High-Level Design Process For NATO High Assurance ABAC Guard

Konrad Wrona
NCI Agency
THE NETHERLANDS

konrad.wrona@ncia.nato.int

Sander Oudkerk
Agent Sierra Consultancy Services
THE NETHERLANDS

sander.oudkerk@agentsierra.nl

Christian Hein, Nadja Menz, Tom Ritter
Fraunhofer FOKUS
GERMANY

firstname.lastname@fokus.fraunhofer.de

ABSTRACT

In the paper we describe the development process for the High Assurance ABAC Guard (HAAG), which is one of the important security enablers the NATO future information sharing architectures, including Information Exchange Gateway Scenario D and Future Mission Networks. The HAAG implements Attribute-based Access Control (ABAC) for information requests, and enforces Content-based Protection and Release policies. Our system design process incorporates a structured way of collecting requirements and takes into account a security risk assessment of the system. The process is based on industry standards and best practices. It is accompanied by a definition of a Common Criteria Protection Profile, which captures security requirements for the HAAG. All phases of the system design process are performed using an integrated modelling environment based on Eclipse and open-source tools. The environment allows us to build and maintain a relatively complex model and, to a large extent, automatically generate the required design documentation.

1.0 INTRODUCTION

In the paper we describe the development process for the High Assurance ABAC Guard (HAAG), which is one of the important security enablers the NATO future information sharing architectures, including Information Exchange Gateway Scenario D (IEG-D) [23] and Future Mission Networks. The HAAG implements Attribute-based Access Control (ABAC) for information requests, and enforces Content-based Protection and Release (CPR) policies [1]. Instead of only a security marking and user's (or system) clearance, the CPR policies consider an extended set of properties of content, user, system and environment when defining access control rules for an information object. The HAAG is an interim solution on a path towards a fully integrated and distributed High Assurance Separation Service (HASS) [28]. The described development process has been defined and executed as part of a HAAG High-Level design project within 2012 Scientific Programme of Work sponsored by NATO Allied Command Transformation.

Initially, we attempted to develop a high-level architecture for the HAAG in accordance with the NAFv3 methodology. However, we have discovered that the extremely broad and generic scope of NAFv3 is not suitable for the definition of high-level architectures for security-critical systems such as the HAAG. The

HAAG is special-purpose security system, which can be reused as a component of several NATO larger-scale architectures, including NATO Network-Enabled Capability and FMN. Whereas, these larger architectures can benefit from NAFv3 views, the scope of most of these views is not-well suited to HAAG and introduces too high burden for the scope of the system design project. In addition, the lack of well-defined methodologies to apply NAFv3 further limits its usability for software design and security engineering. In order to counter these issues, we propose a light-weight system design process for the HAAG high-level architecture.

2.0 SYSTEM DESIGN PROCESS FOR THE HAAG

The HAAG is a software intensive system, which should be developed according to a dedicated software development process. Software intensive means that a system is driven by software in design, construction and deployment [14]. Typically, software development processes are separated in different phases and can be defined as the total set of software engineering activities needed to transform a user's requirements into software [10]. Several models of software development process have been defined. They describe all the activities that are necessary in order to ensure a high quality software product. Prominent examples are Waterfall model, Spiral model or V-Model [22], [2] , [7], and [8]. Further process models incorporate agile methods [15], which include iterative prototyping, or model-driven principles [18] that integrate different abstraction levels of software system (so-called domain models). The IEEE and other standardization bodies also define agreed approaches to different phases of a software development process [5], [11] or [12]. All of these approaches fit in principle to HAAG but are too complex to be fully applied within this project. For this reason a customized development process has been defined, which comprises a requirement specification and risk analysis phase, an architecture specification phase and a design phase, which are basically reflected in all the listed approaches.

The requirements specification phase mainly focuses on collecting requirements from all stakeholders and analysing them. Requirements are often collected iteratively in order to expose inconsistencies between different stakeholders' needs. After the consistent state is reached, the requirements are transformed into a detailed design description within the design phase. Similar to the requirement phase the design can also be separated into different activities and is performed iteratively. In particular the architectural design (high-level design) and the detailed design are well known design activities [9] and [13]. The V-model [8] doesn't strictly separate between requirements and design phase. In fact, both phases are part of a general planning phase, which includes requirement specification and analysis, system specification, and system design. Nevertheless, it is necessary to track which element of the system specification or system design is allocated to a corresponding requirement.

2.1 Development Process

The initial HAAG architecture documentation described the architecture of the HAAG according to NAF, however an architecture description only partially address the required aspects of a high-assurance software development process. In fact, the specific aspects of the software development process are not reflected in NAF at all. Taking into account that the NCI Agency project has focused on the definition of a high-level design as well as on a security risk assessment and mitigation (e.g. by definition of a protection profile), a customized HAAG development process was needed that includes phases for requirement specification, risk analysis, architecture specification, and system design. Definition of the HAAG-specific development process was based on the *System and software engineering – software life cycle processes* specified by ISO/IEC/IEEE [13]. The specification establishes a common framework for software life cycle processes. This framework can be used by organizations, and in particular by software industry, to define their own customized development process with a well-defined terminology. The ISO 12207 [13] specification defines processes, activities and tasks that can be used for different development objectives such as acquisition of

software product or service, supply, development, operation, maintenance and disposal of software products.

