for over 20 years. World famous scientists such as Peter Bishop, Tim Kelly, J Górski and others made a great impact on the concept and its implementation.[1] The concept has evolved, grown and nowadays become a common and generally accepted practice. However, future research is still needed to develop the approach further on and make it even more useful, accurate, efficient, and of course, automated.

In this paper we introduce a concept of Safety Case Core, which can be considered one of the Safety Case methods. We also discuss an approach to assessing security characteristics of OTS components. In order to help security experts in software assessment, vulnerability tracking and management processes, a special Safety Case core for assessing OTS components was developed and implemented as a plug-in for ASCE software tool and as a web service for OTS components assessment.

The paper is structured in the following way: Section 1 provides a definition of safety case core, describes it structure, model and the main concept; Section 2 presents a core for assessing OTS components and gives some implementation details; Sections 3 and 4 describe the implementation of the core as an ASCE plug-in and as a web service for assessing OTS components and tracking vulnerabilities. The paper ends with concluding remarks, future directions and extensions to this work.

1. Safety Case Core concept

Safety Case Core can be defined as a reusable configurable unit that contains a piece of safety assessment logic and can perform various tasks to assist in safety case development and maintaining processes. The concept of safety case cores can be easily understood by viewing the entire system safety assessment process from the standpoint of independent tasks that need to be performed. Some of these tasks are essentially the same or similar in many systems and it would be sensible not to perform those tasks over and over again. Instead, it is better to create separate safety case cores for such tasks and transfer safety assessment logic to the level of such cores. The next steps would be to automate tasks performed within the cores by writing code, and adding a parameterization feature to support various input parameters. The parameterization feature is very important as it makes the core much more flexible, configurable and easier to reuse in different systems.

Parameters serve as input for safety case cores. The values of parameters are usually retrieved from the system requirement profile, because in most cases parameters represent the requirements established for a part of the system assessed using a particular safety case core.

A general set-theoretical model of safety case core can be set up as follows:

$$SCC = \{X, Y, F, D, G, H\}$$

where $X = \{x_1 \dots x_n\}$ is a set of input parameters, which is a subset of all parameters $P = \{p_1, \dots, p_c\}$ supported by the Safety Case core, $X \subseteq P$; c – number of supported parameters; n – number of requirements used as core parameters when assessing a particular system, $n \leq c$; $y_i = F(x_i)$, $i=1...n$; $Y = <H(F(x_1)\dots F(x_n)), G(F(x_1)\dots F(x_n))>$ – the output result from the core; $F = \{f_1 \dots f_m\}$ – a set of functions, defined within the core; $D_f = \{D_{sf_1} \dots D_{sf_n}, D_{df_1} \dots D_{df_p}\}$ - a set of formalized data used by the core, where $Form = \{Form_1, \dots Form_t\}$ –is a set of data formats, $M_{form} = \{M_{form1}, \dots Mf_{ormm}\}$ – a set of data conversion methods, $M_{form_k}: D_{nf_j} \rightarrow D_{f_k}$; H – function that processes and combines the results of F functions execution; G – function that provides a formal representation of the results using one of the safety case notations. The core should pass through all these steps during its execution. The general structure of a core is shown on Figure 1. Figure 2 presents the respective data flow diagram for assessment utilising Safety Case Cores.

2. Safety Case Core for assessing OTS components

There are a number of channels that provide information about software vulnerabilities and are designed to help identify and solve the known security problems. CVE,[2] NVD,[3] Secunia,[4] SecurityFocus,[5] OVAL,[6] and CERT [7] represent just a partial list of such information sources. The need to efficiently store, manage and manipulate this information is at the heart of the assessment process. Thus selecting, organizing and storing vulnerabilities data is one of the first and important steps in the preparation for assessment. The structure of the database should be kept simple and understandable, but at the same time it should be complex enough to include all the necessary characteristics, tables and relations between them. Having analyzed the semantics and structure of the most common vulnerability databases and sources, we propose the following conceptual database schema modelled using the ER diagram (Figure 3):

In order to fill this database with data from various vulnerability sources, a parser converting the information to the appropriate structure is needed. We propose the following basic steps that should be taken to convert the information and load it to the database:

1. Convert the information from the original source format into an object model;
2. Filter the objects created on the previous step to only retrieve vulnerabilities;
3. Map the obtained object properties onto the object properties of our database object model;
4. Convert the resulting object model into the relational database model.

