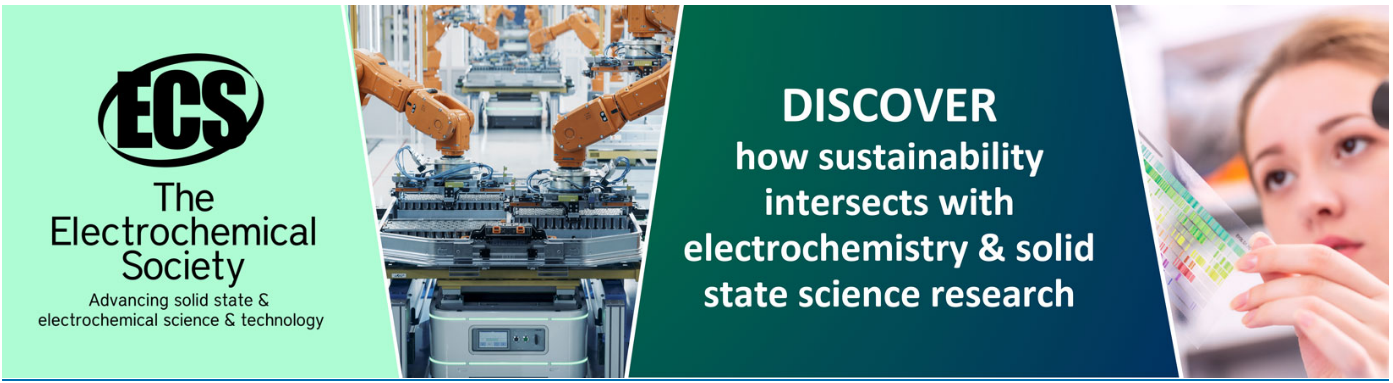
Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Research on Means of Compliance for Airborne COTS Software

To cite this article: Hui Hua *et al* 2021 *J. Phys.: Conf. Ser.* **1827** 012091


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER
how sustainability intersects with electrochemistry & solid state science research

Research on Means of Compliance for Airborne COTS Software

Hui Hua[1*], Jialu Luo[2], Yi Chu[3] and Bing Xu[4]

[1*]Shanghai Aircraft Design and Research Institute, Shanghai, 201206, China

[2]Shanghai Aircraft Design and Research Institute, Shanghai, 201206, China

[3]Shanghai Aircraft Design and Research Institute, Shanghai, 201206, China

[4]Shanghai Aircraft Design and Research Institute, Shanghai, 201206, China

[*]Corresponding author's e-mail: huahui@comac.cc

Abstract. The cost of airborne software development is very high, and the functions implemented by airborne software are becoming more and more complex. Using the COTS software has become a trend in case of saving cost. The COTS software brings conveniency, meanwhile raise problems when showing compliance to airworthiness requirements. This paper researches the design assurance method of COTS software, summaries the certification risks and propose a feasibility analysis of using COTS software, thus providing reference for airborne systems designs.

1. Introduction

With the development of the aviation technologies, the system functions realized by airborne software are getting more and more complex, the number and scale of airborne software are growing very fast. Meanwhile, the airworthiness requirements of airborne software are becoming more rigorous. COTS(Commercial-off-the-shelf) refers to commercial shelf products, which are software or hardware products with interfaces defined by open standards and can be purchased [1]. Compared with developing a new software, the use of COTS software can shorten the development cycle and reduce the development cost, and COTS software has an application basis and can provide a certain degree of confidence, so the use of COTS software in the aviation field is gradually becoming common.

The RTCA/DO-178B released in 1992 provides an acceptable design assurance method for COTS software. However, due to the inherent limitations of COTS software itself, there are only a handful of COTS software that can fully meet DO-178B requirements. In July 2017, the Federal Aviation Administration (FAA) and the European Aviation Safety Agency (EASA) jointly issued an advisory circular [2] recognized RTCA/DO-178C as a software design assurance method, which is based on DO-178B and improves the requirements of COTS software, and sets greater restrictions on the use of COTS software, making it more difficult for the industry to implement COTS software.