There are two main process groups defined in the standard. The *System Context Processes* are targeting standalone software products, services or software systems. For implementation of software product that is a component of larger systems, the *Software-Specific Processes (SSP)* can be used. The HAAG is a software intensive system. Therefore, the relevant major processes of the second group (SSP) are taken into account, namely the *Software Implementation Process* and the *Software Support Processes*. According to the scope of HAAG project, the appropriate processes from this standard are *Software Requirements Analysis Process* and *Software Architectural Design Process*, which are both lower-level processes of the Software Implementation Process. In the context of this IEEE standard the purpose of Software Requirements Analysis Process is to establish the requirements for the software components of the system. The Software Architectural Design Process provides a design for the software that implements and can be verified against the requirements. Nevertheless, the overall customized development process should also reflect activities which go beyond current NCI Agency design project, such as *Software Verification*, *Deployment* or *Maintenance*. The analysis of the relevance of corresponding processes in these areas has been left for the future work.

In addition to classical software engineering, the HAAG project is also targeting the Common Criteria compliance [3]. For this purpose, a HAAG Protection Profile is being developed. This document contains a system overview, a set of threats and security policies as well as security functional and security assurance requirements. This Protection Profile can therefore be used as the source of an initial risk and requirements analysis around which the development process has been established. Figure 1 depicts the outline of the development process for the HAAG. The proposed process incorporates all process activities required by a general development process.

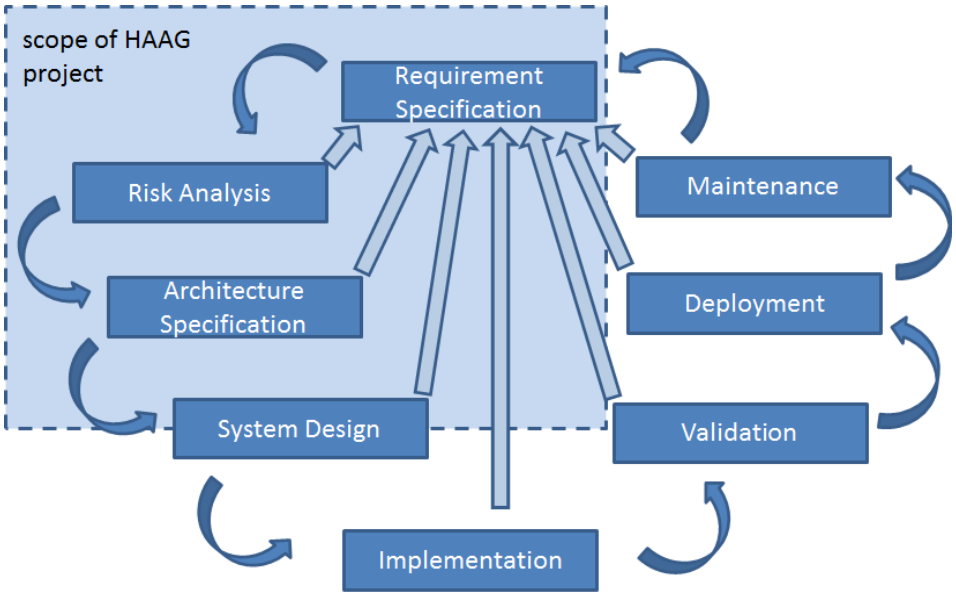

Figure 1 – Proposed customized HAAG development process

The process consists of eight phases, as depicted in Figure 1. The relevant phases for the HAAG high-level design project are:

- Requirement Specification
- Risk Analysis (CORAS, Protection Profile)

- Architecture Specification
- System Design.

The fundamental phase of this process is the Requirement Specification phase. Each phase of the development process is connected to this phase. The main activities of this phase are partially based on the existing process model called V-model, which includes activities focused on determining and analysing requirements. The requirement phase is followed by the Risk Analysis phase, which is based on the CORAS method [16]. The CORAS method is an eight-step method for risk analysis and is accompanied by relevant support tools and a modelling language. The aim of this phase is to identify risks and how to treat them. After risk analysis has been performed, the HAAG system is specified within the Architecture Specification phase. This includes analysis of different possible system architectures, a description of the chosen architecture and description of requirements posed on the architecture elements. The System Design is the last phase of the HAAG high level design project activities. It comprises e.g. the specification of interfaces and other details, which are realizing the system. However, the System Design phase is followed by further phases, which are not part of the HAAG high level design project. The System Design phase is only partially addressed, due to the fact that it also includes detailed (low-level) design of the HAAG.

2.2 PROCESS WORKFLOW

In the context of the HAAG HLD project, a specific workflow has been established that embeds all relevant activities from the customized development process. The aim of the workflow is to detail the mentioned general phases by defining the specific process steps and expected outcomes of these activities. Figure 2 shows the detailed workflow of the development process for the HAAG.

The workflow is divided into 3 layers that relate to the corresponding phases from the general development process. Thereby, the first layer includes the two phases Requirement Specification and Risk Analysis. The reason for this approach is the integration with the Common Criteria (CC) [3] methodology. The second layer represents the activities of Architecture Specification followed by the definition of the System Design, which is located in the third layer.

