“Reverse Engineering”
Software and Digital Systems

DISCLAIMER
This draft document is being made available as a “Limited Release” document by the FAA Software and Digital Systems (SDS) Program and does not constitute FAA policy or guidance. This document is being distributed by permission by the Contracting Officer’s Representative (COR). The research information in this document represents only the viewpoint of its subject matter expert authors.

The FAA is concerned that its research is not released to the public before full editorial review is completed. However, a Limited Release distribution does allow exchange of research knowledge in a way that will benefit the parties receiving the documentation and, at the same time, not damage perceptions about the quality of FAA research.
NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the objective of this report. The findings and conclusions in this report are those of the author(s) and do not necessarily represent the views of the funding agency. This document does not constitute FAA certification policy. Consult your local FAA aircraft certification office as to its use.
Reverse engineering (RE) is a class of development processes that start with detailed representations of software for a system, or hardware description for a device, and apply various techniques to produce more generalized, less detailed representations. The goal is to have more abstract representations that can be used to understand and reason about the structure and the intent of the more detailed representations. RE has been used in many industries, including aircraft applications, for mechanical, hardware, and software components. The scope of this report covers software and electronic hardware device applications of RE for airborne systems and equipment.

The Federal Aviation Administration (FAA) sponsored this research project to provide a clear understanding of what should be considered RE for airborne software and airborne electronic hardware (AEH) devices, and under what conditions it could be deployed or restricted in the aircraft certification environment. The report is designed to provide an overview of the aviation industry’s views of RE, potential issues of employing RE for safety critical airborne systems, recommendations for when application of RE is acceptable, and some associated criteria for successful implementation.

This report provides the results of research into RE. The intended audience includes practitioners who develop compliance evidence, as well as evaluators for airborne software and AEH to be approved under RTCA documents DO-178B and DO-254, respectively. The report proposes a framework for RE of software and electronic hardware for airborne systems and equipment.

The report also validates that framework by presenting two case studies. The software case study chosen is a subset of an Ada runtime library. This library was chosen because it is distributed under the GNU General Public License (GPL) and because it has previously been approved to DO-178B Level A as part of a specific aircraft project. The AEH case study was based on the certification of a system with two programmable logic devices. The intent of these case studies was to validate the framework against two examples which were developed using RE, and which were approved.
ACKNOWLEDGEMENTS

We would like to thank Kelly Hayhurst from NASA Langley Research Center for her valuable contribution to the survey, and Charles Kilgore from FAA, for his technical support and guidance throughout the project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTIVE SUMMARY</td>
<td>vi</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Purpose and scope</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 Why is this report needed?</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Intent of the report</td>
<td>3</td>
</tr>
<tr>
<td>1.3 User guide for the report</td>
<td>4</td>
</tr>
<tr>
<td>2. SURVEY AND INDUSTRY EXPERIENCE</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Industry experience</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Industry perspective from survey</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1 Software survey results</td>
<td>5</td>
</tr>
<tr>
<td>2.2.2 Complex Electronic Hardware (CEH) survey results</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Terminology</td>
<td>7</td>
</tr>
<tr>
<td>2.3.1 Reverse engineering (RE)</td>
<td>7</td>
</tr>
<tr>
<td>2.3.2 Configuration management of reverse engineered artifacts</td>
<td>8</td>
</tr>
<tr>
<td>2.3.3 Certifier</td>
<td>8</td>
</tr>
<tr>
<td>2.3.4 Forward engineering</td>
<td>8</td>
</tr>
<tr>
<td>3. FRAMEWORK FOR PERFORMING RE</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Software development processes and sequence dependencies</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Compatibility of regulatory guidance with RE processes</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Roles</td>
<td>12</td>
</tr>
<tr>
<td>3.4 Processes</td>
<td>13</td>
</tr>
<tr>
<td>3.4.1 An example of source code to LLR development</td>
<td>13</td>
</tr>
<tr>
<td>3.4.2 Generic RE processes</td>
<td>14</td>
</tr>
<tr>
<td>3.4.3 Inputs</td>
<td>15</td>
</tr>
<tr>
<td>3.4.4 Outputs</td>
<td>16</td>
</tr>
<tr>
<td>3.4.5 Entry criteria</td>
<td>16</td>
</tr>
<tr>
<td>3.4.6 Exit criteria</td>
<td>17</td>
</tr>
<tr>
<td>3.4.7 Process description</td>
<td>18</td>
</tr>
<tr>
<td>3.5 RE aspects of software verification</td>
<td>18</td>
</tr>
<tr>
<td>3.6 Generic RE SME verification process</td>
<td>20</td>
</tr>
</tbody>
</table>
3.6.1 Inputs  20
3.6.2 Outputs  21
3.6.3 Entry criteria  21
3.6.4 Exit criteria  21
3.6.5 Process description  22

3.7 Acceptance criteria  22

4. VALIDATION OF FRAMEWORK  23

5. RECOMMENDATIONS  23

6. CONCLUSIONS  25

7. REFERENCES  25

APPENDICES

A - CASE STUDY: ANALYSIS OF PROJECTS COMPLETED AT VEROCEL
B - SURVEY RESULTS
C - ANALYSIS OF ISSUES AND THEIR POTENTIAL MITIGATIONS
D - VALIDATION OF FRAMEWORK FOR SOFTWARE
E - VALIDATION OF FRAMEWORK FOR AEH
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1. Example processes for development and review of LLRs using RE</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2. Generic processes for development and review of different abstraction layers</td>
<td>15</td>
</tr>
</tbody>
</table>
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D</td>
<td>Analog/Digital</td>
</tr>
<tr>
<td>AEH</td>
<td>Airborne Electronic Hardware (includes CEH and SEH)</td>
</tr>
<tr>
<td>ARTE</td>
<td>Ada Runtime Environment</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuit</td>
</tr>
<tr>
<td>CAST</td>
<td>Certification Authorities Software Team</td>
</tr>
<tr>
<td>CD</td>
<td>Compact Disc</td>
</tr>
<tr>
<td>CEH</td>
<td>Complex Electronic Hardware</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off-The-Shelf</td>
</tr>
<tr>
<td>DAL</td>
<td>Design Assurance Level</td>
</tr>
<tr>
<td>DER</td>
<td>Designated Engineering Representative</td>
</tr>
<tr>
<td>DOD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>DVD-ROM</td>
<td>Digital Versatile Disc – Read-Only Memory</td>
</tr>
<tr>
<td>ELOC</td>
<td>Effective Lines of Code</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FTR</td>
<td>Functional Test Result</td>
</tr>
<tr>
<td>GCC</td>
<td>GNU Compiler Collection</td>
</tr>
<tr>
<td>GNAT</td>
<td>GNU NYU Ada 9X Translator</td>
</tr>
<tr>
<td>GNU</td>
<td>A recursive acronym for &quot;GNU's Not Unix!&quot;</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>HAS</td>
<td>Hardware Accomplishment Summary</td>
</tr>
<tr>
<td>HDP</td>
<td>Hardware Development Plan</td>
</tr>
<tr>
<td>HIS</td>
<td>High Integrity Software</td>
</tr>
<tr>
<td>HLR</td>
<td>High-Level Requirement</td>
</tr>
<tr>
<td>HRS</td>
<td>Hardware Requirements Specification</td>
</tr>
<tr>
<td>HWCI-1</td>
<td>Hardware Configuration Item #1</td>
</tr>
<tr>
<td>ICD</td>
<td>Interface Control Drawing</td>
</tr>
<tr>
<td>ICU</td>
<td>Interface Control Unit</td>
</tr>
<tr>
<td>IDEF0</td>
<td>Integration Definition for Function Modeling</td>
</tr>
<tr>
<td>IMA</td>
<td>Integrated Modular Avionics</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>KLOC</td>
<td>Kilo (Thousand) Lines of Code</td>
</tr>
<tr>
<td>LAL</td>
<td>Less Abstract Layer</td>
</tr>
<tr>
<td>LLR</td>
<td>Low-Level Requirement</td>
</tr>
<tr>
<td>MAL</td>
<td>More Abstract Layer</td>
</tr>
<tr>
<td>MC/DC</td>
<td>Modified Condition/Decision Coverage</td>
</tr>
<tr>
<td>MODSIM</td>
<td>Modeling and Simulation</td>
</tr>
<tr>
<td>NaN</td>
<td>Not a Number</td>
</tr>
<tr>
<td>NYU</td>
<td>New York University</td>
</tr>
<tr>
<td>PDR</td>
<td>Preliminary Design Review</td>
</tr>
<tr>
<td>PHAC</td>
<td>Plan for Hardware Aspects of Certification</td>
</tr>
<tr>
<td>PLD</td>
<td>Programmable Logic Device</td>
</tr>
<tr>
<td>PMA</td>
<td>Parts Manufacturing Authority</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>PR</td>
<td>Problem Report</td>
</tr>
<tr>
<td>PSAC</td>
<td>Plan for Software Aspects of Certification</td>
</tr>
<tr>
<td>RE</td>
<td>Reverse Engineering</td>
</tr>
<tr>
<td>RESP</td>
<td>Reverse Engineering Software Plan</td>
</tr>
<tr>
<td>RTCA</td>
<td>RTCA, Inc. (formerly Radio Technical Commission for Aeronautics)</td>
</tr>
<tr>
<td>SAS</td>
<td>Software Accomplishment Summary</td>
</tr>
<tr>
<td>SCI</td>
<td>Software Configuration Index</td>
</tr>
<tr>
<td>SCM</td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td>SCMP</td>
<td>Software Configuration Management Plan</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SDP</td>
<td>Software Development Plan</td>
</tr>
<tr>
<td>SECI</td>
<td>Software Life Cycle Environment Configuration Index</td>
</tr>
<tr>
<td>SEH</td>
<td>Simple Electronic Hardware</td>
</tr>
<tr>
<td>SME</td>
<td>Subject Matter Expert</td>
</tr>
<tr>
<td>SOI</td>
<td>Stage of Involvement</td>
</tr>
<tr>
<td>SQA</td>
<td>Software Quality Assurance</td>
</tr>
<tr>
<td>SQAP</td>
<td>Software Quality Assurance Plan</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>STP</td>
<td>Software Test Plan</td>
</tr>
<tr>
<td>SVP</td>
<td>Software Verification Plan</td>
</tr>
<tr>
<td>TDP</td>
<td>Technical Data Package</td>
</tr>
<tr>
<td>TSO</td>
<td>Technical Standard Order</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned Aircraft System</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very High Speed Integrated Circuit</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

Reverse Engineering (RE) is a class of development processes that start with detailed representations of software for a system, or hardware description for a device, and apply various techniques to produce more generalized, less detailed representations. The goal is to have more abstract representations that can be used to understand and reason about the structure and the intent of the more detailed representations. RE has been used in many industries, including aircraft applications, for mechanical, hardware, and software components. The scope of this report is the electronic hardware devices and software applications of RE for airborne systems and equipment.