This article summarizes the COTS software design assurance methods accepted by DO-178B/C, correspondingly summarizes the difficulties of COTS software design assurance, and on this basis, proposes a feasibility analysis of using COTS software, which provides theoretical support for the use of COTS software for airborne system development.

2. Overview of COTS software

Previously developed software (PDS) refers to software that has been developed and put into use [3,7]. PDS covers a wide range. In civil aviation, PDS refers to software that has been certified with a certain aircraft model, an engine model, or a TSO equipment, or COTS software.

COTS software can be recognized as PDS, if it has been used in a civil aircraft, an engine or a TSO equipment that has been certified, while in some cases it still needs to be agreed with the certification authorization. COTS specifically refer to commercial shelf software, that is, software sold by vendors through public catalogs [3]. The user selects the appropriate COTS software from the market and integrates it into the aircraft system as part of the aircraft design. The design of COTS software is determined according to market needs and normally will not be customized or improved according to customer needs [4]. The design, production, and configuration control of COTS software is probably controlled by vendors. The data that users can obtain is very limited, usually including product manuals, but not the detailed design data and production data of COTS software.

The airworthiness requirements for COTS software are rigorous, so COTS software is generally less used in safety-critical areas, and is mainly used in aircraft systems with lower safety levels. Board support packages (BSP) and real-time operating systems (RTOS) are more often used [5] as COTS. The RTOS that has been successfully used in aircraft systems includes LynxOS-178 developed by Lynux Works, which has passed the DO-178B A-level software certification, Tornado developed by Wind River has been applied on aircraft, Wind River VxWorks developed by Verocel is also working to pass DO-178C level A software certification. Other RTOSs certified include VRTX, PSOS and OSE, etc.

The selection of COTS software generally considers its functions, performance, interfaces, usage standards, quality indicators, etc. Non-index factors such as market share, vendor's performance also needs to be considered.

3. COTS design assurance methods

3.1. Reverse Engineering

The certified COTS software can be certified as a PDS with a new aircraft if it meets the requirements of the design, environment of usage and development assurance level of the new aircraft. For the COTS software that lacks sufficient life cycle data and cannot meet the DO-178C design assurance requirements, the development baseline needs to be improved, mainly includes the following aspects [3]:

1) The system safety analysis process should define the possible failure conditions of the COTS software and assign the software development assurance level.

2) COTS software should meet the DO-178C objectives of the corresponding software development assurance level.

3) COTS software life cycle data should demonstrate that the verification process objectives and independence requirements of the corresponding level of the software are met.

COTS software is designed by vendors based on market demand. Its target customers include aviation fields and electronics, medical, nuclear power, military industry, etc., As the aviation field is not the main market of COTS software, the COTS software rarely uses the DO-178 guidance for software development, as a consequence, its development process and life cycle data are difficult to show compliance with airworthiness requirements. Based on the actual situation, DO-178B/C clearly proposes the use of reverse engineering as a means to improve the development baseline, regenerating missing or unsatisfactory software life cycle data, and supplementing missing life cycle activities.

Reverse engineering refers to extracting higher-level data from lower-level data of software, such as generating software requirements based on software source code. Through reverse engineering, software lifecycle data that does not exist, is inappropriate, or is not available can be generated. DO-178C requires 22 types of life cycle data, so reverse engineering needs to generate at least 22 types of data, including software plans, design data, verification data, configuration management data and quality assurance data. Although the method of reverse engineering is acceptable, it is not recommended, and supplementing life cycle data only is not enough, developing process should be conducted. It is also necessary to prove that the purpose of software design assurance is achieved.

