Criteria for software to safety-critical complex certifiable systems development

N.K. Gorelits <nkgorelits@2100.gosniias.ru>
A.S. Gukova <asgukova@2100.gosniias.ru>
E.V. Peskov <evpeskov@2100.gosniias.ru>
State Research Institute of Aviation Systems,
7, Viktorenko Str, Moscow, 125319, Russia

Abstract. Nowadays there is an actual problem in aviation industry – how to make the development of complex safety-critical systems certifiable according to international and domestic standards and regulations like DO-178C, DO-254, ARP 4754A, ARP 4761 etc. In the article configuration management process from the development lifecycle of DO-178C is considered as the main source of criteria for the development tool selection. Selected criteria can be applied to software tool, which supports entire development lifecycle of aviation software, as well as to software tools supporting some individual lifecycle processes. The activities of configuration management process provide work with all project lifecycle data, its storage, integrity, security, manageability and information support for data exchange between the remaining lifecycle processes, maintenance of the history of changes etc. Compliance with the principles of the configuration management process allows project managers to control development, ensure the required quality and reliability of the product; also, its certifiability and the necessary level of confidence in security, reduce financial and time development costs. As example of using criteria one of the most widely known in industry software tool for requirements development and management was analyzed for compliance with the chosen criteria.

Keywords: DO-178C; qualification requirements 178C; software development; software analysis; software choosing; certifiable systems; complex systems; complex systems development; avionics; on-board equipment; lifecycle processes; lifecycle; configuration management; system engineering.

DOI: 10.15514/ISPRAS-2018-30(4)-4

For citation: Gorelits N.K., Gukova A.S., Peskov E.V. Criteria for software to safety-critical complex certifiable systems development. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 4, 2018, pp. 63-78. DOI: 10.15514/ISPRAS-2018-30(4)-4

1. Introduction

This research was inspired by acquaintance and very productive work communication with untimely gone Michael Saburov. Michael Saburov participated in development of Russian analogs of certification standards and regulations DO-178B [1], DO-178C

[2], DO-254 [3] and DO-330 [4]. Also Michael Saburov participated in implementation of processes from these regulation documents in several industry enterprises. Michael Saburov took an active part in formation of the concepts of research described in this article. All results and experience gained by us during work on this research are dedicated to Michael Saburov.

Development and the following certification of complex safety-critical systems in compliance recommendations of regulation documents DO-178C, DO-254, ARP 4754A [5], ARP 4761 [6] is an actual task and a big challenge for modern Russian aviation industry.

Today among the software announced by its developers as supporting lifecycle of complex systems development a huge number of products are presented to allow software development in accordance with international quality standards.

Nevertheless at the moment assessment of the capabilities of each tool (or often it will be the whole product line of expensive tools) and making a reasonable choice is rather difficult problem.

Big quantity of existing software tools and systems positioned by developers as tools, which support lifecycle processes of complex systems development, don't have well-founded assessments.

Assessments and reviews about such software, based on experience of practical usage in industry projects, are very important – software market proposes a lot of software tools and systems made by Russian and foreign developers. So that's why industrial enterprises have to make difficult choice of software tools for development and the following certification of their critical-safety systems.

It is difficult to choose instrumental environment for support the entire development lifecycle – unfortunately universal multipurpose tool, which would satisfy the requirements of all standards of all industries, does not exist yet.

In general, most of the enterprises use separate tool for support and automate each process of development lifecycle (like requirements development process, configuration management processes, verification etc.). The situation is complicated because often all or the most parts of such software suite have different manufacturers. If the project have big set of weakly integrated software, then product development becomes more and more complex both in atomic tasks of individual specialist and in global meaning of the whole project – labor intensity increases.

The organization of development landscape as a bunch of software tools entails difficulties with tools integration, training costs, implementation costs, purchase of licenses. All these changes increase the amount of resources, which are needed for successful completion of the processes – human resources, financial, and time resources. In this case, reaching project goals, formulated before the beginning of work, become more and more difficult task.

In conditions of State program of import substitution [7] software tools and systems made by Russian developers cause big interests. However, usage experience of

Russian software and consequently number and reliability of assessments according to requirements listed above standards and regulations are not big enough.