The Federal Aviation Administration (FAA) sponsored this research task to provide a clear understanding of what should be considered acceptable RE for airborne software and airborne electronic hardware (AEH) devices, and under what conditions it could be deployed or restricted in the aircraft certification environment. The report is designed to provide an overview of the aviation industry’s views of RE, potential issues of employing RE for safety critical aviation systems, recommendations for when application of RE is acceptable, and some associated criteria for successful implementation.

The RE development process is opposite to the traditional waterfall model that has been a well-known commodity to the certification authorities, and therefore, RE, which is much less known, is of concern to the certification authorities. Without a common set of recognized and accepted terminology, definitions, and constraints on the processes and other issues, the certification authorities are forced to provide case-by-case evaluations of the different RE proposals.

This report provides the results of research into RE. The intended audience includes practitioners who develop compliance evidence, as well as evaluators, for airborne software and AEH to be approved under RTCA documents DO-178B and DO-254, respectively. The report may be used to obtain:

- A view of how others in the aviation industry understand and use RE. This includes analysis of views held by software and electronic hardware developers, Certification Authorities, and Designated Engineering Representatives (DERs).

- A consistent and uniform view of RE developed from industry experience, industry perspective and research within the field.

- Insight into the software planning processes that may need to be adjusted in support of RE. As RE is a development process, planning processes may need adjustment to ensure the integrity of the process steps.

- Insight into verification processes may also need adjustment to ensure that the resulting life cycle data satisfies the objectives of DO-178B and DO-254.

- Awareness of potential hazards and risks to a project if RE is used without ensuring various safeguards are employed. If information is misused in an RE process, certain
faults or errors could be concealed and could remain as latent faults or errors. By exposing the potential vulnerabilities, the process activities could be adjusted to mitigate the potential hazards, risks, and errors.

- Additional objectives and activities or constraints on the current activities and objectives could ensure that the use of RE is consistent with airworthiness requirements.

This research utilized historical data from certification projects conducted by Verocel, Inc. All Verocel projects considered for this study were completed, developed using RE, and achieved certification approval.

The research also employed an anonymous survey that was developed and sent to over 3900 participants in the aviation industry worldwide. The survey was designed to elicit opinions, experience, issues, and implementation success and failures relating to the use of RE. Survey responses were analyzed and the findings show some common threads and important results that form part of the basis of the report conclusions.

The first phase of this study proposed a framework for conducting RE on aircraft certification projects. This framework emphasizes the role of subject matter experts (SMEs) in ensuring that the as-implemented behavior is safe for its intended use, and that the relevant domain and systems knowledge has been captured in the requirements and design documentation. This second phase of the study has validated the framework by means of two case studies. The first case study chosen is the GNU New York University (NYU) Ada 9X Translator (GNAT) Ada Runtime Environment (ARTE). This was chosen for two reasons:

1. It is open source software distributed under the GNU General Public License (GPL).
2. Verocel has previously provided reverse engineered certification evidence for ARTE. As a result, it was approved by the FAA for use in airborne software applications, up to and including DO-178B Level A on the Boeing 787 Dreamliner.

The second case study chosen is an AEH case study based on the certification of two programmable logic devices (PLDs). The certification evidence was developed to design assurance level (DAL) B and delivered in support of a military project using DO-254. The intent of these case studies was to validate the framework against two examples which were developed using RE and which were approved.

These case studies assessed the compliance of the RE previously carried out by Verocel against the framework proposed. It was found that the RE process carried out by Verocel was consistent with the proposed framework. Indeed, a comparison of these completed projects with the proposed framework resulted in a small number of recommendations that would improve the Verocel process. Since the certification evidence created for both projects has already been accepted, these results provide confidence that new projects complying with the framework would also be acceptable.
1. INTRODUCTION

1.1 BACKGROUND

Reverse engineering (RE) is a class of development processes that start with detailed representations of an implementation, and apply various techniques to produce more generalized, less detailed representations. The goal is to have more abstract representations that can be used to understand and reason about the structure and the intent of the more detailed representations. Some examples of this would be the creation of requirements from source code or the creation of requirements of a complex electronic hardware (CEH) device implemented using an Application-Specific Integrated Circuit (ASIC) from the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) code. This type of development is very different from the traditional (forward) waterfall approach, and is therefore of concern to the certification authorities.

RE is not restricted to electronics. It is also used to produce design drawings from physical parts [3], or as the means to obtain the design and design approval of aircraft parts under the Federal Aviation Administration (FAA) Parts Manufacturing Authority (PMA) process FAA Order 8110.42C [4]. RE is accepted in principle by the FAA.

There are several reasons why RE is used. Here are some examples:

- In some communities, it is used so existing systems can be improved or modified. Examples include software programs that were written in one language and then translated into another language implemented on modern hardware.

- In other communities, it is used to generate artifacts needed to maintain existing implementations.

- Some use RE to gain information that was not intended to be published. Most software licenses prohibit the discovery of intellectual property in a program by RE, yet it is possible to extract the program intent to bypass license management checks or to create competing products.

- RE can be used as part of complete life cycles, such as rapid prototyping. Several possibilities exist including Agile development where code requirements and design are developed concurrently without formal baselines, no formal life cycle processes or formal transition criteria, or evolutionary prototyping [5] [6].

- Open source code is available under various license terms. The GPL is often used and the source code is freely available provided the GPL license is used to protect the rights of the users. [7] While GPL source code is freely available, other life cycle data is not always available. Compilers used under a GPL will have code libraries to support language features. If these features are used, then the source code will require life cycle data, and this must be reverse engineered.
Some compilers provide special options that restrict complex programming language constructs, or provide support for these constructs through special code generation sequences. Rather than use run-time libraries, the compiler may generate equivalent code in line with the application code. This code needs to be verified, and an intermediate representation may be generated so that verification evidence can be reverse engineered from this Less Abstract Layer (LAL).

RE has been recognized as a legitimate engineering discipline [8]. It has been used in many industries including aircraft applications for mechanical, hardware, and software systems. This report concentrates on the software and CEH applications of RE for aviation applications.

1.2 PURPOSE AND SCOPE

1.2.1 Why is this report needed?

The guidance in DO-178B [1] was designed to be independent of the life cycle process employed. Section 3 of DO-178B states:

“The guidelines of this document do not prescribe a preferred software life cycle, but describe the separate processes that comprise most life cycles and the interactions between them. The separation of the processes is not intended to imply a structure for the organization(s) that perform them. For each software product, the software life cycle(s) is constructed that includes these processes.”

It further describes many variations of processes that could exist, including a prototyping strategy. DO-178B, subsection 12.1.4 bullet 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.”

When a number of applicants proposed to apply RE techniques to their projects, the Certification Authorities Software Team (CAST) issued a position paper to express their concerns about RE [9]. RE processes differed between applicants. RE is also applied to hardware being approved under DO-254 [2]. Without a common set of recognized and accepted terminology and definitions, the certification authorities were forced to provide case-by-case evaluations of the different proposals.

In order to evaluate the issues that may be associated with the use of RE principles, and under what conditions, if any, they can be used, the FAA sponsored this research project to provide a clear understanding of what should be considered RE for software and CEH devices and under what conditions it could be deployed or restricted in the aircraft certification environment.
1.2.2 Intent of the report

This report provides the results of research into RE. The intended audience is practitioners who develop compliance evidence as well as reviewers, for software and CEH to be approved under DO-178B and DO-254, respectively. The report may be used to obtain:

- A view of how others in the aviation industry understand and use RE. This has been accomplished by conducting a survey and through analysis of survey results.
- A consistent and uniform view of RE. By analyzing the views presented in the survey responses, the findings show some common threads. These are described in the survey results analysis.
- Insight into the software planning processes that may need to be adjusted in support of RE. As RE is a development process, planning processes may need adjustment to ensure the integrity of the process steps.
- Insight into the verification processes may also need adjustment to ensure that the life cycle data satisfies the objectives of DO-178B or DO-254 even though information flow may not be in accordance with a traditional waterfall model approach. [10]
- Awareness of potential risks if RE is used without ensuring various safeguards are employed. If information is misused in an RE process certain faults could be concealed and could remain as latent faults/errors. By exposing the potential vulnerabilities, the process activities could be adjusted to mitigate the potential risks, and faults.
- More objectives or activities may be required in addition to those described in DO-178B. The DO-178B document describes objectives that must be met to provide assurance that the software of a system complies with airworthiness requirements. The objectives and their application are different depending on the assurance level of the system. Additional objectives and activities could ensure that the use of RE is consistent with airworthiness requirements. Similarly, additional objectives or activities may be required for CEH beyond those described in DO-254.
- Recommendations for the production of a new FAA Job Aid. This could contain practical information that would assist the certification and development community to deploy and approve RE processes in a consistent manner.

While RE can be applied to an artifact at any level of abstraction to develop a more abstract artifact, it is helpful to use a specific example in the sections that follow. Therefore, for illustrative purposes, this report will assume that low-level requirements (LLRs) are being reverse engineered from source code. This in no way implies that this report is limited to this set of artifacts and it is indeed applicable to all artifacts at any level where a more abstract description is desired.
This report only addresses the activity of developing or discovering More Abstract Layers (MALs) from LALs. This report does not deal with reviewing or testing artifacts to generate missing verification evidence.

1.3 USER GUIDE FOR THE REPORT

This report is designed to provide an overview of the aviation industry views of RE, potential issues of employing RE for safety critical airborne systems, and recommendations for when the application of RE is acceptable and some associated criteria for successful implementation.

- Section 2.2 summarizes the salient points obtained from the survey conducted to gain industry perspective about RE. Detailed results of the survey can be found in appendix B. The appendix can be used to look at detailed differences between various views of the data. For example, it is possible to see the percentage of projects that have been perceived as rejected by certification authorities versus developers.

- Section 3. examines the framework within which RE exists. This is based on the basic principles of RE and potential error sources as informed by the results of the survey and the issues identified in appendix C.

- Section 4. summarizes two projects that were assessed against the framework, in order to validate the conclusions of this report.

- Section 5. provides recommendations for guidance that could be used to provide a consistent approach for developing and approving RE projects and would prohibit practices that would be error prone. If implemented, these recommendations would provide visibility to ensure that all the DO-178B and DO-254 objectives can be satisfied by RE projects that follow the recommendations in this section.