We note that there may be other ways to load the data, for example, by working with relational database models. However, using object-oriented models is, in our opinion, a better and more practical solution, especially when the structure of a third-party database is quite different from our database structure, and we need to retrieve and load large amounts of data.

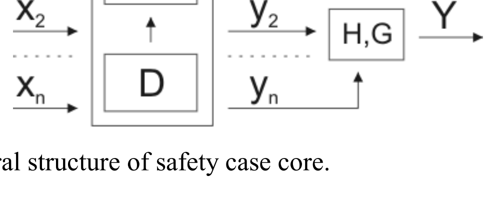


Figure 1: The general structure of safety case core.

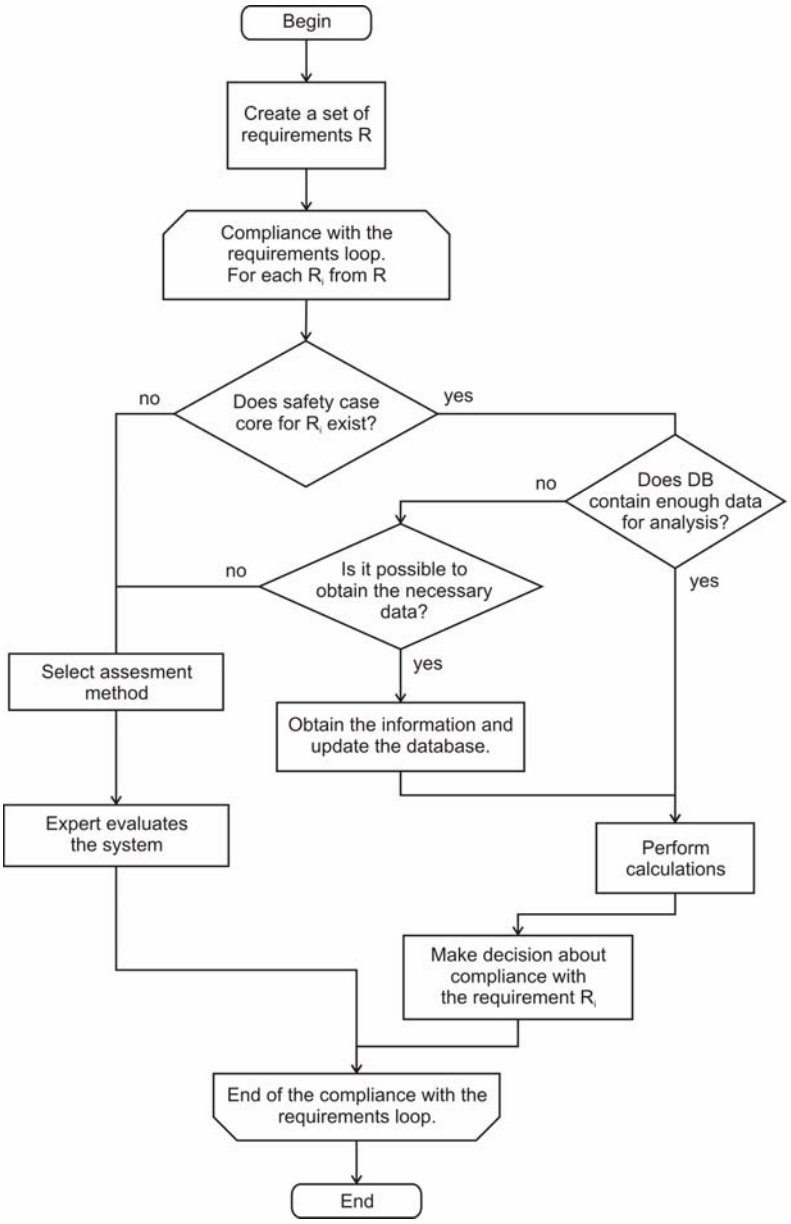

Begin
Create a set of requirements R
Compliance with the requirements loop. For each R_i from R
Does safety case core for R_i exist?
no
yes
Does DB contain enough data for analysis?
no
yes
Is it possible to obtain the necessary data?
no
yes
Select assesment method
Obtain the information and update the database.
Expert evaluates the system
Perform calculations
Make decision about compliance with the requirement R_i
End of the compliance with the requirements loop.
End

Figure 2: Data flow diagram of the assessment process using Safety Case Cores.