According to the ISO 12207 framework for life cycle processes [13], the starting point of this workflow is the Requirement Specification process step. It includes identification of requirements of different stakeholders. The requirements can be grouped as needed, e.g. in functional and non-functional requirements. After that the requirements are analysed for correctness and testability, and in particular the relevant security requirements are identified. These security requirements are the input to the next step of the workflow which is the Security and Risk Analysis according to the Common Criteria methodology. Following the Common Criteria approach, a risk analysis based on the initial system requirements has to be conducted. As this step is deemed axiomatic within the CC methodology, we propose the well-defined CORAS method for this step, as its resulting risk models can be directly translated into threat descriptions required by a PP. As a result of this step, the requirement specification is refined by specific security functional requirements (SFR) that are suitable for the identified threat model and threat levels. Furthermore, the security analysis phase also contributes Common Criteria non-functional requirements (CCNRs) in form of organizational security policies, assumptions and security assurance requirements (SAR) to the requirement specification. The outcome of the security analysis activity is the protection profile (PP) for the HAAG. Finally, the outcome of the Requirements Specification step is a Requirement Specification Document. This document might have several versions. The first version constitutes the baseline for driving further development. The document and its revision are communicated to all affected stakeholders according to [13]. For sharing and reusing those requirements, the Requirement Specification was made available in Requirements Interchange Format (ReqIF) [19].

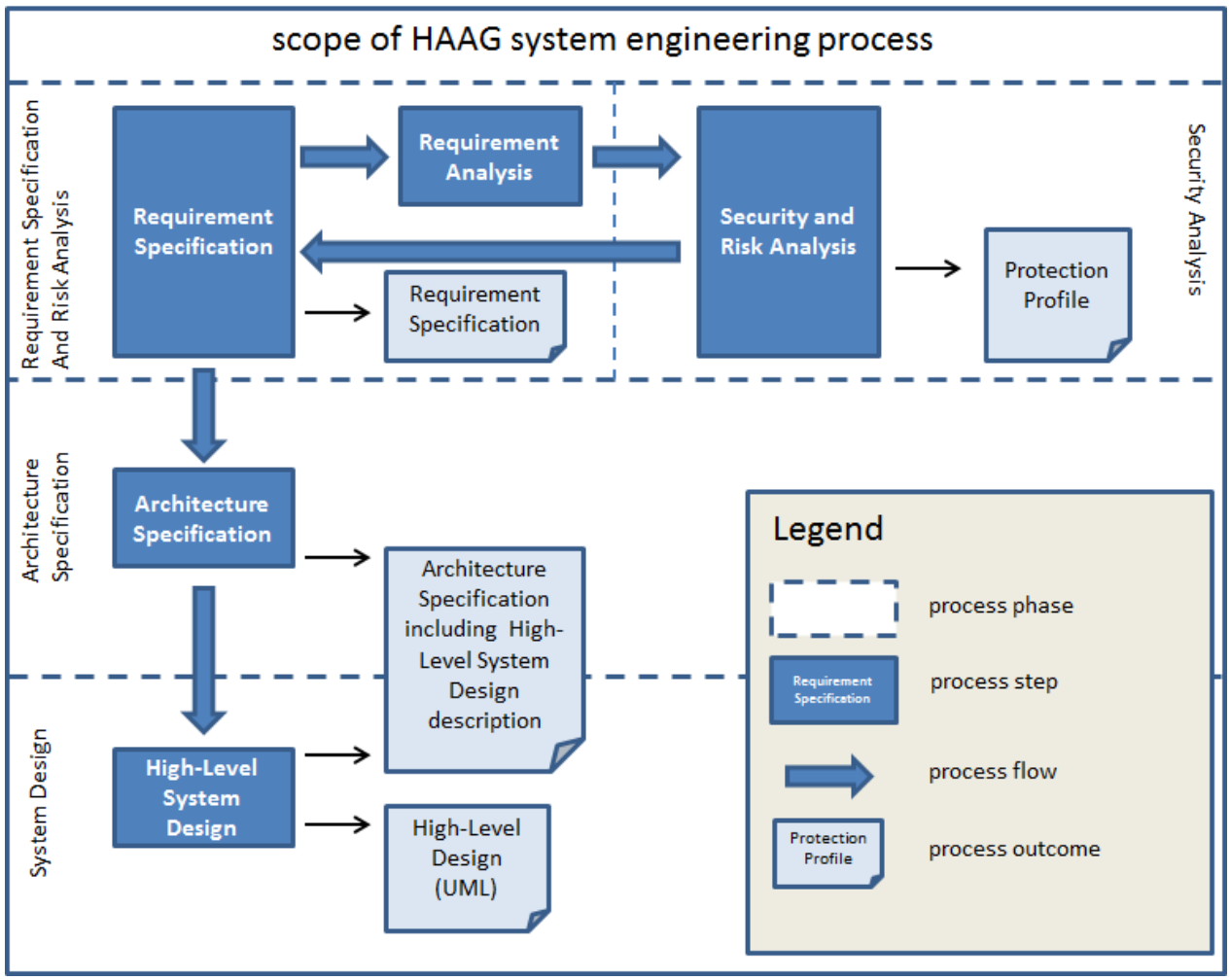

Figure 2 – Detailed workflow of the proposed development process for HAAG

The second layer contains one process step: the Architecture Specification of the HAAG. Based on the life cycle framework from IEEE, the goal of this step is to transform the requirement specification into the HAAG architecture. It is important that all architecture elements are linked to the requirements, in order to ensure consistency and traceability between requirements, architecture and design. The outcome of this step is the Architecture Specification. This Architecture Specification is enriched by a High-Level System Design, which is partially the outcome of the High-Level System Design process step, contained in the third layer of the workflow – System Design. The creation of a UML model representing the High-Level System Design is also part of High-Level System Design process step of the third layer.