In this article, we tried to understand and present what mechanisms and features software tool should have to be useful to simplify and systematize development of certifiable aviation software. This article is a part of series of materials about aviation standards research in context of choosing software tools for certifiable aviation software development [8].

2. DO-178C processes and the role of configuration management process among them

Russian analogue of DO-178C - Qualification requirements 178C [9] – regulates processes of certifiable development of aviation software. The heading of Russian document contains important words – "Requirements to the software of on-board equipment and systems at certification of aircraft". These words uniquely determine goals of recommendations, specified in the document.

Certifiability of product – significant property, because the purpose of most developments is the following release of end product the on relevant market. In the context of aviation systems certifiability means that aircraft with system included will receive type certificate [10].

Under certifiability assurance, we mean the implementation of the development processes in specific way –

- all necessary for certification activities are performed,
- all necessary for certification objectives are achieved,
- all necessary data is collected about development process and its result,
- this data is stored and processed in such a way that certification authority could receive any data at any stage of project in order to examine the data and to trace the history of their interactions and relationships.

Activities and objectives to airborne systems and equipment development are described in document DO-178C (Russian analogue – qualification requirements 178C). DO-178C provides instructive materials and guidance to create airborne systems and equipment. Implementation of activities and objectives achievement listed in DO-178C give a chance to get in the end the result, which **performs its intended function with a level of confidence in safety that complies with airworthiness requirements**.

DO-178C describes a set of development lifecycle processes for aviation systems and equipment. DO-178C divides processes of the development lifecycle to three groups. The first group includes only one process – **software planning process**. The second group called **software development processes** includes four processes – software requirements process, software design process, software coding process and integration process. The third group consists of four **integral processes** – software

verification process, software configuration management process, software quality assurance process, certification liaison process.

During software development, processes directly creation of software of aviation systems takes place along with all previous and accompanying it measures for the design, coding, integration etc. The main result of development processes is the executable object code and its associated additional data are produces and loaded into the target hardware for further integration. This result is necessary to be achieved having carried out all the measures described in qualification requirements.

Integral processes play a role of enabling processes (by analogy with enabling systems in the terms of System Engineering [11]) - created and edited during development processes data is stored and processes through mechanisms and activities of configuration management process, required reviews and analyses are made in the verification process and so on. Data – development lifecycle artifacts or configuration items – may be requirements with different levels of details, software architecture, source code and executable object code and different protocols, problem reports, and many other results of activities.

Explanation the importance of integral processes implementation is very simple – otherwise it is very difficult almost impossible to collect necessary for certification data and to control the development process. It means that it will be difficult to provide **necessary level of confidence in safety that complies with airworthiness requirements**.

Each of integral processes has its own role and importance in the development lifecycle; it could not be ignored or partially abolished during lifecycle. Huge risks await developers who dare not comply integral processes - certification authority will not accept results obtained this way and will not give relying certificate. Also final product may contain errors and defects of varying degrees of critically. This situation will not allow achieving the required level of confidence in safety and quality of result in total, if the development process comes to an end with the release of the working result.

In modern world of computers and upcoming information technologies the whole software development lifecycle (and aviation software is not an exception) passes through software tools, information systems and therefore its databases and repositories. These software tools and information systems for all kinds of operations on data (creation, storage, editing etc.) must be evaluated for their sustainability and compliance with development according to certain standards and other regulation documents.

If perform analyze requirements to development product, which Qualification requirements 178C specifies and requires developer, becomes obvious that the most restrictions and requirements for software (in which aviation software will be developed) come from configuration management process. Activities of configuration management process provide operations with development lifecycle data, its storage, informational support to data exchange between other lifecycle processes, logging the history of changes etc.

In this research, we chose configuration management process as the source of arguments or justifications for choosing of software tools on which certifiable aviation software will be developed. These substantiations are formulated in the form of criteria. Criteria can be applied to potentially interesting software tools and systems from the market and help with assessment and reasonable choice of some of them. There will be described below how to apply selected criteria to the most widely used (worldwide and also in Russia) software for requirements development in the industry.