After the vulnerabilities data is loaded, it is possible to manipulate it and obtain various software reliability characteristics, such as the number of discovered vulnerabilities, their severity rates, failure frequency, evaluate recovery time etc. The assessment

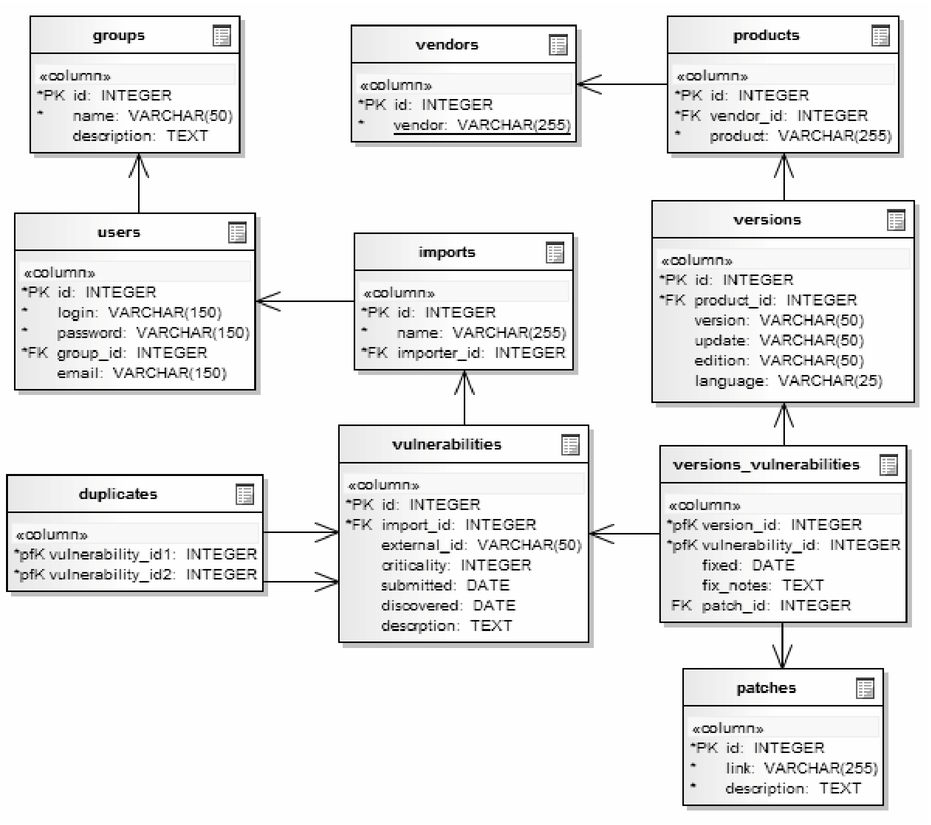


Figure 3: Entity-Relationship diagram.

techniques have subject of analysis.[8] The calculated attribute values are compared with software requirements received as input parameters in order to verify whether the requirements are met or not. After that, the output information from the core is passed onto the next step, folded with other results, if necessary, and used in the system safety case development process.

3. Integration with ASCE tool

The ASCE tool developed by Adelard [9] is the leading commercial system for the development and management of assurance cases and safety cases. ASCE is designed to help creating and managing safety cases with the use of formal notations such as Claims-Arguments-Evidence (CAE) and Goal Structuring Notation (GSN). As part of our work, we implemented a Safety Case Core for OTS components assessment as a plug-in for ASCE tool and demonstrated its applicability by using it to assess several characteristics of OTS components. The main elements of user interface such as the contextual right click menu and the steps of the OTS safety case core wizard are shown Figures 4 and 5. The implementation was performed using Ascad notation as a basis for representation.[10]