3.2. Product Service History

If the COTS software does not use the DO-178C main body design assurance method, the service history can be used as an alternative method to ensure equivalent safety. Product service history refers to a period of continuous time, the software is running in a known environment, and the faults generated have been recorded [6]. The four attributes of the product service history should be evaluated: service duration, product quality and accumulated data during this period, the problem discovery and reporting mechanism of the service system, and the configuration control of software changes.

COTS software that uses product and service history may generally has several characteristics: the software has low compliance with the DO-178B/C design assurance method, the software is very complex, and the vendor's normally unwilling to provide technical or data support, thus there is a big difficulty of reverse engineering. The COTS software is used for a long time and has a wide range of applications, the market has fully verified the quality of the software. And the software is rarely used in safety-critical systems.

The requirements for product service history can be summarized as follows:

1) Software configuration management: the software and data configuration are controlled during the declared service time.

2) Effectiveness of problem reporting activities: software service problems are reported, and the reports are recorded and accessible, safety-related reports should be analyzed and closed, software problems should be distinguished from hardware or system problems.

3) Stability and maturity of the software: software changes must be under control of the configuration, and the stability and maturity of the software can be judged by evaluating the configuration changes.

4) The relevance of the product service history application environment: whether the environment and configuration of the software application are the same as the service history.

5) Actual error rate: Whether the declared error rate is the same as the actual value.

6) The impact of software changes: The verification activities that need to be supplemented or the validity of the service history can be determined by evaluating software changes.

DO-178C integrates the above requirements into three aspects: service history relevance, service history adequacy, and collection, reporting, and analysis of problems found in service history. DO-178C gives a clearer description of the calculation method of service history, the evaluation dimension of service history adequacy, and service history the data to be collected and the evaluation of the problem reporting mechanism, and add the following more stringent requirements:

1) If the software is changed, it is necessary to evaluate whether the service history before the change is still valid. All changes to the software should be recorded and evaluated. In addition, attention should be paid to the completeness of the software problem report resolution process, the integrity of the problem record, and the effectiveness of the software change evaluation mechanism.

2) It is necessary to fully evaluate the use of the software during the service history, including the operating mode of the software, the range of input and output values, and input combinations. The operating system needs to evaluate the execution sequence of events driving the software operation [6]. Software functions that have not been executed cannot be counted as part of the service history.

3) If the hardware environment of the software changes, it is also necessary to evaluate the validity of the service history before the change. The hardware environment here includes the processor and storage, channels and ports used. The hardware environment assessment referred to by 178C combines the characteristics of the hardware and is more in-depth than the requirements of DO-178B.

4) Different applications may use different COTS software functions. Even if no problems are found with the unused functions, it does not mean that there are no errors. The inactive code of the COTS software needs to be identified. If it remains inactive during the service history, it must be suppressed in the new application environment, otherwise additional verification activities need to be

supplemented.

4. COTS Software Certification Risks

4.1. Reverse Engineering

The cost and difficulty of reverse engineering of COTS software is no less than that of developing a new airborne software. As mentioned in section 2.1, COTS software seriously lacks compliance evidence. Without the participation of original vendors and industry experts, reverse engineering based on source code or even object code is extremely difficult, especially for COTS software with high complexity, and its code of low quality and lack of sufficient comments. The application of reverse engineering has the following difficulties:

1) Lack of good planning. Although reverse engineering can supplement and generate a complete set of plans, it lacks operability, and it is difficult to show that software development activities are proceeding as planned.

2) It is difficult to meet the objectives of design assurance, mainly related to the planning process, software standards, design data traceability, and quality assurance. In addition to supplementing all data, reverse engineering must prove to achieve the purpose of design assurance and guarantee the quality of the product. In fact, it is difficult to guarantee good quality of software design data generated in reverse, and it is difficult to show compliance with process-related goals. For example, to extract low-level software requirements from source code, the granularity of requirements description is difficult to grasp, and it is easy to write pseudo-code. If the quality of the requirements cannot be guaranteed, the results of the software structure coverage analysis will not be ideal. Another example is that the initial software review data cannot be obtained, and the software review activities carried out in reverse engineering can easily become a formality, and it is difficult to achieve the true verification purpose.

