J. Software Engineering & Applications, 2010, 3, 894-900
doi:10.4236/jsea.2010.39105 Published Online September 2010 (http://www.SciRP.org/journal/jsea)

Scientific Research

Introduction to a Requirements Engineering Framework for Aeronautics

Robert Abo

Cedric Laboratory, Conservatoire National des Arts et Métiers, Paris, France.
Email: robert.abo@cnam.fr

ABSTRACT

This paper introduces a framework to produce and to manage quality requirements of embedded aeronautical systems, called the 'Requirements Engineering Framework' (REF). It aims at making the management of the requirement lifecycle easier, from the specification of the purchaser's needs, to their implementation in the final products, and also their verification, while controlling costs. REF is based on the main standards of aeronautics, in particular RTCA DO-254, and RTCA DO-178B standards. An implementation of REF, using the IBM Rational DOORS and IBM Rational Change tools, is also presented in this paper.

Keywords: *Aeronautics, Requirements Engineering, RTCA DO-254 and RTCA DO-178B Standards, V-Model*

1. Introduction

To ensure the safety and the reliability of the aircraft's embedded systems, airworthiness authorities (e.g. *US Federal Aviation Administration* [1], *European Aviation Safety Agency* [2], *UK Civil Aviation Authority* [3], etc.) require that they are built under control of processes based on international standards. Among these standards, the main two used in the civilian domain are the well-known RTCA DO-254 '*Design Assurance Guidance for Airborne Electronic Hardware'* standard (aka EUROCAE ED-80) [4] for hardware components and the RTCA DO-178 ed. B '*Software Considerations in Airborne Systems and Equipment Certification*' standard (aka EUROCAE ED-12) [5] for software components. They are referred to as the 'DO standards' throughout this paper.

In this article, we introduce the '*Requirements Engineering Framework*' (REF for short), which aims at producing and managing quality requirements, in order to produce safe and secure embedded aeronautical systems, that must adhere to the rigorous constraints of international standards, while controlling costs. This is achieved by using formalized and mature processes as presented in the following sections. The REF described in this article, does not refer to the practices of a particular supplier or a particular firm in aeronautics.

The rest of this paper is organized as follows. In Section 2, we present the basic notions of requirements management, which form REF foundations. Section 3 presents an implementation of REF, which uses the IBM Rational DOORS tool [6] to manage requirements and to carry out requirement traceability, and IBM Rational Change tool [7] to manage changes between work teams. Section 4 is dedicated to the safety activities, while Section 5 concludes this paper.

2. Requirements Management

2.1. System Lifecycle Model

DO-254 does not prescribe a preferred lifecycle model, nor imply a structure for the performing organization. In the same manner, DO-178B does not designate a preferred software lifecycle, but describes the separate processes that comprise most lifecycles and the interaction between them. The lifecycle for each project should be based on selection, and arrangement of processes and activities determined by the attributes of the project.

Several system lifecycle models exist in system engineering, with different approaches on the manner of leading a project to develop a system: waterfall, V-model, iterative, spiral, agile, and so on. Each one has its pros and cons, and it is up to the chief technical officer and project leaders to determine the most suitable model to lead the projects of their company.

REF is based on *V-model* [8] (aka "Vee model"), which is a variation of the waterfall model. This choice is explained by its advantages. First, it is simple, well organized, and easy to use and to implement. In particular, it highlights the correspondences between the develop-

ment phases (*i.e.* the descending stages, from the early specification to the implementation) and the verification phases (*i.e.* the ascending stages, from the implementation to the product delivery). Thus, it facilitates not only requirement traceability, but also the production of the certification documents required by DO standards, as we will explain in the following sections. Another great advantage of V-model is it can be tailored into a specific project-oriented V-model, because it is independent from any organization and any project. It also provides assistance on the way to implement an activity, and it supports a wide range of development methodologies, in particular *formal methods* [9-11] often use to develop critical parts of systems.