3. Basic criteria to tool from configuration management process

Configuration management process in project must be performed in accordance with the document "Software Configuration Management Plan". Software Configuration Management Plan should be developed for each software development project during Software Planning Process if development corresponds to Qualification requirements 178C. In this document configuration management environment should be determined as well as configuration management process activities which will be performed during software development lifecycle.

Configuration management environment must support activities from section 7.2 of Qualification requirements 178C. The list of configuration management activities contains some process regulations (which restrain project members within the workflow) and requirements to the mechanisms of configuration management environment. It would be very useful if such mechanisms and methods will be implemented in software, which will be used for development because not all of them could be replaced with some organizational regulations.

Configuration management plan contains some requirements to configuration management activities follow-up. As examples of these requirements can be listed: states of configuration items, workflows of problem reports and change requests, inspection procedures, baseline definition rules and rules of versioning configuration items, organizational restrictions, safety details etc. These requirements will not be considered in this article because its implementation can be realized regardless of the instrumental part of configuration management environment.

In this article, we identified the basic principles and mechanisms (basic criteria) determined by configuration management environment and configuration management activities according Qualification requirements 178C.

First of all we would like to highlight single and unified storage for all lifecycle data as basic configuration management principle. It means that project should have unified configuration management system for registration, storage and delivery all software development lifecycle data.

Let us enumerate basic mechanisms of configuration management environment:

- identification of configuration,
- configuration status accounting,

- change management and control,
- traceability,
- versioning,
- registration of inconsistencies and corrective actions,
- storage, retrieval and release.

Described mechanisms (further, criteria) are based on text Qualification requirements 178C and are advisory in nature. These criteria can be used as an additional informational source while choosing software tool for certifiable aviation software development.

Elements of the criteria list will be considered in more detail below.

3.1 Identification of configuration and its configuration items

Procedure of identification of the configuration item (and the whole configuration in general) includes assigning an identifier to the configuration item and registering it in the configuration management system. The identifier of configuration item is a designation uniquely distinguishes one configuration item from another. Identifier of configuration item could not be changed ever. Identifier of configuration together with its version makes unique identifier of configuration item in a particular configuration. Version of configuration item will be described below in one of the criteria.

An example of attributes that we suppose useful for registration of configuration item:

- configuration item identifier (doesn't change ever after registration),
- mnemonics (designation which will help user identify configuration item),
- configuration item name,
- purpose of configuration item (type),
- kind of configuration item (atomic, composite – configuration index),
- version (number, sign if it is baseline or not),
- data control category (Control Category 1 or Control Category 2),
- link to the configuration item source.

Note: software lifecycle data can be classified to Data Control Category 1 or to Data Control Category 2 (section 7.3 of Qualification requirements 178C).

3.2 Configuration status accounting

Status accounting of developing software configuration should be conducted in order to provide the certification authority all necessary information (like configuration index, history of configuration etc.). That is why it is necessary to ensure that registration of the actions performed on the configuration units is automatic.

An example of data which we suppose to necessarily register when performing any action on the configuration item:

- date and time of making changes to the configuration item,
- number of version of the configuration item,
- user id – who made changes to configuration item or created version of configuration item,
- status of version of configuration item including the history of this status changes,
- for configuration items from Control Category 1: link to the change request for this configuration item.

3.3 Versioning, baselines

Rules of naming and versioning for configuration items should be defined.

Note: for example, configuration item's version is denoted as an integer (1, 2, 3 etc.). New value of configuration item's version is obtained by increasing the value by 1. If it was 2, the next value will be 3.

Rules for baseline formation and baseline appointment mechanism should be defined. In addition, restrictions on the baseline's modification should be defined.

Note: baseline is approved and registered version of configuration item, which will be used as basic for further development. Baseline can consist of one or several configuration items.

3.4 Configuration items traceability

Traceability requirements and mechanisms should be defined for link different types of configuration items and related data. Configuration items can be connected with each other, also with reason of creation (source), with dependent items, with history of configuration item's changes etc.

Note: As example of connections, we may mention links between low level requirements with its parent high level requirements, low level requirements with executable object code, problem report with configuration item, problem report with change request and with task for making approved changes etc.

Configuration items traceability is very important in the context of developing software certification. It is necessary for configuration items to trace links with source of its creation with maked to configuration items changes and with reason to making changes etc.

