NOTE: This position paper has been coordinated among the software specialists of certification authorities from the United States, Europe, and Canada. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.
Reverse Engineering in Certification Projects

1.0 Introduction

Some applicants, developers, and commercial-off-the-shelf (COTS) software vendors have proposed reverse engineering as an approach for satisfying DO-178B/ED-12B objectives. DO-178B/ED-12B defines “reverse engineering” as: The method of extracting software design information from the source code [1]. Section 12.1.4 of DO-178B/ED-12B addresses “Upgrading a Development Baseline,” which may be implemented using reverse engineering. Additionally, paragraphs 12.1.4.d and 12.1.4.f of DO-178B/ED-12B provide guidance particularly relevant to reverse engineering.

Paragraph 12.1.4.d states: Reverse engineering may be used to regenerate software life cycle data that is inadequate or missing in satisfying the objectives of this document. In addition to producing the software product, additional activities may need to be performed to satisfy the software verification process objectives [1].

Paragraph 12.1.4.f states: The applicant should specify the strategy for accomplishing compliance with this document in the Plan for Software Aspects of Certification [1].

This paper will attempt to answer some of the common questions about reverse engineering by exploring:

- What is reverse engineering?
- What motivates use of reverse engineering?
- What are certification concerns regarding reverse engineering?

This paper also summarizes the certification authorities’ position regarding reverse engineering.

Those who read or use this paper should consider the following notes:

- There are other approaches to addressing previously developed software, such as service history (see DO-178B/ED-12B, section 12.3.5, and DO-248B/ED-94B, section 4.5 [4]); however, this paper focuses only on reverse engineering. In some cases reverse engineering may be used in conjunction with other alternate methods, such as product service history.
- The purpose of this paper is not to encourage reverse engineering – reverse engineering should only be used in well-justified cases.
- Reverse engineering requires more than just producing the software life cycle data and may be incompatible with “building the quality into” the software product. That is, any reverse engineering approach proposed to certification authorities must demonstrate that it is addressing design assurance and not just creating software life cycle data.

NOTE: This position paper has been coordinated among the software specialists of certification authorities from the United States, Europe, and Canada. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.
2.0 What is Reverse Engineering?

As discussed above, DO-178B/ED-12B defines “reverse engineering” as: The method of extracting software design information from the source code [1].

Roger Pressman defines reverse engineering for software as the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code. Reverse engineering is a process of design recovery [2].

From a certification authority perspective, reverse engineering is an approach to generating software life cycle data that did not originally exist, cannot be found, is inadequate, or is not available in order to satisfy the applicable DO-178B/ED-12B objectives. However, it is not just the generation of the relevant software life cycle data, but a process of assuring that the data is correct, the software functionality is understood and documented, and the software functions (performs) as intended and required by the system. It involves recovery of requirements and design, as well as conducting the relevant verification activities to the appropriate level to ensure the integrity of the software, to ensure all software life cycle data is available and correct, and that an appropriate level of design assurance is achieved.

3.0 What Motivates Use of Reverse Engineering?

A number of applicants, developers, and software vendors (including commercial off-the-shelf (COTS) software vendors) desire to implement reverse engineering in order to use previously developed software (PDS) in their airborne applications. This PDS is often developed outside of the aviation environment and its applicable guidance. In some cases, only the source code exists. However, in some cases other software life cycle data may exist but is incomplete or inadequate to satisfy the DO-178B/ED-12B guidance. If properly applied, a reverse engineering approach may allow an applicant to gain the necessary design assurance for airborne software, complete any missing and/or improve inadequate software life cycle data, obtain an appropriate level of design assurance for its intended use, and establish a baseline for future use and products.

Software that is typically suitable for reverse engineering:

- has a mature and stable version that has been used in a number of applications;
- has shown itself to be of high integrity (e.g., it has minimal problem reports over a long period of time, a robust error tracking system, …); and
- may have been developed to other standards (e.g., military standards or ISO standards) or no standards or guidance and doesn’t satisfy the DO-178B/ED-12B objectives and other airborne software guidance.

Some of the potential advantages that motivate applicants to pursue reverse engineering are:

- It may be more cost and schedule effective to use already existing software than to develop new software.
- It may be an investment in the future projects of the company (i.e., once the reverse engineering project is completed, the data package can be reused and built upon).
4.0 What Are The Certification Concerns Regarding Reverse Engineering?

Throughout the past few years, certification authorities have been presented with several reverse engineering projects. The Real-Time Operating System (RTOS), in particular, has been a software component that many applicants, developers and vendors desire to reverse engineer (see draft AC 20-RSC [3] for further information on software components). Applicants often desire to use existing RTOSs that were not developed to DO-178B/ED-12B guidance and produce a version of their product that “complies with DO-178B/ED-12B”. Using an existing RTOS can utilize a “proven” component and reduce the need for an airborne system developer to hire many operating system experts and develop their own new RTOS.