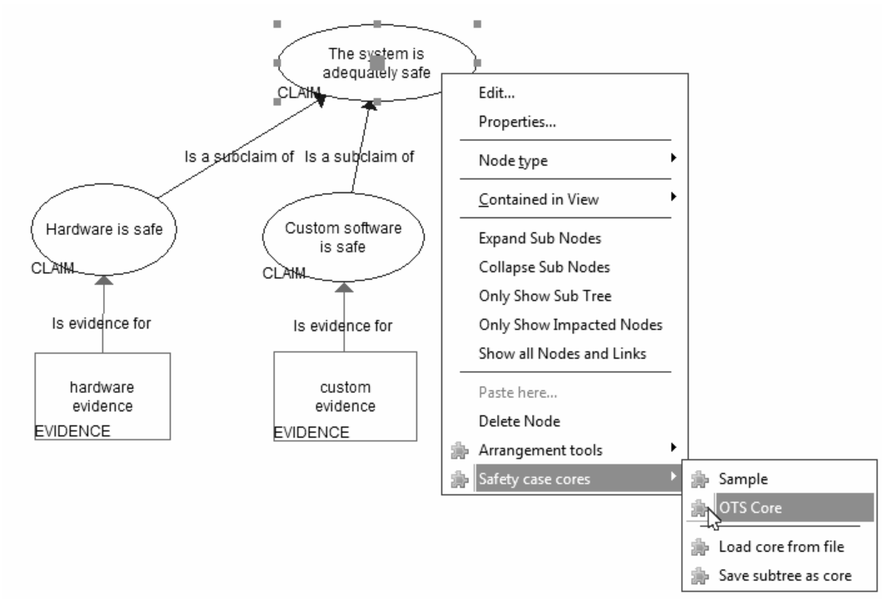
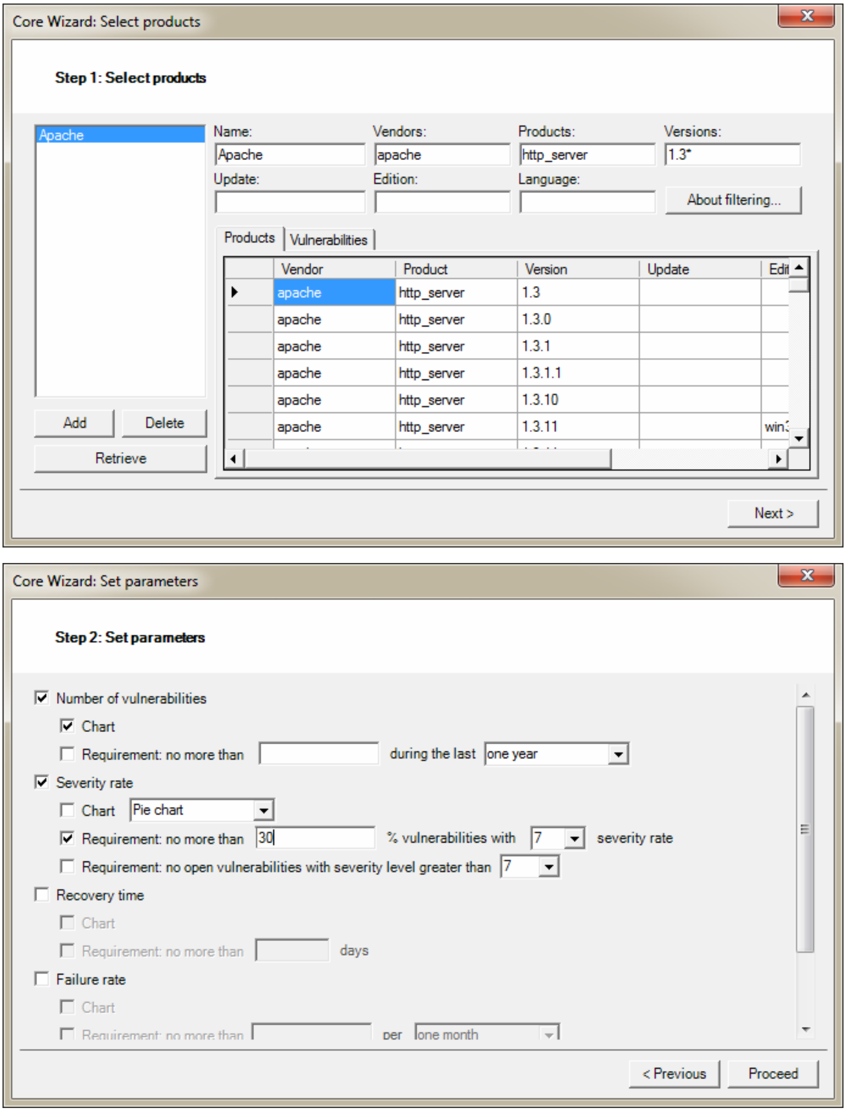


Figure 4: Contextual menu of OTS core integrated into ASCE software tool.