Within the scope of this project the System Design phase is targeting a description of the High-Level design of the HAAG system only. Therefore, it doesn't cover other process steps like low-level design or implementation.

3 ENGINEERING ENVIRONMENT

In the previous section the HAAG engineering workflow was described. In this section, a tool environment is described, which have been used in order to execute the relevant workflow steps. It is an Eclipse [4] based workbench. The Eclipse framework has several benefits for our project. First of all, it offers a set of tools

supporting all HAAG development workflow steps. Secondly, most of the relevant Eclipse tools (plugins) are open source or useable at no charge. Finally, the engineering process itself can also be managed and supporting activities like quality assurance of the produced artefacts or requirement coverage can be executed by the platform. The baseline for the development environment is the Eclipse Modelling Bundle. The used version is Juno. It contains all essential plugins for model-driven engineering such as Eclipse Modelling Framework (EMF) [6] or Object Constraint Language (OCL) [20]. All other tools are plugins for the Eclipse framework and provide new functionality. The following list summarizes the main components which are integrated in the HAAG Eclipse-based development environment:

- Papyrus is a UML modelling tool [21]. The tool is used to specify the High-Level Design with UML. It also contributes to the requirement specification by refining specific requirements with UML use case semantics.
- ProR is a requirements engineering tool. The tool supports definition and management of requirement specifications according to the ReqIF standard. It is used to specify all relevant requirements (functional, non-functional, security) for the HAAG.
- The CORAS tool is an editor for risk analysis according to the CORAS method. The CORAS tool has been integrated into the HAAG development environment and is used in the security analysis step within the HAAG engineering workflow.

In order to ensure a certain quality level of the produced models, a set of modelling rules and guidelines has been specified that is to be validated during the development process (e.g. high level design). Within the HAAG project the evaluation was performed using Metrino. The Metrino tool was incorporated into the HAAG development environment and the tool was used to analyse the high-level design model according to specific modelling rules (e.g. naming conventions, complexity rules).

4 SYSTEM REQUIREMENTS SPECIFICATION

The System Requirements Specification (SRS) document is one of the products created during the execution of the HAAG development process. The requirements contained herein are generated from a requirements database, which is used for managing the requirements, while this document is used only to display the requirement in the document form. Any requirements changes shall be applied first to the requirements database, which shall remain the primary data store for the requirements.

The SRS document details the requirements of the HAAG in different categories. The HAAG is providing several security functions. Therefore, security requirements play a particular role. These requirements are especially defined in the Protection Profile (PP) [25].

The requirements towards the HAAG listed in this document are on an abstract and general level. The intention of the gathered requirements is to guide the definition of the High-Level Architecture of HAAG as described in [26].

4.1 Organization of Requirements

The requirements are organized in a way that they do have dependencies to each other. The general organization of the requirements database and the specifics of the dependencies are shown in Figure 331.