- Section 6. provides some conclusions and proposed next steps based on current findings and analysis of the survey results.
2. SURVEY AND INDUSTRY EXPERIENCE

2.1 INDUSTRY EXPERIENCE

Historical data from the Verocel Problem Reporting databases was gathered. Thirteen (13) projects were considered totaling over 250,000 effective lines of code (ELOC), which averages 357,000 lines of code per project. These projects were all complete, developed using RE, and had certification authority approval. Sixty-seven percent of the projects were approved by DERs on behalf of the FAA and thirty-three percent of the projects were approved by the Department of Defense (DOD). Information about error rates, error sources and other details were extracted and consolidated into a database so that it could be analyzed. The results of the analysis are published in appendix A.

2.2 INDUSTRY PERSPECTIVE FROM SURVEY

A survey designed to elicit opinions, experience, issues, and implementation success and failures was developed and sent to over 3500 participants in the avionics community. The survey was anonymous, and worldwide, although the expectation is that most of the respondents were from the United States (U.S.) and Europe. Details, assumptions, and validity discussions of the survey are contained in appendix B. This section summarizes the important results of the survey that form part of the basis of the conclusions in Section 6.

2.2.1 Software survey results

There were 162 respondents to the software survey. Roughly 30% of these were responsible for approving software: certification authorities and DERs; the rest were industry practitioners. The respondents had an average of 16 years of software development experience. 40% of the respondents reported that they have worked on one or more projects where RE was used. DO-178B [1] Level A and Level B projects (or applications/products) dominated the application of RE process. While Level D projects had the fewest respondents, this might be explained by Level D’s concentration on high-level requirements (HLRs); it is unlikely that source code would be used to develop the HLRs using an RE process. However, the survey did not test for this conclusion. Furthermore, the RE process spanned most of the major airplane types (i.e., transport, regional, small aircraft) as well as rotorcraft. Low participation was registered for engines, ground-based systems, and Unmanned Aircraft Systems (UAS). Over 65% of the developers have used RE on software that has been approved. About 60% of the participants work on Technical Standard Order (TSO) projects with the rest being involved with the issuance of a Type Certificate or a Supplemental Type Certificate. Additionally, less than 1% of RE projects were not approved at all (note this represents only one project so should be treated as a potential anomaly). This demonstrates that RE is widely used by highly experienced certification engineers and developers and is not a niche development process. With the large prevalence of RE projects, it is worthwhile to produce guidance and information on the subject.

The vast majority of the respondents believed that RE was used to discover all or some of the missing higher-level software representations. However, a significant number of respondents included the verification activities as part of the RE process. This demonstrates the need to
provide a standard interpretation of RE to ensure consistency. This is provided and explained in subsection 2.3.1

While the survey respondents provided a strong indication that the FAA’s guidance was insufficient, they were not necessarily dissatisfied with the FAA’s approach for granting approvals. The approval authorities (Certification Authorities and DERs – we will use the term Certifiers for this combination) have approved the overwhelming majority of RE projects. While 18% of the Certifiers indicated they reject any project that contains RE, the developers reported that less than 5% of the RE projects have been initially rejected. Although not specifically asked, the assumption is that by providing additional documentation and life cycle data, projects did achieve approval in spite of the initial rejection. Of those that were approved most required modification to the plans or Plan for Software Aspects of Certification (PSAC). Less than 45% of the Certifiers approving projects used CAST-18 for information. Almost 20% of the developers used or believed the certifiers used CAST-18 to evaluate their RE project. The results of the survey indicate that while RE projects are being approved, there is not a well-defined set of criteria that can be used by both the Certifiers and developers to ensure a consistent approval. Furthermore, the lack of guidance results in unnecessary rework to plans and PSACs.

The majority (about 60%) of the developers indicated that they had access to domain experts during the RE process. The vast majority of respondents said that they used source code as the starting point to reverse engineer other required artifacts. Over half of the respondents used RE to produce missing artifacts from commercial off-the-shelf (COTS) software or from previous projects that lacked the requisite artifacts. Additionally, a large majority used an iterative approach where code is developed from draft specifications of the behavior and these, as well as other artifacts, are refined and developed through multiple iterations of the process. Some respondents indicated that they developed source code from machine code. A small but significant minority that used RE indicated that they didn’t use RE to develop LLRs from source code, indicating that the process was used for other phases of the life cycle (e.g., HLRs from LLRs). A large number of developers used tools to assist them in the RE process. Any guidance or recommendations needs to be compatible with these different approaches to RE.

The survey contained a section for free-form comments for many of the survey questions. Most of the comments were related to FAA policy. However, these comments didn’t necessarily address any specific questions in the survey. It was difficult to establish any common theme among the comments. The few comments that were similar concerned criticism of the survey and the guidance provided by the FAA.

2.2.2 Complex Electronic Hardware (CEH) survey results

Since there were only 19 responses to the survey for CEH, no valid conclusion can be drawn regarding the specific use of RE for CEH. However, because the basic principles use the notion of higher and lower levels of abstraction (i.e., MAL, LAL) rather than the specific artifacts (e.g., source code, HLRs, LLRs), the principles derived from the results of the software survey are believed to be applicable to CEH.
2.3 TERMINOLOGY

2.3.1 Reverse engineering (RE)

A definition of RE was developed and provided in an introduction to the survey:

“The use of one or more development processes which result in representations of the software for the target computer environment, and these processes are analytical techniques using information from a representation at a level closer to the target computer environment to produce representations at a more abstract level.”

The survey included explanations in the introduction and adjustments to the questions to cover the differences for CEH. “The two surveys are very similar but will use a different word to represent one of the representations. We will use the word ‘Software’ for DO-178B/DO-278 and ‘Programs’ for the equivalent representation used in Programmable Devices for DO-254 [2].” For questions, the wording “Developing source code in a programming language such as Verilog, VHDL, RTL, etc.” was used in the CEH survey.

To ensure that a common understanding of RE is used the above definition is broken down and explained, please consider the following:

• “RE consists of one or more development processes…”

RE is a development process. One or several processes may be used while using RE. Examples include LLR development and source code development. RE does not include verification processes, so reviews, test development, and other verification activities are not the subject of RE. The verification activities are still required to show compliance with DO-178B.

• “…which result in representations of the software for the target environment…”

The representations of the software/CEH can take many forms. This includes requirements, design, source code/VHDL, but does not include test cases or tests, which are verification artifacts. The definition does not specify the abstraction levels or the form of the representation. For example, HLRs or LLRs may be developed in tables, documents, or even diagrams.

• “…and the processes are analytical techniques …”

The processes lend themselves to reasoning in a consistent manner – the processes could be manual or they could be automated.

• “…using information …”

This is the key to this definition: information is being used in the development processes.
• "...produced from a representation at a level closer to the operational system configuration (e.g., executable code) than that information produced from a representation at a more abstract level (object code)."

The lowest level is executable object code for software and the circuitry or "burn map" for custom micro-coded devices (AEH), and the more abstract level above that would be object code/bit map/mask, and the more abstract level above that would be source code/VHDL, and the more abstract level above that would be LLRs, and so on.

For example, if information is used from the source code/VHDL to develop LLRs, then that is RE. If LLRs are developed after the source code, but the source code is not visible to the requirements developers, then that is not RE.

2.3.2 Configuration management of reverse engineered artifacts

Unless the representations at various levels are identified, versioned and tracked, it may be difficult to determine the order in which the representations were developed, and it may be difficult to determine if information was used from a more specific representation to develop a more abstract representation. This approach makes it more difficult to determine how much RE, if any, was used. If it cannot be determined that information was not used in reverse, then the assumption should be that some level of RE may have occurred.

For CEH devices, the definition follows the same principles, with representations becoming more specific the closer they are to the final implementation. RE as defined is the process of using information from a specific representation of the program on a hardware device to produce a more abstract representation (inductive reasoning).

2.3.3 Certifier

Certifier refers to a member of a Certification Authority or DER.

2.3.4 Forward engineering

A process in which each LAL is produced from a MAL, e.g. a process that follows the waterfall model.
3. FRAMEWORK FOR PERFORMING RE

A framework was developed to provide a consistent description of the RE processes that can be used to expose vulnerabilities and identify potential benefits of the approach. This framework can be used to develop policy recommendations to capture the objectives and activities that mitigate the potential vulnerabilities and risks of RE.

3.1 SOFTWARE DEVELOPMENT PROCESSES AND SEQUENCE DEPENDENCIES

Section 3 of DO-178B [1] provides a sequence of independence descriptions of software life cycle processes. Specifically: “The guidelines of this document do not prescribe a preferred software life cycle, but describe the separate processes that comprise most life cycles and the interactions between them”. It does not follow that all life cycles constructed of these processes can be shown to comply with DO-178B. In this section, any life cycle that can be shown to satisfy the guidance of DO-178B is considered acceptable.

This section identifies three major processes – planning, development, and integral. The development process is refined further into requirements, design, coding, and integration. The integral processes are further refined into verification (including reviewing, analyzing and testing), configuration management, quality assurance and certification liaison. Any description of RE needs to incorporate these processes that reflect the processes sequencing, transition criteria and development activities, in a manner that will allow a determination of whether the processes can be shown to comply with DO-178B. This is captured in the following sections.

3.2 COMPATIBILITY OF REGULATORY GUIDANCE WITH RE PROCESSES

DO-178B section 3: Software Life Cycle, section 4: Software Planning Processes, and section 5: Software Development Processes provide the framework for all development processes and associated guidance. The following paragraphs discuss some incompatibilities among section 3, 4 and section 5 and how that would affect the oversight of RE projects.

Section 3 of DO-178B, subsection 3.0 states the following:

“This section discusses the software life cycle processes, software life cycle definition, and transition criteria between software life cycle processes. The guidelines of this document do not prescribe a preferred software life cycle, but describe the separate processes that comprise most life cycles and the interactions between them. The separation of the processes is not intended to imply a structure for the organization(s) that perform them. For each software product, the software life cycle(s) is constructed that includes these processes.”

This allows the individual development and integral processes (verification, quality assurance, configuration management and certification liaison) to be described without regards to any specific life cycle. This permits multiple life cycles to be defined from these processes with different sequences. An applicant must define their life cycle by providing plans and procedures describing the individual development and integral processes as well as the sequencing of these processes and associated transition criteria to go from one process to another as described in
subsection 3.2. Subsection 3.2 introduces figure 1 which shows some examples of life cycles. All of these life cycles are representatives of forward engineering approaches.

While the examples are from forward engineering life cycles, none of the material in section 3 would conflict with a “well-defined” RE strategy.