Several reverse engineering projects have been reviewed by certification authorities and have raised some concerns. This section lists and briefly explains each concern. Applicants should be proactive in addressing these concerns in their projects.

4.1 Lack of A Well-Planned Process.

A key to a successful reverse engineering approach is a well-defined reverse engineering process. Performing the assurance activities and generating the necessary and appropriate software life cycle data doesn’t just happen – it must be planned, like any other software development effort. A common downfall of reverse engineering projects seen to date has been poor or non-existent planning data.

Reverse engineering may be considered a life cycle model, going from code to design to requirements. The processes and activities in that life cycle and the transition criteria between those processes and activities should be well-defined in the software plans (i.e., the Plan for Software Aspects of Certification, Software Development Plan, Software Configuration Management Plan, Software Quality Assurance Plan, and Software Verification Plan), software standards (Requirements, Design and Coding Standards), and verification procedures. The plans and standards should clearly define how the DO-178B/ED-12B objectives will be satisfied through the reverse engineering effort.

Before beginning a reverse engineering project, the upgrading/development organization should propose their approach to the certification authorities and get agreement to reduce their project risk. The plans, standards, and procedures should be followed.

4.2 Poor Justification for Reverse Engineering

Some applicants, developers, and vendors have proposed reverse engineering projects without adequate justification for how safety objectives will be met. Certification authorities have observed some of the following problems:

- Use of reverse engineering as a recovery plan for an out-of-control or poorly planned project.
- Use of reverse engineering to save money without a technical basis.
- Failure to satisfy many DO-178B/ED-12B objectives and aviation software policy in the reverse engineering effort.

NOTE: This position paper has been coordinated among the software specialists of certification authorities from the United States, Europe, and Canada. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.
• Using reverse engineering just to satisfy a list of data items, rather than to ensure the design assurance and quality of the product and its appropriateness for an airborne application.

Reverse engineering should be used cautiously and only in well-justified cases (i.e., for a project that has been used in a number of applications and has shown itself to be of high integrity). The use of reverse engineering in new software development is strongly discouraged by the certification authorities (i.e., it shouldn’t be used to compensate for a poor development approach).

4.3 Lack of Access to Experts and Original Developers

Developing the design, requirements, and test cases for a complex software component, such as an operating system, can be nearly impossible without some access to the original developers. Without expertise in the domain being reverse engineered, the ability to accurately determine what the software was meant to do is questionable and can be difficult to determine.

The most successful reverse engineering efforts have been those where developers contacted the original developers in order to gain a thorough understanding of the software functionality, particularly in difficult or ambiguous areas.

Every effort should be made to gain access to the original developers or to hire people with expertise in the specific domain area (e.g., operating system experts for an RTOS upgrading project).

4.4 Complex and Poorly Documented Source Code

Many reverse engineering efforts start with source code that is complex and poorly documented. The code may contain numerous pointers and complex data structures. The code may also not contain commentary statements, which can make it difficult to understand.

Applicants should consider the condition of the code before starting a reverse engineering effort. Overly complex or poorly documented code may make satisfying the DO-178B/ED-12B objectives and aviation software policy difficult or impossible. Poorly documented or overly complex code will make it difficult for the reverse engineering team to assess what the code was intended to do and to develop requirements and design data, and determine how to verify the software. The verification effort will also be difficult to complete with poorly documented or overly complex code.

A thorough understanding of the code is essential to successful reverse engineering. Poorly documented or overly complex code is not a good candidate for reverse engineering.

4.5 Abstraction and Traceability Difficulties

Pressman writes: Reverse engineering can extract design information from source code, but the abstraction level, the completeness of the documentation, the degree to which tools and
human analyst work together, and the directionality of the process are highly variable ...

Ideally, the abstraction level should be as high as possible. That is, the reverse engineering process should be capable of deriving procedure design representations (a low level of abstraction); program and data structure information (a somewhat higher level of abstraction); data and control flow models (a relatively high level of abstraction); and entity-relationship models (a high level of abstraction).

Pressman goes on to say that the completeness of a reverse engineering process refers to the level of detail that is provided at an abstraction level. In most cases, the completeness decreases as the abstraction level increases.

The problem is a balance between completeness and abstraction level. In DO-178B/ED-12B terminology, formulating the higher-level abstraction of low-level requirements and derived requirements to high-level requirements can be extremely difficult.