Traceability of links should work in both directions. Changes in configuration items should trace to sources of changes (for example to change request, which in its turn refers to parent problem report) and back.

It is always useful for users to analyze some visualized view of data. As a variant of useful and intuitive view of links and traces may be a traceability matrix. Traceability

matrix shows how configuration items are connected to each other and their relations type is displayed. Type of relations between configuration items can be presented both in simple form with only displaying link presence or absence, and in the various types of links and communication.

Table 1 illustrates an example of configuration item's baseline formation.

Table 1. An example of traceability matrix: links between configuration items

Configuration items	***CI1***	***CI2***	***CI3***	***CI4***	**…**
CI1	■	X	⇅		
CI2	X	■		⬇	
CI3	⇅		■		
CI4		➡		■	
…					■

■ – not applicable
X – connection exists
⬇, ⇅, ➡ – certain type of connection exists

3.5 Change management and control

The change management of the configuration items must be implemented. Change management activities are responsible for the reaction to recording, evaluating, solving problems through the whole lifecycle of each configuration item.

Any change of configuration item should only be done by creating a new version of changing configuration item. However all previous versions should remain unchanged. Previous versions should be stored in repository and be accessible.

Changing of configuration items from Control category 1 is possible only through special procedure of change management. Problem report should be created and approved, detailed change requested and tasks should be created from this problem report. Changes to configuration items from change request should be also approved and only then changes may be applied to configuration items. All related information about changes must be stored forever – who, when, for what reason have changed that version of configuration item. Changes to configuration items with Control category 2 do not require complex procedure with approvals and reviews of changes.

3.6 Registration of inconsistencies and corrective actions

Once inconsistencies or defects are detected, it is necessary to determine procedure and mechanisms of its registration. Also corrective actions should be established, impact analysis of the proposed changes should be done and making of the approved changes to configuration item should be strictly controlled.

Any project member who discovered an inconsistency or defect or any other type of error, should be able to write it in special configuration item – problem report.

An example of attributes, which we suppose to necessarily register when registering a problem report for any configuration item:

- link to configuration item – source of detected inconsistencies,
- link to index of configuration which includes configuration item with inconsistence or to process or workflow if inconsistence is more global,
- inconsistence description,
- problem report's author id,
- steps to reproduce the problem,
- problem report state,
- link to corrective actions (for example: change request).

An example of attributes which we suppose to necessarily register when registering a corrective action for any problem report (for example: change request):

- link to problem report (change request source),
- link to configuration items in which it is necessary to make changes,
- impact analysis of proposed changes to the rest configuration items of lifecycle data.

3.7 Storage, retrieval and release

Method and proof of data integrity should be determined during its storage and retrieval from backups. Rights to release data should also be defined. Tools for creation, retrieval and integrity control of backups should be implemented according to chosen method.

Note: the need for backup creation can be both for the entire repository and for a separate development project or for separate configuration.

The realization of instrumental support for the creation, retrieval and data integrity control is very important and in demand because it allows to minimize time costs for these procedures and to reduce the risk of data distortion or loss.

Note: using of a checksum mechanisms for backups creation may be a good example of data integrity control realization.

4. Configuration management tools, analysis

The experience of cooperation with Russian developers of avionics system demonstrates that most of them try to create on-board software in compliance with the requirements of the document Qualification requirements 178B/C and then certify their software products.

At the same time there are situations when the software development process is produced without detailed requirements (in fact without requirements at all - only high-level technical specification are used), without configuration management, without reviews or inspections. Software testing is conducted, but unfortunately, its

completeness can be insufficient because of the absence or incompleteness of requirements.

Realizing their unpreparedness for further certification without using of specialized software, aviation enterprises are implementing various tools. An example of such tools can be IBM Rational DOORS, IBM Rational Change + Synergy, IBM Rational Team Concert, Siemens Team Center Requirements, LDRA and others. In this case often overlooked that without understanding the processes (and not having the described processes on a paper at least) it is almost impossible to get the effect of the implementation of the tool.