Among disadvantages of V-model, it is project-oriented instead of addressing the development of systems within a whole organization. Another V-model disadvantage is it fails at covering the maintenance of systems. But, these disadvantages do not impact REF.

2.2. Basics

The concept of *requirement* is in the middle of systems engineering, as the abundant literature on the subject attests it [12-15]. We define a 'requirement' as a customer's elementary need that is to be implemented in the product or service that he receives[1]. In systems engineering, we can refine this rough definition by distinguishing the characteristics of the system to be built, known as the *functional requirements*, from the ways the system achieves its functions, known as the *non-functional requirements* (e.g. performance, quality, interface requirements, etc.). We can also differentiate the customer's needs, from which the supplier's distributed requirements are issued, among three hierarchical levels, which are the system, the high-level and the low-level requirements sets. From now on, by "customer", we mean not only the purchaser of the building system, but also the supplier's teams who require services from other ones along an enterprise workflow dedicated to requirements management. Thus, we distinguish four main requirement levels according to their refinement level, plus a requirement implementation level as shown in **Figure 1**:

1) The 'purchaser's level' corresponds to the purchaser's specifications seen as a set of rough needs developed in the 'Purchaser Specification' (PuS) document.
2) The 'system level': the purchaser's needs are refined and reformulated, by using technical terms understandable for the development teams. The system requirements are collected in the 'System Specification' (SyS) document. It is possible to refine this level, by considering a sub-level dedicated to the embedded equipment.
3) The 'high-level requirements (HLR) level'. The notion of sub-system appears, and hardware requirements are distinguished from software ones at this level. High-level requirements are developed from the analysis and refinement of system requirements, system architecture, safety-related needs and derived requirements. The latter correspond to requirements that are the result of the sub-system development process, and may not be directly traceable to high-level requirements. The HLR are collected in the 'Hardware Requirement Specification' (HRS) and the 'Software Requirement Specification' (SRS) documents.
4) The 'low-level requirements (LLR) level'. Low-level requirements are developed from the high-level requirements, sub-system architecture, and design constraints, by refinement and reformulation. The hardware and software subsystems are directly developed from the LLR. The LLR are collected in the 'Hardware Design Document' (HDD) and the 'Software Design Document' (SDD).
5) The 'implementation level' is the last level and marks the end of the descending phase of the V-model. It corresponds to the hardware components and the source code. The implementation of a requirement consists in giving this requirement an existence from its specification as it appears in the HDD (for hardware components) or in the SDD (for software components).

Requirements are fundamental. Firstly, the supplier's requirements formalize the customer's needs. The supplier ensures the comprehension of the customer's needs, that he has translated this into a form he can use without any misunderstanding. Secondly, requirements allow the identification of the characteristics of the customer's needs. Finally, requirements simplify the taking into account of customer's needs along V-model by formalizing them. They show the customer that the final product answers the needs he has expressed.

2.3. Requirements Specification

It consists of specifying the requirements. In particular, engineers have to define the bi-directional and vertical traceability between the upper and lower requirements. The main objective of the requirement traceability is to show that the purchaser's needs are satisfied by system requirements, high-level requirements, and low-level requirements; and then implemented into the hardware

[1]DO-254 defines a *requirement* as "an identifiable element of a specification that is verifiable" [4]. DO-178B defines a *software requirement* as "a description of what is to be produced by the software given the inputs and constraints" [5].