In reviewing reverse engineering projects, certification authorities frequently find the following abstraction and traceability problems:

- Airborne system requirements cannot be correlated to the reverse engineered product’s high-level software requirements.
- High-level requirements are written like low-level requirements (i.e., the abstraction level is too low) (this makes testing of both high-level and low-level requirements difficult).
- Lack of traceability from high-level requirements to low-level requirements to code and test cases and procedures (i.e., the forward traceability (and often the backward traceability) is not established).
- Performing a combined bottom-up and top-down approach for requirements generation rarely works (i.e., the requirements don’t meet in the middle).
- Establishing traceability and compliance to system-level and safety requirements is difficult in a reverse engineering effort. Many times the organization upgrading the software is not the applicant, system developer, or software integrator. Instead, a third or fourth party company is used, and they may not have an understanding of how their product will be used in the airborne system and may not understand what compliance with DO-178B/ED-12B and other airborne software guidance means.
- Upgraders attempt to put off the traceability effort until the end (which can result in code that doesn’t trace to requirements or requirements that aren’t fully implemented).
- Upgraders attempt to merge requirements and testing into a single level for very complex software (i.e., they try to omit either the high-level requirements or the low-level requirements) (this makes satisfying DO-178B/ED-12B objectives difficult).
- Inaccurate traceability can make it difficult to determine whether code not able to be traced to is dead or deactivated.
- Traceability to and compliance with system-level requirements is difficult or missing or inadequate.
- Derived requirements should be very sparse in a reverse engineering effort, when the effort starts with the source code. When derived requirements do exist, they must be handled very carefully and evaluated by the system safety experts.
• Many times vendors may have no idea of what system their product will be used in and do not know what impact any features or problems with their product may have on the safety of the system in which it will be used.
• Unwanted functionality and unused features may be identified in the code and must be addressed (e.g., a subset of the entire reverse engineered product may need to be generated to address issues found during the effort).

4.6 Interface and Integration Problems

Inadequate or missing interface and integration data has been observed in a number of projects. This leads to misunderstandings and misinterpretations by the users/applicants. Some of the specific problems observed by certification authorities are:

• Missing, incomplete, or out-of-date interface data,
• Missing, incomplete, or out-of-date user’s guides, and
• Missing, incomplete, or out-of-date integration data (e.g., porting guidelines).

Additionally, when non-aviation manufacturers attempt to reverse engineer a software component, they often lack the necessary understanding of DO-178B/ED-12B objectives and the certification process necessary to effectively communicate with the applicant and certification authorities.

4.7 Certification Liaison Process Problems

Many reverse engineering efforts do not perform the certification liaison process well. The following problems often exist:

• Designees or certification authorities are not informed or involved early in the reverse engineering process.
• Applicants often hire sub-contractors or suppliers to reverse engineer a component. However, the sub-contractors do not have a communication avenue with the certification authorities or designees, nor an understanding of DO-178B/ED-12B and other aviation software guidance.
• There is often a lack of communication and understanding between the multiple stakeholders (e.g., certification authorities, applicant(s), integrator(s), and vendor(s)).
5.0 Certification Authorities Position

The position of the certification authorities regarding reverse engineering can be summarized as follows:

- Reverse engineering should be used cautiously and only in well-justified cases (i.e., for a project that has been used in a number of applications and has shown itself to be of high integrity). The use of reverse engineering in new software development is strongly discouraged by the certification authorities. Additionally, a justification of cost savings without technical and safety merit is not acceptable to the certification authorities.

- An applicant should make a case for why reverse engineering is feasible and how it will satisfy the objectives of DO-178B/ED-12B, other airborne software guidance, and the overall safety objectives of the regulations, for their particular project. The following objectives may be particularly difficult to satisfy and require special attention: Objectives 1 through 7 of Table A-1; Objectives 5 & 6 of Table A-3; Objectives 5, 6 & 12 of Table A-4; Objectives 4 & 5 of Table A-5; Objectives 1 & 2 of Table A-9; and Objective 2 of Table A-10.

- Reverse engineering processes should be well-defined and well-planned. The approach should be planned into life cycle processes and activities, with transition criteria, and should be documented in the plans and standards. The plans and standards should be followed.

- Reverse engineering projects should be coordinated with the appropriate certification authorities. Since there are a number of concerns regarding reverse engineering, any projects using it should be coordinated with the certification authorities as early as possible. Certification authorities may perform software reviews to ensure that the developer followed their plans and standards, the software life cycle data produced is complete and correct, and that all applicable DO-178B/ED-12B objectives are satisfied.

- The concerns documented in section 4 of this paper should be addressed by the applicant and/or integrator, as well as any other project-specific concerns identified by certification authorities or their designees.

6.0 References