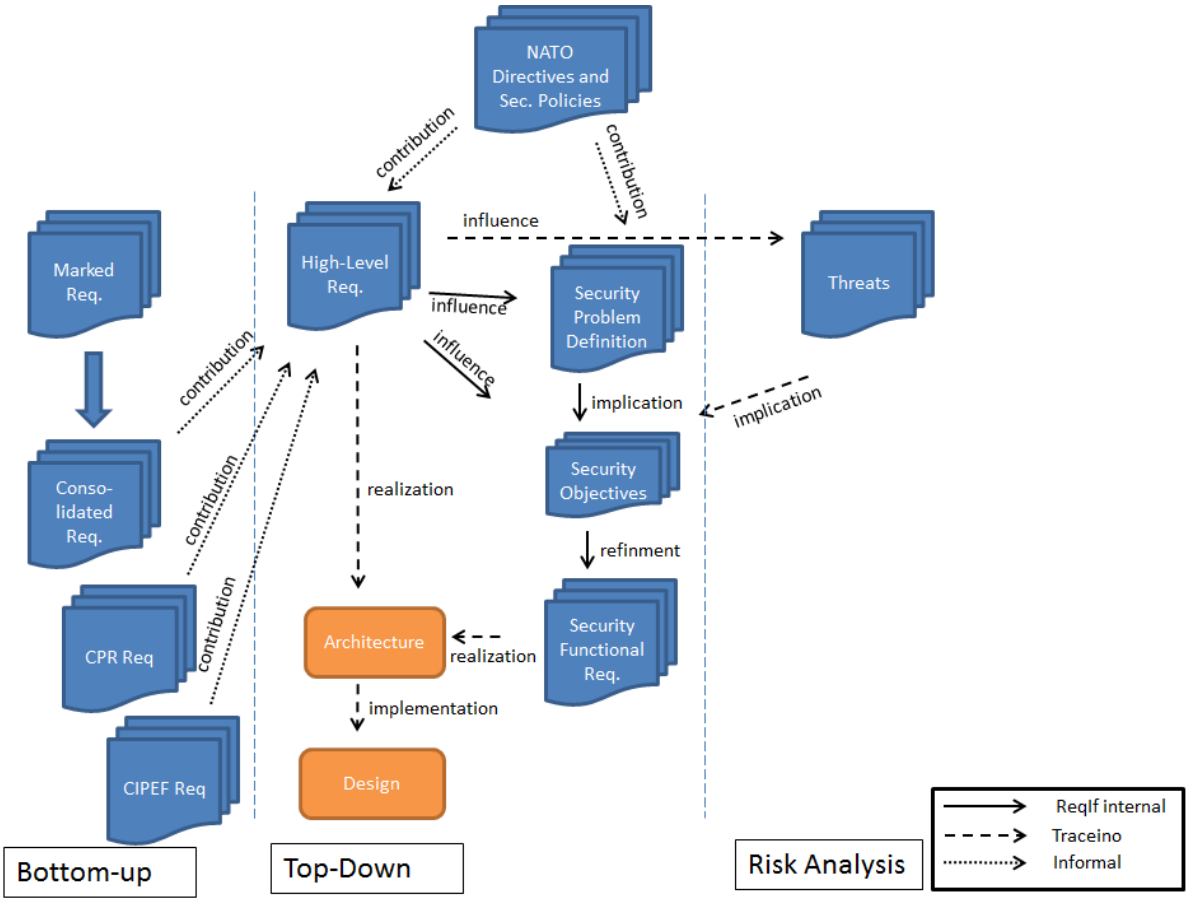

Figure 3 – Requirements Dependencies

The requirements are divided into three groups, depending on their origin (see Figure 331). The "Bottom-up" group shows how existing initial project documentation has been used in order to converge to a comprehensive set of high-level requirements. The initial architectural description of the HAAG has been analysed and sections have been marked ("Marked Req."), and used as input to the requirements statement. This resulted in a particular requirements specification, which is named "Consolidated Req.". In addition, the existing requirements for Content-Protection and Release (CPR) [1] and Content-Inspection and Policy Enforcement Framework (CIPEF) [29] have been gathered as well in separate requirements specifications ("CPR Req" and "CIPEF Req"). No elements of the "Bottom-up" group are part of this SRS specification document, as they do not explicitly form the high-level requirements of the HAAG but only contribute to requirements definition in an indirect way and are therefore displayed here only for presenting the complete picture of the requirements analysis phase.

The middle part of Figure 331, called "Top-Down", shows the main focus of the requirements gathering activities where existing NATO Directives and Security Policies contributed to. The requirements are grouped in categories (high-level requirements, security problem definition, security objectives, and security functional requirements). The "High-Level Req." group represents the requirements on the HAAG stated from a general perspective and does not contain detailed technical requirements on individual components of the HAAG. Those detailed technical requirements shall be formulated in subsequent refinement processes which are outside the scope of the project. The high-level requirements are traced with a "realisation" trace to the HAAG architecture elements, which have been created during the System Design phase (see Figure 2). An additional trace, marked as "implementation", links the system architecture elements with the system

design elements. The trace information helps to check that high-level requirements are appropriately covered by the system design.

Furthermore, the high-level requirements constitute input ("influence") towards formulation of the Security Problem Definition (SPD), and determine which relevant Policies and Assumptions are applicable to the SPD. The term "influence" means that the description of the SPD and Security Objective is effected by the high-level requirements. The SPD also contains the definition of Threats, which were identified within the Risk Analysis part, which is described in section 3 below. The SPD implies the definition of security objectives, which is marked with an "implication" trace link. Security objectives are another category of requirements. The security objectives are described in detail in the HAAG Protection Profile [25] and are contained in the SRS document for completeness and for facilitating the complete traceability of requirements and system design.

In addition, there are influences from high-level requirements to the selection of security objectives as part of the SRS. This means, that the definition of security objectives is an indirect consequence of the high-level requirement specification. This link is captured with the "influence" traces.