DO-178B, subsection 4.1 makes the following statement:

“The purpose of the software planning process is to define the means of producing software which will satisfy the system requirements and provide the level of confidence which is consistent with airworthiness requirements. The objectives of the software planning process are:

a. The activities of the software development processes and integral processes of the software life cycle that will address the system requirements and software level(s) are defined (subsection 4.2).

b. The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria are determined (section 3).

c. The software life cycle environment, including the methods and tools to be used for the activities of each software life cycle process, has been selected (subsection 4.4).

d. Additional considerations, such as those discussed in section 12, have been addressed, if necessary.

e. Software development standards consistent with the system safety objectives for the software to be produced are defined (subsection 4.5).

f. Software plans that comply with subsection 4.3 and section 11 have been produced.

g. Development and revision of the software plans are coordinated (subsection 4.3).”

The rest of section 4 provides expansion on the above basic principles. Any definition of an RE life cycle would have to have plans that would have all of the information described in subsection 4.1.
Subsection 4.6 provides the basic goals of review and assurance of the planning process, which are repeated below:

“Review and Assurance of the Software Planning Process

Reviews and assurance of the software planning process are conducted to ensure that the software plans and software development standards comply with the guidelines of this document and means are provided to execute them. Guidance includes:

a. The chosen methods will enable the objectives of this document to be satisfied.

b. The software life cycle processes can be applied consistently.

c. Each process produces evidence that its outputs can be traced to their activity and inputs, showing the degree of independence of the activity, the environment, and the methods to be used.

d. The outputs of the software planning process are consistent and comply with section 11.”

There is nothing in section 4 that would show a preference or a bias toward either a forward engineering approach or a RE approach. In summary, the goal of section 4 is to require plans, procedures, and standards that will ensure that all of the objectives are satisfied, processes can be applied consistently, and the implemented processes comply with DO-178B. Any plans or standards for an RE project that can accomplish this should be acceptable under section 4.

Section 5 of DO-178B discusses each of the development processes and the objectives and activities associated with each. Some of the guidance in this section is based solely on a forward engineering approach. Subsection 5.2.1 bullet a. lists the objectives associated with the design process. These objectives specify that the LLRs and architecture are developed from the HLRs. This corresponds to objectives 3 and 4 summarized in Annex A, Table A-2. All of the other objectives are worded in a manner that would make it independent of either a forward engineering or RE development approach. The current wording in subsection 5.2.1 bullet a. would make it impossible to comply with those objectives if RE was used.

The remaining sections are devoid of any forward engineering or RE dependencies.

DO-178B section 12.1.4 bullet d states that reverse engineering may be used to regenerate software life cycle data that is inadequate or missing in satisfying the DO-178B objectives. If the certification authorities wish to provide a consistent implementation and assessment of RE projects, then additional guidance will be needed to modify the current wording in subsection 5.2.1 bullet a. Additional explanation should also be provided to ensure that subsection 3.2 can include RE projects.
3.3 ROLES

The following roles are identified for the software engineering processes. While these roles are similar to other life cycle descriptions, in the descriptions below the unique aspects for RE are identified. These roles may be performed by people with tools or without tools:

- **Subject Matter Expert (SME):** A role that possesses knowledge of the required behavior of the final product and/or interim representations of the product. This role may or may not require any software expertise depending on the representation of the product. This role can provide judgments on whether the operational behaviors of the product are acceptable. While the survey used the term ‘domain expert’, SME is used to provide a more general notion not restricted to aviation applications. Examples of SMEs include experts in autopilot systems, engine systems, operating systems, mathematical functions, etc. When requirements are being developed from source code, an SME who has expert knowledge of the source code is required. This SME could be the original software developer. Inconsistencies and errors in the source code would be resolved by this SME. For complex systems, it is unlikely that any one individual will possess expert knowledge of the system domain as well as detailed knowledge of the source code, so a team of SMEs will be required. In a forward engineering process, this role provides the starting point for development, hence is implicit. In RE processes, this role must be made explicit.

- **Requirements developer:** A role that possesses the ability to develop more abstract requirements from less abstract requirements or design. For example, this role could develop the LLRs from the source code, or HLRs from LLRs. As this role develops requirements, then it must also establish trace data from the source of the information to the destination of the information. Provided the trace data is bi-directional, traceability from higher-level requirements to lower-level requirements or LLRs to source code is established.

- **Architecture developer:** A role that possesses the ability to develop architectural descriptions from some detailed implementation representation. For example, this role could develop the architecture description from the source code.

- **Source Code developer:** In the RE context this could mean (a) a role that possesses the ability to develop the source code from the object code, or (b) a role that develops source code for which the LLRs and architectural descriptions are incomplete or non-existent.

- **Verification role:** A role that confirms the properties described in (Section 6) the Verification plan. There are two types of properties:
  
a. Properties that apply to the relationship between representations produced at different abstraction levels. Examples include, Traceability (Do the HLRs trace to the correct LLRs? And vice versa?), Compliance (Does the intended behavior of a set of LLRs represent the intended behavior of the HLRs to which they are traced?), etc.
b. Properties that apply to specific products. Examples include, Verifiability (Are the HLRs, LLRs and source code verifiable?), Consistency (Do the requirements, architecture descriptions or source code comply with the Requirements Standards, Design Standards, or Coding Standards?), etc.

- Configuration manager: A role that establishes the configuration management properties of any life cycle data (e.g. identification of individual components, version control, establishment of baselines, change control, etc.). There are no unique properties required for RE but proper application of configuration management states may be more critical during RE as the products are baselined or versioned in reverse order (source code is baselined or versioned before the LLRs are developed, LLRs are baselined or versioned before HLRs are developed).

- Quality Assurer: A role that ensures that the process and procedures defined in the plans are being followed by all executers of that plan, including the transition criteria. This role is also responsible for performing the conformity review. There is nothing unique to RE about this role assuming the plans and standards have been completely defined, and if RE is being used, then this is also described in the plans and standards, including the transition criteria for the defined processes.

3.4 PROCESSES

This section defines a generic prototype for performing an RE activity for software.

3.4.1 An example of source code to LLR development

An example of an RE set of activities is shown in figure 1. Only the pertinent activities for the example are shown. A set of processes may combine some of these activities, as this may be a more efficient approach. The process plans should describe this together with the evidence produced.

The source code is developed without LLRs or architectural descriptions. Even if some LLRs and architectural descriptions exist, unless they are baselined and controlled through configuration management, then some level of RE may occur.

The source code should be reviewed to ensure its properties are satisfied. This can be done in the absence of requirements as this is focusing on process compliance with coding standards, and internal consistency. At this point, the source code should be baselined and changes should follow a consistent problem tracking process. In an RE process this may be performed informally during the “Develop LLRs” process and repeated formally during the source code review process.

The LLRs can be developed using available information. The source code could be used in this RE activity, but if available, any additional specifications, descriptions, and higher-level requirements may also be used. The sources of information should be baselined during the development process to ensure a consistent starting point. Traceability to higher-level requirements is required, but is not shown in this diagram.
The LLRs may be baselined and reviewed for internal consistency, and conformance to the requirements standards.

The source code may then be reviewed against the LLRs, and both of these must be baselined. At this point, the activity focuses on the consistency (compatibility) of the source code against its requirements. Note that this activity may find problems in the source code or the LLRs, as both of these are used in the review process.

Figure 1. Example processes for development and review of LLRs using RE

3.4.2 Generic RE processes

The above example may be generalized so that it applies to any level during the development and verification life cycle. The RE process is a combination of individual processes that perform some activities using a set of inputs to produce an output. Some of the inputs are used to guide and control the activity and some of the inputs provide information that is used directly to produce the output.

The processes in this section assume that an artifact at a MAL is being developed from an artifact(s) at a LAL. A generic process is defined first and then it is instantiated for different specific processes along with how the process must be designed to comply with DO-178B.
Figure 2. Generic processes for development and review of different abstraction layers

3.4.3 Inputs

3.4.3.1 LAL artifact

The LAL artifact is the result of a completed process activity. This may be a complete artifact, for example, all of the HLRs for a software/AEH program. It may be part of an artifact, for example, the LLRs for part of the software/AEH program. It may also be a component of an existing artifact/system, for example, a math library source file that includes the functions required by the software/AEH program.

The LAL artifact(s) must be identified and versioned, and it must be possible to obtain an exact copy of the LAL artifact(s) from a configuration management system.

3.4.3.2 Project-specific RE development process

The process activities to be performed must be described. This description could be part of the standard processes, if they describe RE, or they could be specific procedures, addendums, work orders, etc. which make appropriate adjustments for RE.

3.4.3.3 RE standards applicable to the development process

In a forward engineering process plan, the standards for requirements, design and for source code development should exist before any of requirements, design and source code is written. In an RE process the code may exist before a formal coding standard is written.
In an RE process, a coding standard should be developed before any development of the requirements is started. This will provide a level of assurance that even though the original developer did not produce the code in accordance with a coding standard, the engineer developing the traceability between requirements and source code will provide this first level check.

The standards should be no different whether a forward engineering or RE process is used, but it is important that they be available and should be used in accordance with the development and verification processes.

3.4.4 Outputs

3.4.4.1 MAL product

This may be a complete or partial product depending on the specific defined RE life cycle process. For example, if a program uses information managed as a linked list, then a set of functions may be provided in a library that groups these functions together. Such a library may have LLRs developed before the rest of the software. This would be a partial artifact and, provided the requirements can be identified, versioned and baselined, they can be developed ahead of the rest of the program. The final MAL product would be a collection of the partial artifacts. This collection would be identified with a baseline identifier such so that it could be carried forward to the certification package and identified in a software/AEH configuration index.

An alternative iterative life cycle may produce an artifact(s) that will contain all of the LLRs required eventually, but the artifact(s) could be partially completed containing sections for the requirements that have been developed at any particular iteration. The artifact(s) must be correctly versioned and controlled such that only authorized changes are incorporated in subsequent revisions. The final artifact(s) would be identified and carried forward to the certification package and identified in the software AEH configuration index.

The MAL product need not be a specific artifact while in partial state. The information could be held in a requirements repository, held in a database, organized in tables, or in sets of related hypertext files. It is however key, that the artifact(s) is clearly identified and can be retrieved for inspection. There may be several levels of MALs, which link the levels of abstraction. At the highest level (the most abstract level), the MAL would be represented by HLRs, which are traced to system requirements. These should be written in a way that an SME can understand the HLRs, and can confirm the traceability between system requirements and HLRs.