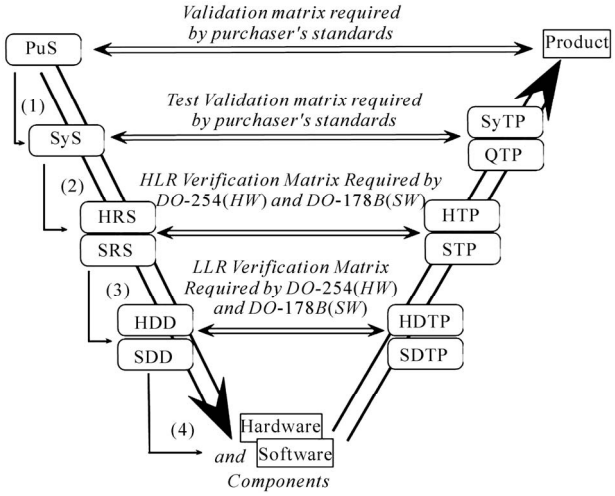

Figure 1. The documents of a project, issued at the different stages of V-model with (1): System level requirement validation matrix; (2): Specification Analysis Matrix (DO); (3): Design Analysis Matrix (DO); (4): Hardware/Code Analysis Matrix (DO). The requirements are produced by successive refinements along the descending phases of V-model. In the figure, 'TP' stands for *Test Plans*, and 'Q' for *Quality*.

components or the source code.

2.4. Requirements Justification

All the supplier's requirements at any level have to be justified. A justification records the reasons for a requirement's existence, and its compliance with a customer's need. It also records the reasons for the implementation choices; and it keeps the analysis for the future designs and the modification assessments. Finally, it justifies the activities link to requirements, in particular the safety ones. Requirement justifications make the requirement analysis phase easier.

2.5. Requirements Review and Analysis

This phase is also referred to as "requirements validation". Its purpose is to ensure that all the customer's needs are specified (*i.e.* there is no under-specification of the customer's needs) and nothing more than these needs is specified (*i.e.* there is no over-specification of the customer's needs). Moreover, this analysis consists in ensuring that the requirements at each level are good and well-specified requirements, *i.e.* they are sufficiently correct, complete, unambiguous, consistent, self-contained, achievable, verifiable, etc., so the delivered product will meet all the customer's needs and airworthiness authorities' constraints including DO requirements.

We must notice that whether the writers and the reviewers are the same engineers, they cannot perform the validation of the requirements they specified, in particular for the requirements of the most critical software referred to as Level A or Level B by the DO-178B standard[2]. Project managers and team leaders must organize the work of the engineers taking this into account. A specific team performs the safety activities as described in Section 4.

2.6. Requirements Verification

This activity deals with the rise of V-model. It consists in evaluating the implementation of the supplier's requirements to determine, whether or not, they have been met. There are several means of verification: tests, code analysis, model checking, simulation, etc. For aeronautics real-time embedded software, the low-level requirements are often implemented by using the Esterel Technologies' SCADE Suite [16]. This tool complies with DO-178B, and allows for generation of a certified source code from low-level requirements without any unit tests.

3. Implementing REF

The REF processes are implemented through two main tools namely: IBM Rational DOORS [6] for the management of requirements, and IBM Rational Change [7] for the management of changes impacting requirements. This choice and the use of these tools are not mandatory, and other ones with similar functionalities can be used, according to the final customer's choices. Reviewing all of them is out of the scope of this paper, but we can quote Geensoft's Reqtify [17] or IBM Rational RequisitePro [18] as other examples of requirements management tools. IBM Rational ClearQuest [19] and Serena TeamTrack [20] are other examples of change management tool.

3.1. Requirements Management

DOORS is a requirements management tool that provides an easily collaborative environment, to make the achievement of processes linked to the specification, the analysis, the verification and the traceability of requirements easier.

3.1.1. Data Organization

Data is stored in DOORS databases, each of which are organized as folders, projects and modules. Projects are specific folders that contain data related to a particular project. They can contain folders and sub-folders, both contain modules. We define a module as a collection of objects with attributes, each of which relate to a particular topic. Each module has its own attributes as name, type, description, date of creation and so on. Different

[2]Software level is based upon the contribution of software to potential failure conditions as determined by the system safety assessment process. Their effects on the aircraft, the crew and the passengers categorize the failure conditions. They spread out from 'A' (catastrophic effects), to 'E' (no effects) [5].

kinds of modules can be defined.