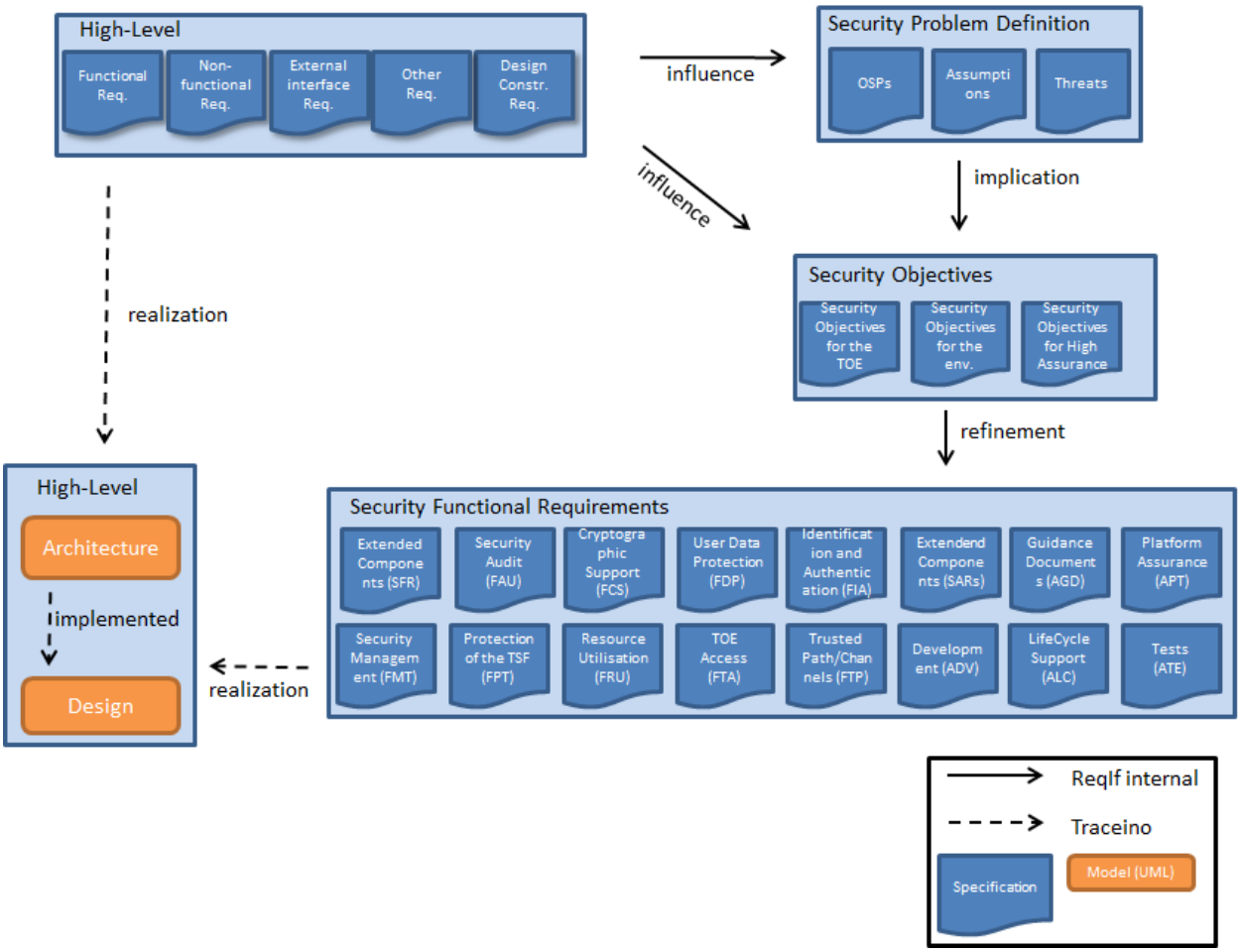

Figure 4 – Detailed Requirements Dependencies

The security objectives are further refined into Security Functional Requirements (SFR). Those SFR describe

concrete functional requirements for the system from the security point of view. Each SFR is traced to the system architecture with a "realization" trace in order to show which HLD element is actually fulfilling the requirement.

The right hand side of Figure 331 shows a particular part of the requirements analysis – the Risk Analysis. Based on the high-level requirements a number of security threats have been identified and assessed. These threats are traced to the security objectives with an "influenced" trace, which helps to check the appropriate mitigation of security threats through the security objectives and the subsequent security functional requirements. The Security Assurance Requirements (SAR) are not traced to threats via objectives for the TOE or the environment. The set of SARs as defined in the PP was chosen based on security objectives for high assurance (which is as such not defined in the CC) [25].

Figure 4 depicts the details of the high-level requirements, security problem definition, security objectives and security functional requirements captured in the SRS and in the PP documents. The high-level requirements are grouped in functional requirements, non-functional requirements, external interface requirements, design constraints and other requirement.