3) It is difficult to meet the requirements of aircraft systems, especially safety requirements. The design of COTS software is difficult to correspond to the design of aircraft systems. The software requirements generated by reverse engineering cannot normally be traced back to the system requirements, especially when a third party is used for reverse engineering, because the third party may not understand how the COTS software will be applied to the aircraft systems. Therefore, OEMs often have to make a series of assumptions about the performance and reliability of COTS software, but there is insufficient evidence to show that these assumptions can meet the requirements of aircraft systems.

4) Difficulties in airworthiness contact. Reverse engineering is often completed before the airworthiness authority involves in the software review, so the airworthiness authority cannot review the COTS software reverse engineering activities at early stage. If a third party is used for reverse engineering, the third party is likely to lack an understanding of airworthiness requirements. Both of these points will cause hidden risks for subsequent software reviews.

4.2. Service History

Adopting service history as an alternative to COTS software also faces many problems, which can be summarized as follows:

1) The collection channel of service history is narrow. The service history applies to previously developed software used in certified aircraft and will be applied to new aircraft. Although DO-178C does not clearly state, it places high requirements on the similarity of software application environments. Therefore, the confidence of service history collected outside the aviation field is weak.

2) The validity of the service history is difficult to prove. The service history shall explain the model, system, application environment and usage mode of the COTS software application, and explain how to count the usage time. At present, a stable and effective feedback mechanism has not been formed between airlines, OEMs, equipment suppliers and COTS software vendors, making it very difficult to accurately collect service history that meets the above requirements and provide

strong proofs.

3) It is difficult to compare software usage and application environment. The OEM masters the design and application environment of the COTS software for their own models, while airlines and COTS vendors do not know the specific application of COTS software. Due to IP restrictions, there are great obstacles in collecting related service history outside of our OEM.

4) Problem discovery and reporting mechanism. Problem reports generally have problems such as unclear descriptions of software configuration, and obscure records in the discovery, monitoring and reporting process of failures, which make it difficult to reproduce software problems and are not sufficient as evidence of service history. Vendors are subjectively unwilling to expose product problems, making some problem reports difficult to obtain. In addition, most of the problem reports only record software problems, and the OEM needs to additionally evaluate its impact on the aircraft system.

5) Change and configuration control: COTS software vendors control the complete configuration of the software, and the update iteration of COTS software is relatively fast, causing certain difficulties to the configuration management activities of the OEM, and there is a problem of configuration accuracy. In addition, the configuration of COTS software is more flexible. COTS software with the same part number may also have different versions. It is recommended to control by software components.

6) Inactive codes: For aircraft systems using COTS software, inactive codes are unplanned functions and may affect the performance of the safety-critical functions of the system. Identifying functions that have never been used in the service history requires opening the design of the COTS software to a certain extent and analyzing the security impact. Without the help of the vendor, the identification work is difficult. Also, if you want to use an inactive code, it may invalidate the service history.

5. Feasibility analysis of using COTS software

Carrying out a full assessment before deciding to adopt COTS software can pass the airworthiness review more smoothly, which can really save development costs and shorten the development cycle. Combining the feasible design assurance methods or alternative methods of COTS software proposed in Chapter 2 of this article, and the COTS software certification risk proposed in Chapter 3, compared with COTS software that cannot be recognized as PDS, the reverse engineering has a higher possibility to succeed. But its cost is high, and there is a problem with the demand traceability of the system. The service history has a lower possibility to succeed, but it can really achieve the purpose of saving costs. This paper proposes the following feasibility analysis based on the characteristics of the two methods:

1) Software design assurance: Evaluate whether the COTS software comply with a certain design assurance method during development, and whether there are corresponding software development process requirements within the vendor. Even if it is not developed using the DO-178 series of guidelines, it can improve the confidence of the software product to a certain extent.