Each project should contain at least:

1) Modules for customer specification;
2) Modules for system, high-level and low-level requirements;
3) Modules for applicable standards, documents, and plans;
4) Modules for requirement verification (test cases, test procedures, results, and analysis);
5) Modules for requirement justification;
6) Modules for requirement validation.

Within a module, objects can be organized in a hierarchical manner. Information is displayed through views that can filter attributes according to user choice. Objects can be linked together, in particular hierarchical objects, which is very important to define objects traceability. It is possible to define several kinds of objects:

1) Requirements collected in the specification modules;
2) Validation objects collected in the validation modules;
3) Justification objects collected in the justification modules;
4) Verification objects collected in the verification modules;
5) Other objects in particular texts, that can contains titles, notes, remarks or any other textual explanations that are not requirements but are useful to understand the specifications. Indeed, we must keep in mind that these modules can be published as official documents for the purchaser and the end users.

DOORS administrators can regularly create module baselines, which are frozen modules that cannot be modified. They record the history of the module since its last baseline, including information about objects, their attributes, and also module sessions.

3.1.2. Documents Issues

DOORS allows exporting a module into several formats, that can be Microsoft Office, HTML, FrameMaker, etc. This functionality is particularly interesting to deliver definitive documents to purchasers. It is possible to choose the attributes to be printed on documents extracted from DOORS modules. In that case, the text of the requirement is automatically put between the identification of the requirement and the 'End of Requirement' tag. The attributes to be printed should be, at least:

1) The requirement identifier;
2) The requirement text;
3) The upper requirement(s) covered by this requirement;
4) The delivery version of the product where this requirement appears.

3.2. Change Management

3.2.1. Basics

The configuration management process is interfaced with IBM Rational Change [7]. Specifications, test cases, test procedures and any documents are managed with DOORS. Change is a web-based tool for change management solutions, allowing teams involved in the system development to get together. Across the enterprise, it tracks change requirement requests.

3.2.2. Process Description

Updates of requirements, justification, and validation objects are decided by a committee. They are only authorized through a change management process described in the following text. Each modification or evolution need is recorded through a *Specification Change Request* (SCR) that details the origin of the evolution, the standard of applications and the evolution need. This SCR can lead to several *Requirement Change Requests* (RCR), each of them impacting one or several requirements of a specific module. The Change tool traces the links between an SCR and its RCRs. Each RCR is realized in DOORS. Thus, each requirement modification must be traced with the relevant RCR. Once the SCR is approved in commission, the requirement or procedure is then proposed for the validation process. An SCR or an RCR can be reworked, if conflicts are detected. The SCR manager can close an open SCR after having checked it:

1) All impacted requirements have been validated;
2) All modifications are well traced in DOORS;
3) All verification modules have been updated;
4) All impacts on lower and upper requirements have been taken into account;
5) All justifications have been updated;
6) All impacts on previous standard specification have been taken into account;
7) The standard of applicability has been clearly identified.

Figure 2 shows the SCR and its associated RCRs lifecycles, with the corresponding processes enabling to pass from a stage to another.

3.3. Requirements Documentation

Some attributes are generic and DOORS automatically manages them. These usually are the object identifier, its date of creation, its date of last modification, the name or the user identification, etc. The object identifier is unique, and must contain the identification of the module that the requirement belongs to, and a number. The module identifiers should be, at least: SYS for 'System', HW for 'Hardware', SW for 'Software', SAF for 'Safety', VAL