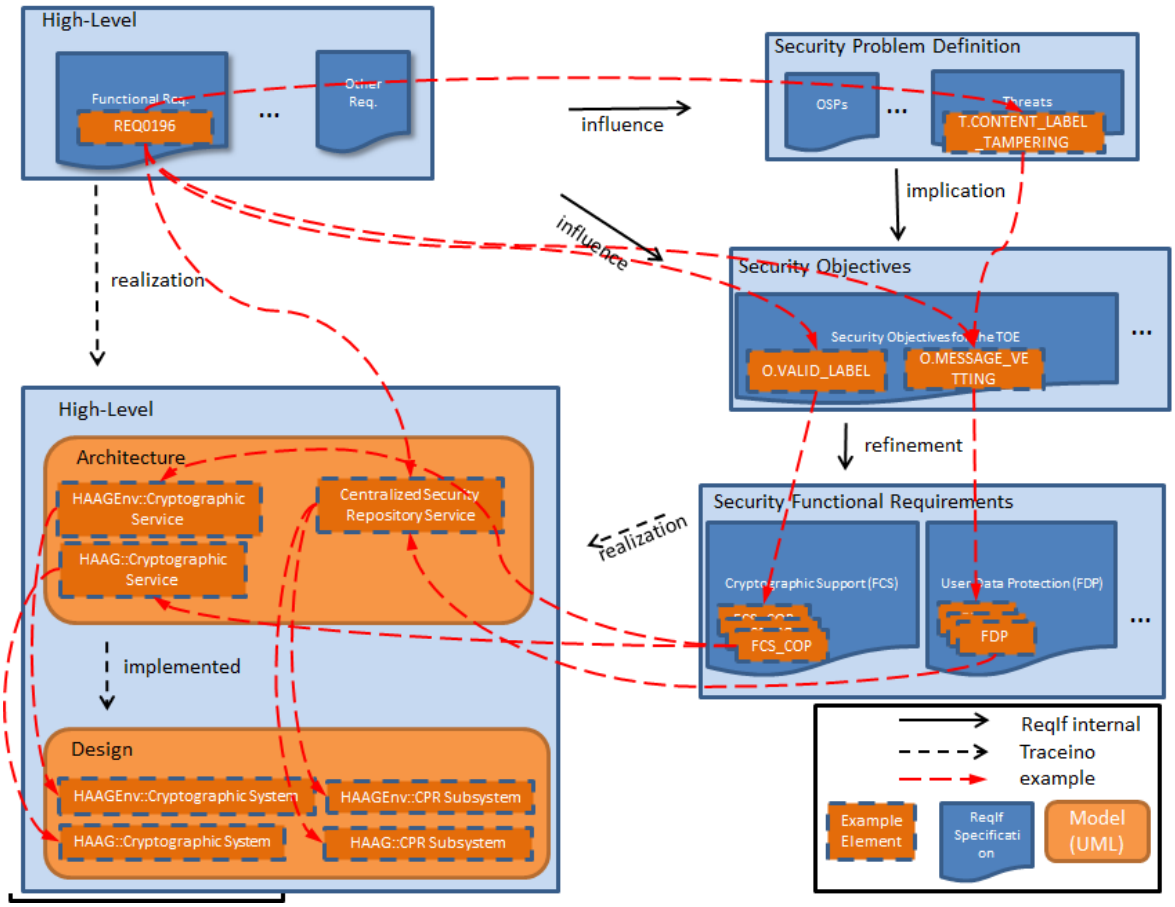

Figure 5 – Example Requirements Traces

In order to ease the understanding the relationship between the various requirements specifications, Figure 4 shows the refinement and the traces of requirements with the help of an example.

The example requirement is a high-level functional requirement. It has an identifier with value REQ0196 and

it states that “The HAAG shall support the NATO XML-Labelling specifications as defined in [30] and [31].” This requirement is influencing the definition of a security threat, which is called “T.CONTENT_LABEL_TAMPERING”. In turn, this security threat implied definition of the security objective “O.MESSAGE_VETTING”.

The requirement REQ0196 is also a direct input to (i.e. it influences) the definition of the security objectives. In this example the security objective O.VALID_LABEL and O.MESSAGE_VETTING are defined. O.VALID_LABEL defines that “The Target of Evaluation shall validate the origin, integrity and binding of security label to a data object before it is used”. Furthermore, O.MESSAGE_VETTING describes that “The Target of Evaluation shall control the flow of information from the external security domain to the internal security domain by only relaying messages that are allowed as part of the TOE security policy.” Within the CC approach [3] the security objectives are refined to produce a set of security functional requirement. In the case of REQ0196 the outcome of this security objective refinement is a set requirements for Cryptographic support (FCS) and a set of requirements for User Data Protection (FDP). Figure 4 also depicts the high-level design phase of the HAAG including architecture specification and definition of the high-level design. The requirements (high-level, security objectives, security functional requirements) are the basis for the HAAG architecture. In the example depicted in Figure 5, the Centralized Security Repository Service and the Cryptographic Service elements of the HAAG architecture realize the REQ0196 and the identified dependant requirements. This information and all other links are captured in the corresponding trace model. The requirement internal links are provided within this SRS document. The link information between requirements and high-level design is provided in [27].

4.2 FUTURE EXTENSIONS

Two additional categories of requirements have been foreseen in [26] for use in the HAAG system requirement specification, which have not been adopted in the current version of the HAAG SRS. Those two categories are “inverse requirements” and “use cases”.

Inverse requirements supplement the list of “normal” requirements with statements about what the system shall not do. They are defined in order to clarify and complement other requirements. As the overall requirements specification is considered to have a sufficient level of precision and for the sake of simplicity of the dependencies between the requirements, the inverse requirements have not been used in the HAAG SRS.

Use Cases are a means to describe system properties in an abstract way by describing how users (or other systems) do interact with a system. This may help in early phases of requirements gathering and in particular, when the general functionality of a system is rather vague. As use cases are used again in the high-level architecture specification in order to show what functionality of the HAAG (i.e. which interaction with the system) is realised with what part of the system design, it was decided to not include use cases in the SRS and thus prevent possible overlaps and redundancy in the documentation.

Similarly, it has been decided to exclude the security requirements from the SRS document in order to avoid duplication with the HAAG PP, where these requirements are contained [25]. However, the trace information from security requirements to high-level requirements, as well as the risk assessment diagram addressing the security threats, are included in the SRS document as this information is not contained in the HAAG Protection Profile.

The requirements stated in the SRS document are gathered as part of an analysis of existing design studies and through the definition of a security protection profile [25]. However, an additional requirements solicitation phase would be needed in order to detail the requirements for the functional part in a way that is comparable with the security requirements as covered by the protection profile.