3.4.5 Entry criteria

3.4.5.1 Status of the inputs

This would be a specification of the configuration management criteria for the LAL product whose information would be used by the RE process.
3.4.5.2 Status of the process

This would be a specification about the status of the process description for the development process being started. For example, this could require that the plans and standards for the process be approved and released, and the PSAC approved or planning review completed.

3.4.5.3 Status of the standards

Other inputs that may affect the development of the MAL or play a role in detecting errors would be the standards. These should be under configuration control.

For example, in the case of developing LLRs from source code for a math library, the transition criteria for starting the development of the LLRs would require that the source code be under configuration control and verified to comply with a set of minimal coding standards.

3.4.6 Exit criteria

3.4.6.1 Status of the output

This would be a specification of the configuration management criteria and the development criteria needed on an output artifact(s) that would establish the completion of the development process.

Different exit criteria may be defined for partial completion and for total completion of the RE process. If only complete baselines are used, then all of the code could be reviewed against the coding standard, followed by the development of all of the LLRs for that code, followed by the development of HLRs for those LLRs. This requires careful configuration control as well as problem report (PR) tracking as a revision to a version of the output will contain changes for many change requests. An iteration of the MAL product could contain many updates.

The MAL partial product could be hierarchical to reflect the software or requirements hierarchy. For example, in the case of developing LLRs from source code for a math library, the partial artifact could be the LLRs for a single math function. Another partial artifact could be the LLRs for a different math function from the same math library.

It could be that a LLR is reviewed and the LLR and its source code is under configuration control. This might require that all source code is traced to the LLR and all LLRs have completed a design review, to include the SME.

3.4.6.2 Completion criteria

The completion criteria of the artifact depend on how the artifact is organized and managed.

If the artifact is a single entity, such as a complete LLR document, then the completion criteria must include the change history of all changes made to the document and show how each change was dispositioned.
If the artifact is organized hierarchically, then the completion criteria must also be organized hierarchically. If all of the functions in the math library have had LLRs developed, then the LLR representing the math library itself can be completed, as it is at that point that its completion can be confirmed.

The completion criterion for a partial artifact is a statement of the form “ready for review”.

3.4.7 Process description

This would describe the complete set of process steps that would transform the inputs to the outputs including any process path selections due to the entry and exit criteria. This would also discuss any actions for exceptions as well. For example, the process may state that the developer shall establish all of the LLRs for any “.c” file for a math library that meets the entry criteria.

The process description should reference the applicable standards to be used or project-specific procedures that may refine the activities. For example, during the development of some LLRs from source code using an RE process, the engineer may find sequences of conditions that use arithmetic operators instead of logical operators. This may violate the coding standards because arithmetic operators combining conditions do not prescribe an evaluation order. Short circuit logical operators are typically preferred. The coding standard violation should be raised as a PR during the development process and should not wait until the subsequent code review process (which cannot start until the LLRs are baselined).

During the RE process, the engineer should find and record problems in the LAL product if they cannot develop a complete MAL product. For example, if a function in the source code has a parameter that points to a value to be used, then a requirement may be inserted for a robustness check to ensure the pointer has been initialized before use. If the source code does not have a null pointer check prior to dereferencing the parameter, then this should be raised as a PR and not just be left to the source code reviewer to address.

3.5 RE ASPECTS OF SOFTWARE VERIFICATION

The verification process for the delivered system must have some means of establishing total correctness, where correctness means consistency with the requirements. Since software must execute in the context of a system, the system behavior must be assured to be correct. However, this is where traditional forward engineering processes differ from RE processes.

In forward engineering, each individual development process uses an assumed correct artifact as input. The process can introduce some errors, which must be discovered by the verification process. These errors can be detected by analysis, reviews, or tests based on the correctness of the input to a previous process. In the case where source code is developed from LLRs, we may assume the LLRs are correct and any discrepancies between the LLRs and the source code will result in changes to the source code. The common theme of all these evaluations is the assumption on the correctness of the inputs.

In RE, we may know little or nothing about the correctness of the input to the process. For example, having only the source code for math libraries we do not know if the source code properly implements the math libraries or, in some cases, which math functions are implemented.
So as we develop MAL from LAL (e.g., LLRs from source code), we can only assure that the MAL is a faithful representation of the LAL, but not whether all the required functions are present (completeness) or whether those functions are the desired or correct function. At some point we must have some way of establishing the correctness of the final MAL (e.g., system requirements or HLRs).

All the software requirements are verified by reviews and analysis in accordance with the DO-178B objectives. Any new derived requirements, or any changes to existing derived requirements, should be provided to the system processes, including the system safety assessment process. If any system requirements were found to be inadequate or incorrect, the RE process should capture the issues and refer them to the system processes for resolution. The SME should be involved in liaising with the system processes.

The SME role in relationship to RE is critical. The SME is one or more persons. The SME may be aided by a specification for a similar system, a set of desired system specifications refined from some knowledge of the end system, or some combination thereof. The RE process must establish the correctness of the last MAL artifact in the development chain.

The verification process, in general, would be the same for any specific development process regardless of the sequence of processes. The goal would be to evaluate an artifact produced by a development process. This evaluation would include a comparison of the developed product (e.g., LLR) against the artifact from which it was developed (e.g., source code) for the DO-178B criteria of traceability and compliance. The two-way traceability would ensure that the artifacts are a faithful representation of each other. Additionally, the DO-178B process provides assurance that specific desirable properties exist in the developed artifact. These verification goals are independent of the order of development. For example, the verification process for a specific development process is not sensitive to whether the source code was developed from the LLR or the LLR was developed from the source code. The description of the development objectives in DO-178B section 5 tends to be dependent on the direction of development. For example, table A-2(4) reads, “Low-level requirements are developed”, but DO-178B section 5.2.1a reads “The software architecture and low-level requirements are developed from the high-level requirements”. This is inconsistent with the statement in DO-178B section 12.1.4d, which reads, “Reverse engineering may be used to regenerate software life cycle data that is inadequate or missing in satisfying the objectives of this document”. The verification objectives in DO-178B section 6 tend to be independent of the direction of development. For example, table A-4(1) reads “Low-level requirements comply with high-level requirements”, and DO-178B section 6.3.2a reads, “The objective is to ensure that the software low-level requirements satisfy the software high-level requirements and that derived requirements and the design basis for their existence are correctly defined”.

Another aspect of the RE process is the behavioral discovery process. In some cases the RE process might discover a behavior that is desirable at the end system level but the SME role did not include this behavior in their system view.

This may occur during a forward engineering process or an RE process. In a forward engineering process the insertion of additional behavior is normally limited to additions of robustness checks, which catch errors discovered through possible consistency violations (e.g., range checks before
indexing). Other insertions are less likely because the developer is producing the LAL under the assumption that the input as presented in the MAL is correct.

The RE processes proposed include propagating information from the LAL to the MAL, and raising PRs if necessary. The RE process may develop derived LLRs, which would be propagated to the SME who would interact with the System Development and Safety Assessment processes.

Another source of problems may be lack of functionality in the LAL, or the SME may identify required functionality at the system level that was not contained in the original LAL used to initiate the RE process. Conversely, the LAL may contain functionality not needed at the MAL or system level for a specific installation, potentially resulting in extraneous or dead code at the system/product function level.

Process models proposed for a RE approach should incorporate an SME to be a part of the verification process.

3.6 GENERIC RE SME VERIFICATION PROCESS

3.6.1 Inputs

The inputs to the SME verification process include the highest level of software requirements, which should be traced to a set of system requirements. The SME must be familiar with the intended behavior of the system. Their role is to ensure that the intended behavior of the system matches the intended behavior of the software.

3.6.1.1 MAL artifact

This may be a complete or partial artifact. An example would be where all of the reverse engineered HLRs are available for an aircraft power control system, or only a specific subset of HLRs for the power distribution system associated with the engine start function are available. It is also possible that portions of the reverse engineered HLRs are available as would be the case in an iterative development process.

3.6.1.2 SME knowledge base

This would be the collection of data that would be used to verify/validate the correctness and completeness of the MAL artifact. This could be a set of forward engineered system requirements, the mental model of a human expert, perhaps aided by specifications from a similar or previous system.

3.6.1.3 RE standards applicable to the SME verification process

The standards used by the SME would be the same as those used by the software developers and verification engineers. As the SMEs also work at the system boundary, they may be required to use additional System Development and Safety Assessment standards.
3.6.2 Outputs

3.6.2.1 Verification records

This may be a complete or partial artifact depending on the specific defined RE life cycle. For example, an iterative life cycle may produce multiple versions of HLRs to be verified as more functionality is added from the RE process.

3.6.2.2 Problem reports (PRs)

These would identify the discrepancies between the SME knowledge base and the MAL artifact. Some of the PRs may be genuine problems, and some may be issues raised by the developers due to incomplete information or to confirm specific assumptions. For example, these could represent errors between the SME view of how a tangent function should respond and the HLR for the tangent function. Responses to denormalized floating-point numbers could be underspecified, and the treatment for Signaling NaNs (Not a Number) could have been specified while a Quiet NaN could have been missed. The HLR developer could have raised an issue using a PR addressing the relative precision of the tangent function as its return value approaches the asymptote.

Even though the PRs are resolved, they form a record of the discussion between the SME and the development and verification teams.

It is important to identify the SME in the PR process.

3.6.3 Entry criteria

3.6.3.1 Status of the inputs

This would be a specification of the configuration management criteria and the verification criteria needed on an input artifact or information before the specific SME verification process would start. For example, in the case of HLRs for a math library, the transition criteria for starting the SME verification activity could be a requirement that the HLR be under configuration control and all verification tasks for the HLRs have been completed with no open PRs. An additional requirement could be that the system requirements for the math library be documented and under change control.

3.6.3.2 Status of the process

This would be a specification about the status of the process description before the SME verification process is started. For example, this could require that the plans and standards for the SME verification process be released and the PSAC approved or planning review completed.

3.6.4 Exit criteria

3.6.4.1 Status of the output

This would be a specification of the completion status of the SME verification results.
Different exit criteria may be defined for partial completion and for total completion of each process activity. For example, there may be a requirement that each HLR can be traced to a review record.

3.6.5 Process description

This would describe the complete set of process activities that would transform the inputs to the outputs including any process path selections due to entry and exit criteria. This would also discuss any actions for exceptions as well. For example, the process may state that the system engineer completes the system requirement/HLR checklist for each system function. If there are any functions discovered in the HLRs that should be added to the system requirements, a system level PR should be generated.