2) Software Certification: collect software certification information and applied aircraft models and systems, thus to reflect the confidence of the software on the side, and know the software usage mode and usage environment.

3) Requirements for COTS software functions. In system software development, try to implement more functions, especially key functions, through newly developed software, and restrict the use of COTS software functions.

4) Product changes: The frequency of changes can be judged through the history of software configuration, and try to choose COTS software that has fewer changes and does not use new technologies. Generally speaking, the longer the COTS software is used, the higher confidence level could be, but it also means that its development time is earlier, and lacks evidence of compliance with airworthiness requirements.

5) Code quality: If the COTS software code can be obtained, the code quality can be preliminarily evaluated, including the standardization of code writing, the completeness of comments, the call

between functions and the complexity of nesting, etc. Code of high quality improves maintainability and also provides the possibility of reverse engineering.

6)Supplementary verification: When service history is used as an alternative method, but its confidence is weak, other methods can be considered to provide supplementary confidence, such as supplementary verification, including review of existing compliance data, and supplementary testing of the software, namely laboratory test, on-board ground test and flight test.

7) COTS software development assurance level: The lower the COTS software development assurance level assigned to the COTS software by a system using COTS software, the higher the probability of passing the airworthiness review. The level D and level E COTS software is low-risk, and the level A, B, and C COTS software has more difficult objectives to meet, mainly include software standards, low-level demand development and code structure coverage analysis.

8) COTS software data package: The completeness and quality of the data package is an important criterion for considering whether the COTS software can be used. The data provided by the vendor is the most accurate. The complete data package can provide detailed design information of the software, reducing the workload of secondary development.

9) Vendor Support: If the vendor provides sufficient support, it will help users understand the COTS software to better perform system design and avoid design defects in the early stage. If the vendor's system is more mature, the after-sales service is also more complete, and it is more likely to provide continuous and stable services during the software certification process to reduce project risks.

Based on the above analysis, the selection criteria of compliance methods can refer to Table 1.

Table 1 the selection criteria of compliance methods

Reverse Engineering		Feasibility Analysis		Service History
Incomplete	<------	Design Assurance	------>	Complete
Few	<------	Certification	------>	Many
Many	<------	Functions	------>	Few
High	<------	Function criticality	------>	Low
Many	<------	Product Change	------>	Few
High	<------	Quality of Code	------>	Low
Yes	<------	Supplemental Verification	------>	No
High	<------	DAL	------>	Low
High	<------	Data Package integrity and Quality	------>	Low
High	<------	Vendor support	------>	Low

6. Conclusions

This paper studies the acceptable compliance methods of COTS software, analyzes the difficulties and limitations of using COTS software, and based on these limitations, proposes a feasibility analysis method for using COTS software. The use of COTS software will become a trend. It is recommended that users consider the reusability in other projects when choosing the compliance method of COTS software.

References

[1] Guo Qiiun. (2014) Analysis of Application and Certification for COTS in Civil Aircraft Software Design[J]. Journal of Anhui Jianzhu University., 22: 54-58.
[2] FAA. Airborne Software Development Assurance Using EUROCAE ED-12() and RTCA DO-178(), AC 20-115D, [2017-7-21].
[3] RTCA. Software Considerations in Airborne Systems and Equipment Certification, DO-178C.

Washington, D.C: RTCA Inc, 2011.
[4] FAA. Commercial off-the-shelf (COTS) Avionics Software Study. Washington, D.C. 20591: Office of Aviation Research, 2001.
[5] Liu Dong. (2005) COTS-Based Avionics Software Development. Avionics Technology., 36：25-30
[6] RTCA. Supporting Information for FO-178C and DO-278A, DO-248C. Washington DC: RTCA Inc, 2011.
[7] RTCA. Software Considerations in Airborne Systems and Equipment Certification, DO-178B. Washington, D.C: RTCA Inc, 1992.