It is necessary to apply the certification process with a complex approach to achieve the best result. It means - to develop the processes, to provide their support by tools, to develop plans and standards (Plan for Software Aspects of Certification, Software Development Plan, Software Verification Plan, Software Configuration Management Plan, Software Quality Assurance Plan; Software Design Standards, Software Code Standards, Software Requirements Standards) and to conduct development in full compliance with these plans and standards.

Often enterprise of the aviation industry implement only tool for writing and storage requirements. Typically, this tool has minimal change management capabilities. Developers try to manage requirements ignoring or paying low attention to the configuration management process – this approach is fundamentally incorrect.

Below we put a list of the most widely used tools to support the software development lifecycle, implemented in Russian aviation enterprises.

To support requirements management processes are often used: Microsoft Excel / Word, IBM Rational DOORS, Siemens TeamCenter Requirements Management (mainly in those enterprises where Siemens TeamCenter PLM was previously implemented in the design department) and even more rare - 3SL Cradle.

Due to the State program of import substitution, products of Russian developers arouse great interest. Among the most ambitious, it is possible to highlight product, which supports the entire development lifecycle of systems - Devprom.

To support lifecycle data change management processes are often used: IBM Rational Change + Synergy (tools are not supported by the vendor, but are still in use in some enterprises), IBM Rational Team Concert, and the most popular project and task management tools - Redmine and Attlassian Jira.

In situation when the software product Redmine or Jira are used to manage changes to the lifecycle data, the integration between these tools is rather nominal – all tools supported development lifecycle work independently, links between change requests and requirements are fixed in a text file.

This approach does not contradict the principles of configuration management prescribed in Qualification requirements 178C, but not only doesn't simplify the development process, but also makes the process management even more difficult (dependence on the human factor, the inability to track changes (the absence of a change marker), the lack of quick switch from a change request to the changed data, etc.).

To support configuration management processes are often used: GitHub - the most popular and freely distributed tool among code developers and SVN (Subversion)– a traditionally used repository for file sharing in enterprises in Russia (also distributed under the conditionally free Apache license).

The functionality of these tools when it used as configuration management systems does not allow you to fully support all activities of the configuration management section 7.2 of Qualification requirements 178C. Moreover, the use of all the functionality of this software may be considered as a violation of some of them. It is almost impossible to restrict the functionality of tools that are useful to traditional code developers in order to comply with the process specified in the Configuration Management Plan.

For example, GitHub does not store intermediate versions when you merge code branches (or other files when you use this tool as a configuration management environment) and you cannot track changes that precede the merge.

Quote from DO-178C (section 7.2.4 e): “Throughout the change activity, software life cycle data affected by the change should be updated and **records should be maintained** for the change control activity”.

For the analysis for compliance with the criteria described in the previous section, we present the summarized results of the requirements management tool IBM Rational DOORS use in State Research Institute of Aviation Systems (GosNIIAS) and the results of the analysis of the entire IBM Rational product line for lifecycle management [12].

We can analyze requirements management tools for conformity by Configuration Management process criteria, because the requirement is one type of configuration items and recommendation of section 7.2 of Qualification requirements 178C about its storage and handling must be observed.

To evaluate the criteria, the following values (weight) were selected:

- 0 – criteria is not supported;
- 0.5 – criteria is partially supported;
- 0.75 – criteria is supported through tool configuration, adaptation or any integration;
- 1 – criteria is fully supported.

The analysis results are shown in the figure below on Fig.1.