3.7 ACCEPTANCE CRITERIA

In a forward engineering process, the developer is expected to understand the MAL before they can develop the LAL artifacts. For example, a source code developer will need to understand the LLRs before developing the code; otherwise, the code will not comply with the LLRs. In an RE process, the LLRs may be developed from source code without due regard to the difference in the abstraction level, with the resulting LLRs being too close a match to the code, e.g. pseudocode. Since such translations may be performed with little intellectual effort or understanding the intent of the code, these practices should not be permitted on RE projects as they do not provide the same level of confidence as a forward engineering process.

The difference between abstraction levels should demonstrate some level of understanding, either by differences in the representations or the provision of additional information, such as context information or rationale between HLR and LLR mapping.

The other acceptance criteria follow on from those required for DO-178B and DO-254 [3].
4. VALIDATION OF FRAMEWORK

The framework was validated by examining two projects that had already been accepted by the FAA, and assessing how closely these projects matched the proposed framework. One software project and one AEH project were chosen. The two projects were:

1. GNAT ARTE. This project was chosen because it is open source software licensed under the GPL, and can therefore be freely distributed.

2. An AEH project involving two PLDs.

The use of RE on both projects were found to be consistent with the framework described in this report. We therefore believe that if an RE project adopts the framework suggested in this research report, that it will be likely to be acceptable to the FAA. The detailed assessment of these projects can be found in appendix D and appendix E.

5. RECOMMENDATIONS

The recommendations resulting from this research are:

1. If RE is to be used on a project, then it is important that the process plans and standards should clearly describe the information flow during the development of the system or software component(s) or AEH component(s). While this is expected from a well written set of plans and standards, the information flow may be assumed and not documented clearly. The transition criteria, configuration management (especially when data is placed under configuration control), problem reporting, change control, and status accounting may differ from the corresponding activities on a forward engineering project.

2. The PSAC or Plan for Hardware Aspects of Certification (PHAC) should be open and direct in describing the RE processes being proposed. Failure to explain clearly how RE is to be used on the project is likely to lead to rejection of the PSAC or PHAC, or cause problems when the extent of RE is uncovered during the Stage of Involvement (SOI) audits.

3. QA records should describe in-process audits conducted to ensure that process plans and standards are being followed. While this is not specific to RE, it may be more difficult to track the information flow, so it should require additional attention.

4. The use of SMEs is critical to any RE project. Their use should be identified in the PSAC/PHAC and required as part of the guidance. Depending on the extent of RE on a particular project, SMEs are likely to be required to cover the following topics:
   a. Domain and system knowledge, and liaison with system processes
   b. Software architecture and source code

5. Careful consideration should be given to the difference between abstraction levels, to ensure that there is sufficient intellectual value added to demonstrate a thorough
understanding of the two representations being traced and verified. The difference between the abstraction levels should such that through sound engineering judgment a higher level representation could be transformed to an equivalent lower level representation through the application of sound design decisions. For example:

a. If the LLRs are a simple restatement of the code (e.g. pseudo-code that is very close to the code), then testing based on such LLRs will only exercise the implementation and not the expected behavior and functionality. The capacity of low-level testing to detect incorrect functionality, missing functionality, and unintended functions will be severely compromised. The design decisions representing the difference between LLRs and code are non-existent or unsound.

b. If the gap between the HLRs and LLRs is too great, then it will be difficult to verify that the HLRs were developed correctly into LLRs, and derived requirements may be missed. The design decisions between HLRs and LLRs are too big and it would too difficult to develop HLRs into equivalent LLRs using sound design decisions.

6. It is particularly important that the LAL should be placed under configuration management before commencing the development of the MAL. It can be difficult to determine whether RE is being used if the artifacts are not under proper configuration management.

7. When reviewing each reverse-engineered artifact, the following issues should be considered:

   a. Is there more functionality in the LAL than required by the MAL — are the LAL elements necessary?

   b. Is there less functionality in the LAL than required by the MAL — are the LAL elements sufficient?

   c. Are there any errors in the LAL-MAL translation — is the translation between the LAL and the MAL correct?

8. It should be a policy that when RE is performed, all robustness checks should be called out as LLRs. This will ensure that they are considered in the testing process, the source to LLR reviews, and propagated up in line with DO-178B [1] objectives.

9. The Software Requirements Standards and Software Design Standards should define how RE techniques are to be used, when appropriate.

10. The process plans should describe and formalize the use of SMEs.

11. The review checklists should ensure that the LLRs show some level of understanding of the intent of the source code, and are not too close a match to the source code.
While the report is written using DO-178B [1] as the primary reference, the recommendations also apply to AEH components reverse engineered to comply with DO-254 [2].

6. CONCLUSIONS

The survey found that the majority of respondents have already used RE. Nevertheless, our literature survey found very little published research on the application of RE techniques to safety-critical software. The majority of the reported projects using RE were accepted by the FAA, though usually with changes requested to the PSAC/PHAC.

Although DO-178B [1] contains an explicit reference to RE, the majority of the guidance is written from a forward engineering perspective. There is therefore need for additional guidance for RE, to incorporate the recommendations we have made in Section 5. above. The development of this additional guidance is out of scope of this report. The references in this report relate to DO-178B. While the belief is that this report should be applicable to DO-178C based on the current state of the document at the time this report is released, an evaluation of the impact of DO-178C on this report will have to be done after DO-178C is released.

The framework was validated by assessing two RE projects against the framework. These two projects had already been accepted by the FAA. One software project and one AEH project were chosen. The RE activities performed on the two projects were found to be compatible with the framework described in this report. The software project is described in appendix D, while the AEH project is described in appendix E.

The survey showed the importance of projects being open and honest in their use of RE. The RE aspects need to be addressed during the planning phase, otherwise the PSAC/PHAC may be rejected. The use of RE needs to be agreed with the certification authorities, otherwise the software mail fail to gain approval when the extent of RE is revealed during the SOI audits.

The case studies confirmed the importance of involving SMEs in the RE process. SMEs provide the expert knowledge of the domain, required system behavior, software architecture, and source code that might otherwise be missing in an RE environment. It is unlikely that a single individual will have all the required expertise, so an RE project is likely to need a team of SMEs.

7. REFERENCES


5. Agile Manifesto (agilemanifesto.org).


APPENDIX A—CASE STUDY: ANALYSIS OF PROJECTS COMPLETED AT VEROCEL

Over the last 11 years, there were many projects completed at Verocel where certification evidence in compliance with DO-178B [1] was developed using RE.

The following data was extracted from the PR database. Only 13 projects were included, and only completed projects were analyzed. Only 13 projects were used because they were diverse. Projects from the same customer which were variants were not included as this may have skewed the results. The starting point for all of these RE projects was source code, and any other high level requirements, specifications and descriptions if they were available. These varied by project.

The 13 projects had a total of ¼ million ELOC. For the C programming language an ELOC excludes blank lines, comment lines and lines which only contain a begin bracket “{”, a “then” or “else” keyword etc. For the Ada programming language, statements correspond to semicolons “;”. For assembly code, three lines of assembly code are treated as one ELOC. Using this assembly code expansion metric, we have found at Verocel that the functionality as expressed by requirements and verified by tests, and other artifacts corresponds to the functionality of other programming languages. This metric is used to normalize the correspondence between the artifacts.

The majority of the code approved was level A.

A.1 PROBLEM DETECTION METHOD

As each problem is reported in the database, its severity is recorded. Level 1 errors are those that must be fixed. Level 2 errors are those that should be fixed; if not fixed, additional verification or justification is required to mitigate the problem. An example would be a square root function that returned a wrong value if called with a negative number. If analysis showed that only calls with
positive numbers are made, then the code need not be fixed (although a recommendation would be made that it was). Level 3 errors include comment errors. They should be fixed but need not be. The PR remains open until the code is changed which would draw attention to anyone attempting a change to the source file. Level 4 errors are typically insignificant and include formatting errors; for example, indentation not being compliant with the coding guide.

For the analysis carried out below, only level 1 and 2 errors were considered, which are approximately 50% of the total set.

PRs include information describing how the problem was found:

1. **Analysis** – manual inspection of any of the life cycle data, typically performed when developing a MAL using a LAL, e.g., finding errors in the source code when developing LLRs, or finding errors in the HLRs as they are being developed through RE using the LLRs.

2. **Observation** – manual inspection not related to the specific artifact being worked on. For example, while developing requirements for one component, an error could be observed on a different but related component.

3. **Beta Test** is synonymous with Functional Test.

4. **Functional test** – requirements-based test. Problems are usually found while the tests are being developed or if some different but related software component is changed and the change affects the function being tested.

5. **Coverage Analysis** – Faults found during the coverage analysis process.

6. **System Test.**
The database included 2,064 level 1 and 2 PRs. As can be seen, the 77% of the problems were found by engineers using engineering judgment (analysis+observation+review), and 23% or the problems are found by automation (beta test+functional test+coverage analysis+system test). More importantly, 54% of the errors were found during the manual RE development effort (analysis).

Even when reverse engineering, the majority of the errors are still found through intellectual effort.

It should be noted that typically, the projects that Verocel undertakes for RE are software-based, where the software has been developed and is reasonably robust. Informal testing has been completed, but is not requirements-based because the LLRs have not been developed. The complete test suite is developed after the LLRs are written.
A.2 PROBLEM TYPES

The PRs are classified as they are reported. The following problem types were recorded in the database.

![Problem Types Pie Chart]

Documentation errors are errors in user manuals and other descriptions and specifications.

Modified Functionality – These are problems that identify errors in the intended behavior. They will result in changes to the requirements and the design, the code and possibly the documentation. These are always checked by SMEs.

As can be seen, most of the problems are raised against the source code. Requirement errors represent 12% of the changes required. Requirement errors include imprecise, ambiguous, or incorrect system requirements or HLRs presented with the source code.
A.3 CODE ERRORS

Considering source code errors in isolation from the other types of error:

![Code Errors Diagram]

**Figure A-4. Example RE projects, Source Code Errors**

As can be seen, 75% of the errors in the source code are found using manual processes (analysis+observation+review), and 25% are found using tools and technology (beta test+functional test + coverage test + system test).

The striking observation is that the most prolific error detection method when performing RE is engineering judgment, which is performed as a development process and producing LLRs and establishing the traceability between LLRs and source code.

Even though informal testing has taken place, a rigorous review of the code with sufficient detail to be able to develop LLRs reports more requirement errors than any other method.
A.4 COMPARISON WITH FORWARD ENGINEERING

Capers Jones [11] has captured a wealth of statistics on software productivity and quality. As a comparison between the RE process at Verocel and the Jones gathered statistics, consider the following.

The code delivered for certification to Verocel is complete, it runs, and it has had debugging performed on it. Verocel’s task is to reverse engineer the certification evidence, not debug the software.