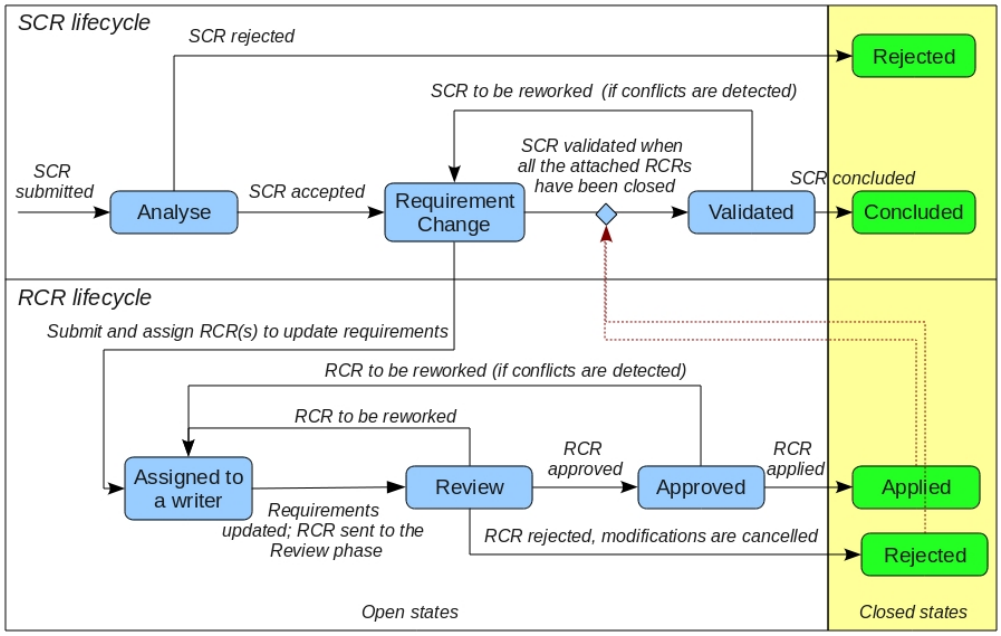

Figure 2. the SCR and RCRs lifecycles for REF. In general, several RCRs are associated to one SCR. RCRs permit to trace requirements updates.

for 'Validation', JUS for 'Justification', and others necessary identifiers as, for example, QLY for 'Quality', PRG for 'Programs', etc. For requirements, there must be the following major attributes. They have an impact on the validation status.

1) A main description to describe the requirement. It may contain drawings, tables, figures or mathematical formulas.
2) An assumption or a set of assumptions for the requirement, if any. Assumptions must be identified, justified, and validated.
3) The domain of activity, for example, SYS for 'system' level, HW for 'hardware', or SW for 'software' level.
4) The type of requirement: 'derived' requirements, which are the results of the sub-system development process and may not be directly traceable to high-level requirements. A 'terminal' requirement cannot be traced to lower levels. A 'normal' requirement is neither derived nor terminal.
5) The delivery version of the system in which the requirement appears (for example V0, V1.0, V1.1, etc.). It is possible to qualify a version as 'partial' to indicate requirements are partially implemented in it.
6) Links to requirements not under the DOORS control.

Even if it is obsolete, a requirement must never be deleted. This basic rule is necessary to avoid losing traceability and to keep a trace of its existence. Besides, this deletion must be justified in the justification object linked to the deleted requirement.

Low-level requirements have specific attributes as the identification of the function that calls it, the description of its input and output parameters, etc., plus a data dictionary in which all data, types, variables, constants, and definitions of applications are defined.

3.4. Requirements Justification

The DOORS justification module embeds three categories of justification objects expected for certification issues:

1) Justification of all the requirements (normal, derived, and terminal).
2) Justification of the validation of requirements.
3) Justification of safety assessment of derived requirements.

As far as possible, the requirement justification process must be complete before entering the requirement validation phase as the latter contains a checklist of criteria to ensure completeness and correctness of this activity.

3.5. Requirements Review and Analysis

We perform two kinds of requirement analysis: the transversal and the unitary analysis.