5 CONCLUSIONS

Our system design process incorporates a structured way of collecting requirements and takes into account a security risk assessment of the system. The process is based on industry standards and best practices. It is accompanied by a definition of a Common Criteria Protection Profile, which captures security requirements for the HAAG. All phases of the system design process are performed using an integrated modelling environment based on Eclipse and open-source tools. The environment allows us to build and maintain a relatively complex model and, to a large extent, automatically generate the required design documentation. Despite the immaturity of some of the open-source tools, our environment proves to be cost effective, technically adequate and sufficiently user-friendly for the development of a complex high-level architecture for the HAAG. We think that our approach is a promising candidate to support the use of NAFv3 when designing security-critical systems for NATO.

REFERENCES

[1] Armando, A., Oudkerk, S., Ranise, S., Wrona, K., “The HAAG Security Model and Policy Language”, NCI Agency Technical Report TR/2012/SPW008418/11-4, NCI Agency, The Hague, Netherlands, 2013 (NATO Unclassified).

[2] Boehm, B., "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes", "ACM", 11(4):14-24, August 1986

[3] Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model, Version 3.1 Revision 4, CCMB-2012-09-001, September 2012.

[4] Eclipse Foundation (on-line), www.eclipse.org, Eclipse Framework, 2012.

[5] Electronic Industries Alliance (EIA), EIA-632, “Processes for Engineering a System, Government Electronics and Information Technology Association a Sector of the EIA”, 1998 http://geia.org

[6] Eclipse Modeling Framework (on-line), http://www.eclipse.org/modeling/emf/, Eclipse Project, 2012.

[7] Forsberg, K., Mooz, H., “Proceedings of the First Annual NCOSE Conference”, 1990.

[8] Forsberg, K. Mooz, H. and Cotterman, H., “Visualizing Project Management”, Wiley, 2000.

[9] Guimaraes, L. R., Viela, P. R., “Comparing Software Development Models Using CDM”, SIGITE, (339-347), Newark: NJ, 2005

[10] Humphrey, W., “The software engineering process: definition and scope”, Proceeding ISPW '88, 1988.

[11] IEEE 830-1998, Recommended Practice for Software Requirements Specifications, 1998.

[12] IEEE 1016-2009, Standard for Information Technology - Systems Design - Software Design Descriptions, 2009

[13] IEEE 12207-2008, Systems and software engineering - Software life cycle processes, 2008

[14] IEEE 1471-2000, Recommended Practice for Architectural Description for Software-Intensive Systems, 2000

[15] Larman, C., “Agile and Iterative Development: A Manager's Guide” Addison-Wesley, 2004

[16] Lund, M., Solhaug, B., Stølen, K., “Model-Driven Risk Analysis, The CORAS Approach”, Springer Verlag Berlin Heidelberg, 2011

[17] Wrona, K., “High Level Design for the NATO High Assurance Guard”, NC3A Reference Document 3381, NC3A, The Hague, Netherlands, 2012 (NATO Unclassified).

[18] Object Management Group, “Model-Driven Architecture”, http://www.omg.org/mda/, 2012

[19] Object Management Group, “Requirements Interchange Format (ReqIF)”, Version 1.0.1, 2011

[20] Object Management Group, “Object Constraint Language (OCL)”, Version 2.3.1, 2012

[21] Eclipse Papyrus, http://www.eclipse.org/modeling/mdt/papyrus/, Version MDT, 2012

[22] Royce, W., "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON, 1970.

[23] Booth, M., Henwood, J., Kubicka, A., “IEG Scenario D Use Case and Analysis”, NC3A Reference Document RD-2974, NATO C3 Agency, The Hague, 2010 (NATO Unclassified).

[24] Oudkerk S., “NC3A XML-Labelling Guard Version 2.0 Documentation Package”, NC3A Reference Document RD-2904, 2011 (NATO Unclassified).

[25] Wrona, K., Menz, N., “Protection Profile for the NATO High-Assurance ABAC Guard – Version 1.3, TR-2012-SPW008418-13-4, NCI Agency, The Hague, Netherlands, 2013 (NATO Unclassified).

[26] Hein, Ch.; Ritter, T., Wrona, K., “Deliverable D1.1, Proposed revisions to the HAAG High Level Design”, TR/2012/SPW008418/12-1, NCI Agency, 2012 (NATO Unclassified).

[27] Hein, Ch.; Ritter, T., Wrona, K., “HAAG High Level Design, including the UML design documentation”, TR/2012/SPW008418/12-4, NCIA Agency, 2013 (NATO Unclassified).

[28] Wrona, K., “Feasibility study on development of NATO High Assurance Automated Guard”, NC3A Reference Document RD-3084, NC3A, The Hague, The Netherlands, 2011 (NATO Unclassified).

[29] Ross, A., Oudkerk, S., “NATO Content Inspection Policy Enforcement Framework Functional Specification, Technical Note TN-1486, NC3A, The Hague, Netherlands, 2012 (NATO Unclassified).

[30] Oudkerk, S., “NATO Profile for the “Binding of Metadata to Data Objects”, Version 1.0, NC3A Technical Note 1455, NC3A, The Hague, Netherlands, 2011 (NATO Unclassified).

[31] Oudkerk, S., “NATO Profile for the XML Confidentiality Label Syntax”, Version 1.0, NC3A Technical Note 1455, NC3A, The Hague, Netherlands, 2011 (NATO Unclassified).