Jones cites four severity levels. For a typical C program of about 12,500 lines of code, the distribution of the errors is shown in figure A-5.

![Figure A-5. Forward Engineering Severity Levels](image)

This shows that the two top levels (critical and significant) are 13% of the total, whereas at Verocel the distribution is 50% for level 1 and 2 errors compared to all of the errors.
Jones describes some defect statistics using the following two terms:

- **Defect potential** – the total universe of errors or bugs that might be expected in a software project.
- **Defect removal efficiency** – The percentage of potential defects eliminated prior to releasing a software project to customers. It is usually measured by comparing the volume of bugs found prior to release with the volume of bugs reported by users in the first year of deployment.


![Figure A-6. Forward Engineering Potential Defects per Thousand Lines of Code (KLOC)](image)

The above figure describes the number of defect potential faults found for DOD software. In this Forward Engineering compared with figure A-3, many more errors are found during the requirement and design verification processes compared to the coding verification processes.
This should not be surprising, as requirements and design reviews are done before coding is started. Based on a 13\% ratio of Level 1, 2 errors, the total shown is 9.9 defects per KLOC. This is slightly higher than what is normally expected at Verocel. Over these projects, we averaged eight level 1 and 2 faults per KLOC.

![Pie chart showing residual defects per KLOC]

**Figure A-7. Forward Engineering Residual Defects per KLOC**

Assuming a distribution of just 13\% of these bugs being severity 1 or 2, the chart above shows one (1) residual defect per KLOC after delivery, reported within a year.

Using an RE approach, for the projects used in this analysis there were no delivered defects reported against the software (there were a few minor defects that were reported, but by analysis were deemed inconsequential and were documented at the time of delivery).
APPENDIX B—SURVEY RESULTS

B.1 SURVEY RESPONDENTS.

This analysis captures many facets of the RE research. The RE survey was used to analyze the respondents, their work and attitudes to RE. The survey captured opinions from Certification Authorities and DERs, as well as practitioners or managers of the development of aviation-based software at various software levels. The approach and attitudes to certification were probed especially in the context of RE.

The DO-178B [1] RE survey was distributed in July 2010.

The surveys were sent to several groups and lists:

<table>
<thead>
<tr>
<th>LinkedIn Groups</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DO-178B – Standard for safety-critical software</td>
<td>660 members</td>
</tr>
<tr>
<td>DO-178b’ists</td>
<td>1,120 members</td>
</tr>
<tr>
<td>Safety Critical Professionals</td>
<td>930 members</td>
</tr>
<tr>
<td>DO-254</td>
<td>43 members</td>
</tr>
<tr>
<td>DO-178B and DO-254 Programmers Group</td>
<td>162 members</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e-mail lists</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Verocel Internal list</td>
<td>380</td>
</tr>
<tr>
<td>FAA List</td>
<td>400</td>
</tr>
<tr>
<td>SC-205 list</td>
<td>Approx. 300 sent</td>
</tr>
<tr>
<td>Total DO-178B potential survey candidates</td>
<td>3,995</td>
</tr>
</tbody>
</table>

| Table B-1. Distribution of the RE Survey |

It should be noted that there is a large potential overlap, where an individual could be on several or all eight lists.

The objective was to reach certification practitioners and DERs, as well as developers, whose opinion is based on experience and knowledge of the research topic being addressed. Analysis of the results indicates a high level of experience with certification issues was represented by the certifiers and their representatives as well as developers. While the sample over the entire industry may be small, the response by senior participants is appreciated.

The surveys were anonymous when presented for analysis, although each was tagged with a numeric identifier to maintain traceability throughout analysis. The charts were labeled [1.0, ACK] corresponding to individual sheets, which also correspond to the column identifiers in the master sheet, and serve no useful purpose other than maintaining traceability to the original data.
There are many more practitioners in industry than Certification Authorities and DERs. It may appear that the Industry was under-represented, but the expectation is that the mailing lists used from the FAA and other sources had a higher proportion of Certification Authorities and DERs than would have been expected from a random population of people involved with the DO-178B based industry. The mix is still considered fair and provides a good cross section of people whose opinions were sought.

Figure B-1. Survey Respondents [1.0]
On average, the respondents were very experienced. Responders had more experience as practitioners than as managers, (the first four questions are approximately 4 years shorter than the subsequent four questions). This is not unexpected, as managers often have a development role as well, especially on smaller projects.

The average experience level of the respondents confirms that the opinions were submitted by people whose opinions should be valued.
The question asked was “What size of project teams have you been involved in? This includes all the people needed to produce software for an aircraft or aircraft related function?” (select all that apply).

The count is the numbers of respondents that responded “yes”. It is very surprising to see so many projects with team sizes of greater than 21. On reflection, the question may have been misinterpreted. It was not clear from the question if the project team included the entire team including system, hardware, and software engineers, or if they were to include software only, as this survey was software related. Taking the ranges and number of respondents into account and assuming that the largest project was not greater than 100, the average project size is 15.
The levels for the projects was surprising, as the general “feeling” was that there would be a lot fewer Level A projects than Levels B and C projects. Further scrutiny of the data showed that many people worked on more than one project. Of the people that worked on Level A, 28% worked on Level A only, and 59% worked on Levels A and B. Of the people that worked on Level B, 57% worked on Levels B and C. Note that the chart does not show how many projects were being worked, but a “Yes” is scored even if there is only one project worked by an engineer at a specific level. The chart indicates the use of RE at the various levels and not the number of projects at each of the levels that use RE.

As there is less lifecycle data developed at level D, it is quite plausible that less RE would be performed and source code would not be used to generate requirements as this introduces additional detail that may be unnecessary.
The survey respondents were predominantly from the commercial areas, (Part, 23, 91, and 25 aircraft). Military and Rotorcraft were well represented. Ground-based aviation systems were a welcome addition, as were the engine certification. The surprise was UAS. The expectation is that while UAS do not have a formal certification process yet, many DO-178B practitioners are involved with UAS, and DO-178B artifacts are developed even though not mandated.

The chart shows that the majority of the projects (60%) receive a TSO authorization.
The level of respondents experience with projects that have been approved was high. Two thirds of them had between 6 and 50 project approvals. Only one (1) percent of the projects were not approved.
There are several interpretations of Reverse Engineering in Industry, which of the following statements would you consider true: (select all that apply)

- Starting with source code and some documentation, create all required documentation, and perform verification to create verification records
- The source code is reviewed as means to improve the design and requirements
- System or software high-level requirements are used to create source code in an informal manner to obtain advance visibility of issues and then the other documentation is created to fill in the...
- A third party receives requirements, design, and source code documentation and performs verification to create the verification records
- Starting with only the source code and no other material and producing all the other life cycle documentation

Figure B- 8. Interpretations of RE [AT-AY]

RE as perceived by the respondents aligns closely with the proposed definition. Most of the respondents believe that using source code as a starting point for development of other development related artifacts (requirements and design) constitutes RE.

Reviewing source code to improve the design and requirements was not considered RE, presumably because “review” is a verification activity and RE is a development activity. If requirements are used to develop source code, or requirements, design and source code are provided to a third party and then verification processes are used, then this is not considered RE except by a few respondents. The respondents are reasonably unanimous on their views and interpretations of RE.
The Question asked was “Would you say the written FAA policy and guidance is ample, barely sufficient, or insufficient in providing you with the information needed to see approval of RE Projects?

The question was asked of the Developers. The responses were very polarized; either Ample or Insufficient. There were 16 responses who claimed the policy and guidance are insufficient, and there were 15 respondents who answered that Cert Authority/DERs required Interpretations, Constraints or additional life cycle data (question Q15 in figure B-14). This lines up very closely.
This chart shows a strong bias to the approval process used by the FAA. The approval mechanisms appear popular, but the interpretations of RE are questioned.
The Responses from the Cert Authority/DER who were involved in RE projects, indicate that most projects are approved (16% are rejected). The percentage of rejections is skewed by one “outlier” result. While most respondents answered “none”, one respondent rejected one project, two rejected two projects, and one rejected three projects. One respondent rejected 12 projects that used RE. This particular result is being discarded, as there is a strong likelihood that this was a data entry error.

When asked of the Cert Authority/DER if they would accept a reverse engineered project provided it satisfied all of the requirements of DO-178B, 53% replied “YES”. A similar question asked of the developers but reversed, yielded 62% acceptance. See Figure B- 14.

The Cert Authorities/DERs were twice as likely to use the CAST-18 position paper as the developers were. Documenting RE processes in the PSAC featured very high in both the Cert Authority/DER responses as well as the Developers.
This question was asked of the Engineers. Unfortunately, the question was not precise enough. The intent of the question was to explore the efficiency of the RE processes in finding errors in the software. After analyzing the results, it is now unclear if the question is related to the RE processes themselves or the software to which RE is being applied.

The results show that whichever interpretation is assumed, the “same number of problems” dominates, and the difference between the “greater number of problems” and “Fewer number of problems” is only 13% of the total number of projects in the responses.
Developers: Have you been involved in a project which used reverse engineering?

Level A: 40%
Level B: 33%
Level C: 10%
Level D: 17%

The respondents were asked to consider a specific project when responding to subsequent questions. Each respondent could select the one project from different levels. The proportions of the levels above do not represent the number of certifications by level, but it does give a basis for considering the responses in figure B-14.
Only four percent of the respondents claimed that the use of RE was rejected by the Certification Authorities. This correlates very closely with the responses in figure B-11. Most projects had access to domain experts while performing RE. Most projects documented the RE process in the PSAC. The CAST position paper was only used in about a third of the projects, and for 38% of the projects, the certification authority required interpretations, constraints or additional lifecycle data beyond those described in DO-178B.

This is a troubling component. Guidance should be clear and not require special interpretations. This means that either the developers produced certification data that was not acceptable, or the Cert Authority/DERs did not accept what was produced and required more work. On a positive
note, the Cert Authority/DERs accepted 62% of the projects that provided all of the required DO-178B evidence.

![Pie chart showing 68% 'Yes' and 32% 'No']

**Figure B-15. Engineers – Have you used RE [BZ]**

The question asked of Engineers was “have you ever used RE on one of your projects that have been approved, or been proposed for approval, under either type certification process, supplemental type certification process or TSO?”

As can be seen, two thirds of the respondents have used RE in their projects.
The use of RE is very prevalent in the industry. Practitioners were asked about the type of RE used. Performing RE from source code or object code was reported by 91% of the respondents. This corresponds to the responses to the questions on processes used shown in figure B-17 which shows that 11% of the LLRs are not developed from source code, meaning that 89% are. It was surprising to see that 70% of the respondents develop HLRs using an RE approach. The numbers correspond to those shown in Figure B-17 where 68% of the respondents said they were reverse engineering HLRs from source code and 69% from Design LLRs.