3.5.1. Unitary Analysis

It is requirement-oriented. The requirement conformity with the DO standard criteria applicable to requirements

is checked using DOORS. All requirements are analyzed one by one: the system requirements; the hardware high-level and low-level requirements (DO-254 Subsections 6.1.2.2, 6.1.2.4 and 6.1.2.5) and also the software high-level and low-level requirements (DO-178B, Subsections 5.3.2 and 6.3.1). We check the quality of each requirement *i.e.*:

1) Its *adaptability* for its level of specification, e.g. no detailed requirement at system or high level, or no rough and non refined requirement at low level (requires by DO-178B Subsection 5.1.2 g for SW);
2) Its *completeness* with no missing information, in particular, concerning the acceptance criteria (requires by DO-254 Subsection 6.1.2.4 for HW and DO-178B Subsections 6.3.1a, b, d and 5.1.2 f for SW);
3) Its *correctness* by expressing a need and not a solution for that need; if possible, the contrary must be rigorously justified (requires by DO-254 Subsection 6.1.2.5 for HW and DO-178B Subsection 5.1.2 g for SW);
4) Its *consistency* by not being contradictory with other requirements of the same level (requires by DO-178B Subsection 6.3.1 b);
5) Its *feasibility* by checking it can be implemented on the target architecture (requires by DO-254 Subsection 6.1.2.5 for HW, and DO-178B Subsection 6.3.1 b, c, d for SW);
6) Its *unambiguity* and *precision* by checking that nobody can interpret it (requires by DO-254 Subsection 6.1.2.5 for HW, and DO-178B Subsection 6.3.1 b and d);
7) Its *verifiability* by checking that its verification is possible (requires by DO-254 Subsection 6.1.2.5, and DO-178B Subsection 6.3.1 b, d for SW);
8) Its *traceability* by checking links with upper and lower requirements (requires by DO-254 Subsection 6.1.2.4 and DO-178B Subsection 6.3.1 a);
9) Its *conformance to standards* (requires by DO-178B Subsection 6.3.1 e);
10) Its *algorithms* (if any) must be *accurate* and *correct* (requires by DO-178B Subsection 6.3.1 g);
11) Its *topicality* by checking it does not refer to an obsolete part of the system.

NB. Software scripts can be used to check general rules automatically, that major attribute fields are not empty, editing requirement rules are complied with, etc. For this, each attribute must be correctly filled in.

3.5.2. Transversal Analysis

It is document-oriented. The DO standard criteria applicable to a document are used to validate the whole document from a quality point of view. It consists in checking several points among which:

1) Its availability and its consistency;
2) Its compliance with the purchaser and airworthiness standards;
3) The completeness of its references;
4) Its readability;
5) Its compliance and traceability with upper documents if any;
6) Its correctness, completeness and accuracy;
7) Its compliance with development standards;
8) Its maintainability.

3.6. Requirements Verification

Each requirement is associated to one or more test cases, with each of them specifying the test objective with a description. If the test case defines a test of the product (laboratory, vehicle, flight, environment, etc.) then a script or detailed procedure and the associated test results shall be written. If the test case is defined by analysis, a detailed procedure is used to reach the test result. Test cases shall only specify the objective of the analysis. Test results shall contain the full analysis and the result status for each standard. Then three levels of verification modules are provided:

1) The test case level aiming at containing test case(s) covering requirements. A tests case describes test sequences, objectives, input/output conditions, required environment and accepted criteria from a general point of view: no implementation details linked to test benching or particular tools need to be described, unless there are particular constraints.
2) A detailed test procedure or script level that is the implementation of test cases with regards to test bench facilities, software capacities, specific tools to be used, or other precise implementation details required to ease test runs and avoid mistakes in test procedure execution. Test scripts are dedicated to automated procedures and detailed procedures to manual tests. Both can be used for tests requiring manual sequences. For test cases by analysis, the detailed procedure is used to reach the test result.
3) A test result level containing all the verification results.

4. Safety Analysis

The safety activities are exclusively related to the needs impacting the safety of the system to be built. They affect the documentation, the justification, and the validation of safety-related requirements. An independent team of en-

gineers, referred to as the "safety team", performs the safety activities, that are based on the analysis of all the safety-related requirements (normal, terminal, and derived) that contribute to reach the customer's safety needs. A set of safety-oriented attributes is defined for each requirement.