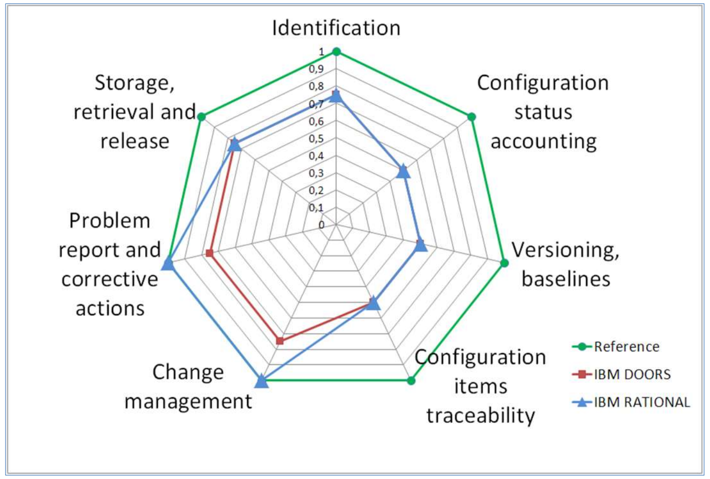

Fig. 1. Tools analysis

5. Conclusion

Configuration management process – is the main source of criteria for choosing the tools, which support aviation software development lifecycle. Configuration management process acts as unifying “input-output bus” for all lifecycle data. Therefore, tools with support of the software development lifecycle should focus on the mechanisms, embedded in the configuration management process, in order to be able to interact closely (be integrated). Such a close relationship (integration) through the configuration management process can significantly help with the development process, provide a predictable (and positive, if the tool was chosen correctly) result of aviation software development and help with preparing to the certification. It is important to note, that the purchase of the software tools and instruments does not ensure success in passing the certification – methodological support is also needed.

The task to select software tools for development lifecycle support is not easy, because it is rather difficult to determine in advance whether all requirements of chosen for this project lifecycle process will be supported by software tool, system or a set of tools. Analysis of configuration management process and selecting criteria from it to tools allows to define the boundaries of necessary for the project systems and tools. Analysis gives as result formulated requirements to the tool, which can be applied for choosing and buying suitable tool or in case of independent development such instrumental environment. In case of buying these requirements and criteria will help to choose exactly that product whose functions are necessary and sufficient for development goals without spending a lot of money for buying disparate software tools of different manufacturers, which will complicate the solution as a whole.

These conclusions are confirmed by the above analysis of one of the tools. Using of the set of tools extending the functional brings the environment closer to the reference state of configuration management process. In addition, there are difficulties: often the cost of licensing significantly increases (you have to buy additional tools), the time for installation, integration and implementation of the process increases, number of tools used in the project is growing and requires management efforts. As a result, the total complexity of development increases.

References

[1]. Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178B), 1992.

[2]. Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011.

[3]. Design Assurance Guidance for Airborne Electronic Hardware (RTCA DO-254), 2000.

[4]. Software Tool Qualification Considerations (RTCA DO-330), 2011.

[5]. Aerospace recommended practice. Guidelines for development civil aircraft and systems (SAE ARP 4754A), 2010

[6]. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (SAE ARP 4761), 1996

[7]. The Order of the Ministry of Industry and Trade of the Russian Federation of March 31, 2015 № 663 “About the approval of the industry plan of actions for import substitution in branch of civil aircraft industry of the Russian Federation” (with changes and additions)

[8]. Gorelits N.K., Peskov E.V., Requirements management as efficiency measure for software development in aviation industry. Sbornik trudov VIII Mezhdunarodnoy konferentsii “IT-STANDART 2017” [Proc. of VIII International conference “IT-Standard 2017”], Moscow, 2017, pp.105-113, ISBN 978-5-98597-346-4 (in Russian)

[9]. Qualification requirements part 178C, IAC, 2014 (in Russian)

[10]. M.A.Saburov, Yu.A.Solodelov, N.K.Gorelits. Development of the certifiable avionics software by the example of JetOS real time operation system. Navigatsiya, navedenie i upravlenie letatel’nymi apparatami: tezisy dokladov Tret’ey Vserossiyskoy nauchno-tekhnicheskoy konferencii [Proceedings of Third All-Rus. Scient.-Techical Konf. “Navigation, guidance and control aircraft”], Moscow, 2017, pp.241-243, ISBN: 978-5-93728-133-3 (in Russian)

[11]. System engineering — System life cycle processes (ISO/IEC/IEEE 15288:2015), 2015

[12]. Koverninsky I.V., Kan A.V., Volkov V.B., Popov Yu.S., Gorelits N.K. Practical experience of software and system engineering approaches in requirements management for software development in aviation industry. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp.173-179. DOI: 10.15514/ISPRAS-2016-28(2)-11

Критерии, предъявляемые к программному обеспечению для разработки сложных сертифицируемых систем, критичных по безопасности

Н.К. Горелиц < nkgorelits@2100.gosniias.ru>
А.С. Гукова < asgukova@2100.gosniias.ru>
Е.В. Песков <evpeskov@2100.gosniias.ru>
Государственный научно-исследовательский институт авиационных систем
Россия, 125319, г. Москва, ул. Викторенко, 7