4. Implementation as a web service

Based on the same concept, a web service was developed in order to assist in OTS component assessment, vulnerability tracking and management processes. The web service helps increase the overall reliability of software systems and protect them from malicious attacks. It provides information about discovered vulnerabilities and patches that are available to fix them, thus assisting in remediation of system components and preventing the exploitation of already known vulnerabilities. The service also calculates various software security and reliability characteristics, which can help client applications that support decision making and reconfiguration activities dynamically reconfigure the software based on the obtained results.

The web service is developed on Microsoft .Net platform using IIS and MS Sql server. The interface is described in a machine-processable WSDL format. Client applications interact with the web service using SOAP messages, transmitted over HTTP protocol with POST method. In our scheme Configuration Control Server acts as a client. It contains information about all software components used within the system as well as the way those components are configured.



Core Wizard: Select products

Step 1: Select products

Apache

Name: Apache
Vendors: apache
Products: http_server
Versions: 1.3*
Update:
Edition:
Language:

About filtering...

Add | Delete | Retrieve

Products | Vulnerabilities

	Vendor	Product	Version	Update	Edit
▶	apache	http_server	1.3		
	apache	http_server	1.3.0		
	apache	http_server	1.3.1		
	apache	http_server	1.3.1.1		
	apache	http_server	1.3.10		
	apache	http_server	1.3.11		win

Next >

Core Wizard: Set parameters

Step 2: Set parameters

☑ Number of vulnerabilities
☑ Chart
☐ Requirement: no more than ___ during the last one year
☑ Severity rate
☐ Chart Pie chart
☑ Requirement: no more than 30 % vulnerabilities with 7 severity rate
☐ Requirement: no open vulnerabilities with severity level greater than 7
☐ Recovery time
☐ Chart
☐ Requirement: no more than ___ days
☐ Failure rate
☐ Chart
☐ Requirement: no more than ___ per one month

< Previous | Proceed

Figure 5: The interface of the OTS core wizard steps.

The whole process of using the web service can be summarized in the following steps:

1. The Configuration Control Server (CCS) checks the security state of its components on a periodic basis by sending POST requests to invoke a web service operation. The body of such request is composed of an XML mes-

sage that contains the list of software products, their vendors, versions and languages.

2. Web service processes the request and connects to the database to retrieve the necessary information for each particular software product. Based on the retrieved information, open security threats are identified. Then various security and reliability characteristics such as the number of discovered vulnerabilities, their severity rates, failure frequency, recovery time, faultness probability etc. are calculated and prediction based on extrapolation of calculated results is carried out.[11] If available, the links to security patches are retrieved. The obtained and calculated information is used for generating a SOAP response.

3. Based on the response from the web service, CCS makes a decision to dynamically reconfigure the system, apply patches to vulnerable software products or reboot the hosts with alternative non-vulnerable components.

The Configuration Control Server can also contain logic for comparing characteristics of different software products and choosing an optimal system configuration even when there are no open security breaches or when every product of the same type has some sort of vulnerabilities in it. The web service architecture is presented on Figure 6.

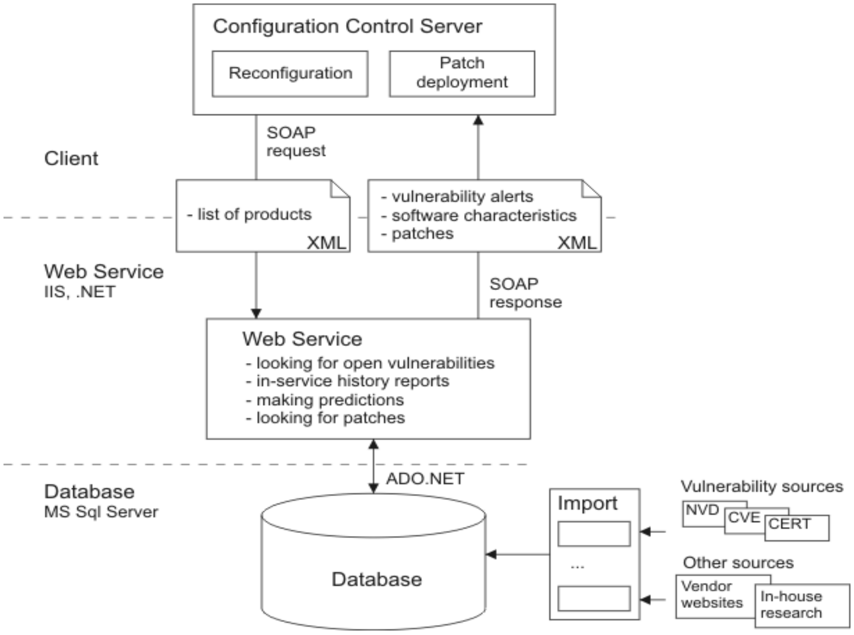


Figure 6. Web service architecture.