4.1. Safety Activities in Specification Modules

A special attribute should be used to mark any safety-related requirement. It must adhere to the lower requirements in order to identify requirement trees that need a safety analysis precisely. If a requirement is not safety-related, its attribute shall be set to 'NO'. Safety teams shall be specially warned of every evolution of this attribute for each requirement. All updates of this attribute for any requirements must imply a new safety validation phase. When it is set to 'YES', this attribute must be visible in the published version of specifications.

4.2. Safety Activities in Justification Modules

Different attributes should be used to justify the safety aspect of a requirement. The first attribute should state whether a requirement has an impact on the safety analysis and must require special attention. The second should detail the reasons why the previous attribute was filled as 'YES'. Another one should detail the analysis performed by the safety team in order to comply with the safety objectives. Some other justification attributes should be added.

4.3. Safety Activities in Validation Modules

Only the safety team fills out the attributes of these objects. They should record at least, the accepting of the requirement in accordance with the safety criteria, the reasons of the acceptance or the rejection, the name of the engineer who performed the validation, and the date of the validation in order to ensure it is still current.

5. Conclusion

This paper presents a general framework, which we have called the "Requirements Engineering Framework" or REF for short, dedicated to the management of requirements of aeronautical systems, during their whole lifecycle. It aims at producing quality, secure and safe systems in accordance with the rigorous DO constraints, while controlling manufacturing costs. This framework can be implemented in several ways according to the specific needs of suppliers. In this paper, we have outlined the interests of using DOORS [6] and Change [7] tools to implement REF.

In a future paper, we envisage to describe the possible implementations of REF in greater detail.

6. Acknowledgements

I would like to acknowledge the discussions and suggestions from different persons including my work colleagues and my friends. I would like to acknowledge especially Prof. Kamel Barkaoui of the *Conservatoire National des Arts et Métiers* in Paris, France, who encouraged me to write this paper.

REFERENCES

[1] "US Federal Aviation Administration". http://www.faa.gov/

[2] "European Aviation Safety Agency". http://www.easa.eu.int

[3] "UK Civil Aviation Authority". http://www.caa.co.uk

[4] RTCA DO-254 (EUROCAE ED-80), "Design Assurance Guidance for Airborne Electronic Hardware," 2000.

[5] RTCA DO-178B (EUROCAE ED-12B), "Software Considerations in Airborne Systems and Equipment Certification," 2nd Edition, 1992.

[6] "IBM Rational DOORS". http://www-01.ibm.com/software/awdtools/doors/

[7] "IBM Rational Change". http://www-01.ibm.com/software/awdtools/change/

[8] "Das V-Modell". http://v-modell.iabg.de/ (some pages about the fundamentals of V-model are in English).

[9] C. M. Holloway, "Why Engineers Should Consider Formal Methods," *Proceedings of the* 16*th Digital Avionics Systems Conference, Irvine,* California, October 1997.

[10] N. A. S. A. Langley, "Formal Methods web site". http://shemesh.larc.nasa.gov/fm/

[11] "Formal Methods in System Design Journal," Springer. http://www.springer.com

[12] R. R. Young, "Effective Requirements Practices," Addison Wesley, Boston, 2001.

[13] K. E. Wiegers, "Software Requirements," 2nd Edition, Microsoft Press, 2003.

[14] K. E. Wiegers, "More About Software Requirements," Thorny Issues and Practical Advice, Microsoft Press, 2006.

[15] V. I. Fort Belvoir, "Systems Engineering Fundamentals," Defense Acquisition University Press, USA, 2001.

[16] "Esterel Technologies SCADE Suite". http://www.esterel-technologies.com/products/scade-suite

[17] "Geensoft Reqtify". http://www.reqtify.com

[18] "IBM Rational RequisitePro". http://www-01.ibm.com/software/awdtools/reqpro/

[19] "IBM Rational ClearQuest". http://www-01.ibm.com/software/awdtools/clearquest/

[20] "Serena TeamTrack". http://www.serena.com/products/teamtrack/change-request-management.html