Аннотация. На сегодняшний день в авиационной отрасли существует актуальная проблема – как инструментально поддержать и обеспечить сертифицируемость разработки критичных по безопасности сложных систем в соответствии с международными и отечественными отраслевыми стандартами, нормативными документами, такими как КТ-178С, КТ-254, Р-4754, Р 4761 и др. В статье рассматривается процесс управления конфигурацией при разработке по КТ-178С как основной источник критериев для осуществления выбора инструментального средства поддержки разработки. Выделенные критерии могут быть применены к инструменту поддержки всего жизненного цикла разработки авиационного ПО в соответствии с КТ-178С, а также к инструментам, отвечающим за поддержку отдельных процессов жизненного цикла. Мероприятия процесса управления конфигурацией обеспечивают работу с данными жизненного цикла, их хранение, целостность, безопасность, управляемость, информационную поддержку обмена данными между остальными процессами жизненного цикла, ведение истории изменений и т.п. Соблюдение принципов процесса управления конфигурацией позволяет осуществлять контроль разработки, обеспечить требуемые качество и надежность продукта, его сертифицируемость и необходимый уровень доверия к безопасности, снизить финансовые и временные затраты на разработку. В качестве примера использования критериев приведен анализ одного из распространенных в отрасли инструментов разработки и управления требованиями на соответствие указанным критериям.

Ключевые слова: КТ-178С; DO-178C; разработка ПО; анализ ПО; выбор ПО; сертифицируемые системы; сложные системы; разработка сложных систем; авионика; КБО; процессы ЖЦ; жизненный цикл; управление конфигурацией; системная инженерия.

DOI: 10.15514/ISPRAS-2018-30(4)-4

Для цитирования: Горелиц Н.К., Гукова А.С., Песков Е.В. Критерии, предъявляемые к программному обеспечению для разработки сложных сертифицируемых систем, критичных по безопасности. Труды ИСП РАН, том 30, вып. 4, 2018 г., стр. 63-78 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(4)-4

Список литературы

[1]. Квалификационные требования часть 178В, 2002 – АР МАК.

[2]. Квалификационные требования часть 178С, 2014 – АР МАК.

[3]. Руководство по гарантии конструирования бортовой электронной аппаратуры КТ-254, 2011 – АР МАК.

[4]. Software Tool Qualification Considerations (RTCA DO-330), 2011.

[5]. Руководство Р4754 по процессам сертификации высокоинтегрированных сложных бортовых систем воздушных судов гражданской авиации. АР МАК, 2010

[6]. Руководство 4761 по методам оценки безопасности систем и бортового оборудования воздушных судов гражданской авиации, 2010

[7]. Приказ Министерства промышленности и торговли РФ от 31 марта 2015 г. N 663 "Об утверждении отраслевого плана мероприятий по импортозамещению в отрасли гражданского авиастроения Российской Федерации" (с изменениями и дополнениями)

[8]. Горелиц Н.К., Песков Е.В., “Управление требованиями как критерий эффективности при разработке программного обеспечения в авиационной отрасли”, Сборник трудов VIII Международной конференции “ИТ-Стандарт 2017”. Москва, 2017, стр. 105-113, ISBN 978-5-98597-346-4

[9]. Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011

[10]. Сабуров М.А., Солоделов Ю.А., Горелиц Н.К. Разработка сертифицируемого бортового программного обеспечения на примере операционной системы реального времени JetOS. Навигация, наведение и управление летательными аппаратами: тезисы докл. Третьей Всерос. научно-технической конф. (Москва – Раменское 21-22 сент. 2017 г.), 2017, стр. 241-243, ISBN: 978-5-93728-133-3.

[11]. ГОСТ Р 57193 Системная и программная инженерия. Процессы жизненного цикла систем, 2016

[12]. Koverninsky I.V., Kan A.V., Volkov V.B., Popov Yu.S., Gorelits N.K. Practical experience of software and system engineering approaches in requirements management for software development in aviation industry. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp.173-179. DOI: 10.15514/ISPRAS-2016-28(2)-11