To summarize, the main advantages of the developed web service are as follows:

- The web service helps monitoring security channels for vulnerability announcements and alerts a client to the danger when a new vulnerability is discovered.
- Calculates various software security and reliability characteristics based on the information about previous failures (retrospective analysis). This feature is of great help in analyzing products in-service history. It also allows predicting the potential issues that may occur in the future.
- Provides information about security patches. As a rule patches are provided by software vendors or third-party developers and located on their official websites. Monitoring and applying those patches are tedious and time-consuming tasks, which often require manual effort. That's why it is useful to have a single place that contains security patches for all applications. The links to security patches are returned within the web service response, which allows the client application to configure automatic updates and deploy patches without any, or with very little manual manipulations.

The developed web service is recommended to be used in conjunction with the intrusion-tolerant architecture [12] or similar architectures for constructing complex software based systems.

Conclusion

In this paper we have outlined the concept of Safety Case Cores and presented a core for assessing OTS components. The core was developed using an object-oriented approach and integrated both as a plug-in for ASCE tool and as a web service for assessing OTS components. The practical applicability of the core was confirmed by taking a set of OTS components, evaluating their reliability characteristics and comparing them with the required values. For future work, we plan to extend the functionality of our OTS core and deploy our web service as a cloud-based solution.

Notes:

[1] Peter Bishop and Robin Bloomfield, "Methodology for Safety Case Development," Proceedings of the Sixth Safety-critical Systems Symposium, Birmingham, UK, February 1998; Timothy Patrick Kelly, "Arguing Safety – A Systematic Approach to Managing Safety Cases," PhD diss. (University of York, 1998); Timothy Kelly, "Managing Complex Safety Cases," in *Proc. 11th Safety Critical System Symposium* (Berlin: Springer, 2003): 99–115; Janusz Gorski, "Trust Case – a case for trustworthiness of IT infrastructures," in Janusz S.

Kowalik, Janusz Gorski, and Anatoly Sachenko, eds., *Cyberspace Security and Defense: Research Issues* (Berlin: Springer, 2005), 125-41.

[2] *Common Vulnerabilities and Exposures*, Mitre Corp, www.cve.mitre.org.

[3] *National Vulnerability Database*, http://nvd.nist.gov.

[4] *Vulnerability and Virus Information*, http://secunia.com.

[5] *Community of Security Professionals*, www.securityfocus.com.

[6] *Open Vulnerability and Assessment Language*, http://oval.mitre.org.

[7] *Computer Emergency Response Team*, www.cert.org.

[8] K.I. Lobachova and Vyacheslav S. Kharchenko, "Assessing Software Vulnerabilities and Recovery Time: Elements of Technique and Results", *Radioelectronic and Computer Systems* 8 (2007): 61-65; Sung-Whan Woo, Omar H. Alhazmi, and Yashwant K. Malaiya, "Assessing Vulnerabilities in Apache and IIS HTTP Servers," *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing* (DASC'06), 103-10; *Report on the Application of Safety Techniques to Security*, Part 2, *Quantitative Modeling Produced* (London: Adelard LLP, 2010).

[9] See the company website at www.adelard.com.

[10] ASCAD: *Adelard Safety Case Development Manual* (London: Adelard, 2010).

[11] For details on the evaluation techniques see Lobachova and Kharchenko; Woo, Alhazmi, and Malaiya; and part 2 of the Adelard *Report on the Application of Safety Techniques to Security.*

[12] Anatoliy Gorbenko, Vyacheslav Kharchenko, Olga Tarasyuk, and A. Furmanov, "F(I)MEA-Technique of Web-services analysis and Dependability Ensuring," in Michael Butler, Cliff B. Jones, Alexander Romanovsky, and Elena Troubitsyna, eds., *Rigorous Development of Complex Fault-Tolerant Systems* (Berlin: Springer, 2006), 153-67.

KATERYNA NETKACHOVA is a post-graduate student. She received a specialist degree in Computer Systems and Network from Tavrida National V.I.Vernadsky University. Research interests: safety analysis techniques, software security and reliability assessment, formalization and automation of the assessment process. She is currently finalising the work on her PhD thesis.