An iterative process, which develops code from draft versions of the requirements followed by subsequent refinement, is used by 83% of the respondents.

Between 67% and 80% of the respondents used RE to supplement information developed, either on new projects or on previous projects that were being adapted.
Only 50% of the respondents used RE on COTS products. This number is lower than the developed software, but this could be because COTS manufacturers sometimes provide certification packages for the software. This software represents a part of the embedded software used in aviation-based systems and not all of the respondents may be involved with the COTS components.

Figure B-17. RE processes used [BQ-BT]

This question probed how much RE was occurring and during which development phases. The responses asked for simple categorization, none, less than average and more than average.

The development of HLRs from source code and from design specifications was almost identical, and 60% of the respondents admitted to some level of RE of HLRs. This is surprisingly high, because extracting the black-box behavior to the external interfaces of the software by analyzing the software is surprisingly hard, and therefore expensive.
A surprising 17% of the respondents claimed that they did not develop LLRs from source code. Moreover, a third of the respondents claimed that over 50% of their LLRs are developed from source code.

Unsurprisingly, 90% of the respondents claimed that they did not develop source code from machine code. A small number of respondents do develop source code from machine code. The most likely explanation is that they may obtain software libraries and hardware interfacing code and convert this to Assembly Level, which would then be treated as source code. The question was not precise enough to validate this information, so it remains an assumption.

Figure B-18. Use of Other technologies. [BU-BW]

The charts indicate what percentage of respondents that responded to this question responded with “Yes”.

For the Agile Development question, note that the question refers to the development process only. Agile Engineering is characterized by evolving requirements, design, implementation and testing incrementally through the software evolution. This process is used successfully when the system requirements are not firm and when the behavior of the software reflects directly in the behavior of the system. The end user can then observe the behavior and help with the refinement of system requirements. This process finds system level errors early and requirements, design and source code and tests are modified together. In practice, it is often the case that the requirements process and testing become less formal and full traceability is not managed well. As a result, RE plays a key role in recovering the information that did not evolve with the software.
The use of Model Based development and Object-Oriented Methodologies in conjunction with RE corresponds to the level of RE use in general, as shown in figure B-18.

Response to the question on use of tools for configuration was surprisingly low. The expectation is that most projects use some form of configuration management system to maintain baselines. It may be that the respondents did not consider them to be tools used specifically for RE thus the response is lower than expected.

Execution simulators are often used when target hardware is not available, or when it is easier to generate data using a simulator than manually.

Use of tools to analyze object code was small. Interestingly, web searches for RE tools provide a large number of papers, research materials, and analysis on tools used to re-engineer information from object or machine code. This is a different type of RE, one that focuses on design recovery for maintenance purposes, or for the use of code path recovery typically used to find the location of the code that checks license keys. There is very little material on the RE of requirements from source code.

There are tools that provide call graphs, document set/use for data values, and document parameter use. These tools are used to help document the code and use this to capture higher-level representations. These tools are an accurate labor saving mechanism, as it is much more
convenient to extract such information rather than develop it manually. 45% of the respondents said they use such tools.
APPENDIX C—ANALYSIS OF ISSUES AND THEIR POTENTIAL MITIGATIONS

A number of issues have been identified as problems that could be potentially created by the RE process. These are issues that have been gleaned from CAST position papers, literature, the survey, and experience of the authors. These have been summarized below along with how they can be mitigated by following the principles provided in this report.

Numbers, other than CAST Position Paper numbers, are only used to make reference and discussion easier.

C1. LACK OF ASSOCIATED BEHAVIORAL SPECIFICATIONS

Issue Description

If there are no associated behavioral specifications (e.g. system requirements or HLRs) then there may be many solutions for requirements from source code.

Potential Mitigations

It is important to distinguish between real system intent vs. perceived system intent developed using the RE processes.

Using RE, source code may be analyzed in an attempt to discover the intended behavior of the system. For all but the simplest of systems, various assumptions will need to be made during this process. If these assumptions are incorrect or inconsistent then the ‘discovered’ intended behavior may be incorrect or inconsistent. The RE approach is performed by engineers whose experiences, knowledge and skill levels influence the decisions they make. This may introduce some variability in the statement of intended behavior that is inherited from the source code up through the requirement levels. The variability of this intended behavior is reflected in system requirements or HLRs when reverse engineered using this process.

If the system is sufficiently simple, then the source code representation may be a very close mapping to the intended behavior. In this case, the abstractions level between source code and intended system behavior is small and few assumptions need to be made. This means that the number of different interpretations is small and the System level requirements may be accurate.

More complex systems may have mappings that are not so obvious, and more decisions need to be made. The transformation levels between the different representations may be large, making the transformation more error prone. The number of transformation levels may be increased by increasing the number of abstraction levels. While this makes the individual transformations simpler thus less error prone, there are more of them so the opportunity to make errors is increased.

As RE is a manual process (albeit one that may be supported by tools), the knowledge, experience and skills of the engineer will have a direct effect on the number of assumptions made during the development of the information in the abstraction level. The lower the number of assumptions then more accurate is the mapping from the LAL to the MAL.
While engineers familiar with the intent of the system may make assumptions which correct the MAL in the presence of certain errors in the LAL without changing the LAL, or they may restate the intended behavior in the MAL after changing the LAL. The range of abstraction levels spans the system requirements and HLRs to the source code. It may be that the systems engineers are not skilled software engineers and the software engineers are not skilled system engineers. This may result in assumptions and communication difficulties. This problem is not specific to RE during the development process.

In a more abstract way, this is stating that there may be more functionality in the LAL than expected in the MAL – are the LAL elements necessary?

C2. ERRORS IN LAL MAY PROPAGATE TO MAL

Issue Description

The LAL may contain errors such that the RE would create erroneous descriptions for MAL.

Potential Mitigations

Follow independent verification process that checks MAL descriptions against the LAL.

The less abstract level may contain errors which are synthesized in the MAL using an RE process. On the other hand, if a waterfall process was used the errors may be inherited in the LAL from the MAL. The presence or absence of errors in not the issue. The real question is – is there a different class of error inserted by the RE process which would be less likely in the forward engineering process?

Certain classes of errors may be found during the RE process. If the error is one that demonstrates inconsistencies between software constructs then this could be found, e.g. a buffer declared with boundaries between 0 … 9, and a loop that indexes from 1 … 10 is a software error that should be found during the RE process. This problem should also be found during the verification processes especially during code review.

A straight translation without consideration to correctness with the expectation that subsequent verification steps will find translation problems looses an opportunity to find errors. Part of this process step should be a requirement that engineers are vigilant for possible faults in the LAL.

During the source coding and debugging process, engineers often used additional code to output internal states that are not visible at the boundaries of the system level. Typically, “print” functions are inserted, to provide a history of the execution states. To obtain specific values, additional intermediate states may be added which consolidate internal information. This debugging technique is often controlled and special compiler directives are added so that this code can be conditionally included or excluded from the operational code. As this is a human process, mistakes can be made and the resulting trace code may be left in. Function calls to reporting routines are easy to notice, but internal data collection may not be so obvious. This may result in additional code, which should be treated as dead code. If RE is used, then this code may have LLRs written for it, which may have HLRs written for it. Subsequent verification steps will review the requirements, and the code against the requirements, followed by requirements
based tests that will verify the code against the requirements. The dead code will no longer appear as dead code. Fortunately, code that is inserted to record internal state does not change it (otherwise, it would defeat its purpose). This should be found and addressed in the verification processes. These trace calls and their corresponding data manipulations are usually simple and are often accompanied by source code comments. (Note: this is something that should be recommended for the coding process especially if RE is used – trace code must be surrounded by compiler conditional directives or comments so that they are noticed as extraneous behavior that should be removed).

In a more abstract way, this is stating that there may be errors in the LAL / MAL translation – Is translation between the LAL and the MAL correct?

Note that this correctness does not depend on the direction; it is required whether forward or backward translation was used.

Self-checking correctness is easier to notice as it is manifested through the lack of consistency in behavior. Code reviews should be used and should find indexing errors, deadlock problems (where resources are locked out of sequence and progress is held up until resources are released), overflow/underflow arithmetic errors, loop termination errors, etc.

The key point here is that errors can creep into the system and processes should be formulated to which detect these errors with the appropriate level of confidence. In some cases, the mitigations of these errors are independent of direction, and some are unique to RE (e.g. the importance of identifying a qualified SME).

C3. MISSING FUNCTIONALITY IN THE MAL

Issue Description

The LAL may not contain entire solution to a system where it is used, e.g. there are functions missing from the LAL that should be in the MAL.

Potential Mitigations

Need to emphasize that going bottom up can’t guarantee that all of the desired features at the system level will exist. Some means of introducing real system intent, and establish completeness of the reverse engineered perceived system intent.

There may be several abstraction levels that are ultimately represented by source code and then the final operational system. The traditional expectation is that with a formal waterfall model, an engineer is given an abstraction layer and will translate this to a LAL using only information provided by the MAL. In practice, engineers will often ‘refine’ the translation by adding detail some of which is necessary and some of which is unnecessary functionality. The MAL may not specify certain robustness checks, yet by coding conventions, tradition, or experience constructs that perform null pointer checks, index boundary checks or consistency checks may be added to improve the robustness of the code.
It should be a policy that when RE is performed, all robustness checks must be called out as LLRs. This would ensure that they are considered in the testing process, the source to LLR reviews, and propagated up in line with DO-178B [1] objectives.

There are several reasons why there may be missing functionality as represented in the source code:

- The source code is incomplete through omission. During the software development process, some combinations of parameters may be coded and some level of testing performed to check the implemented functionality. The functionality may be incomplete, but the testing is performed on the parts that are written so far. The engineer may fully expect to return to the coding process to complete the functionality once a level of confidence is built up with the code written to that point. Well-organized software engineers will often annotate the cases that have not been completed as null code statements with appropriate comments. This draws attention to the incomplete behavior. This may not always be the case, and it may be difficult to detect if the required functionality has been fully implemented, or if the implementation has skimped on the required code paths.

- The interfaces may have changed after the implementation of the source code that uses them. An example of this may be when an enumeration type was first written as [RED, GREEN] and the code written to respond to a variable of this type. If the enumeration is then changed to [RED, GREEN, AMBER] then if the code is not changed then the response of the system when the value of AMBER is undefined. This additional behavior is missing in the software, and an RE process that does not include analysis of the interface specifications will result in incomplete LLRs. This may propagate up through the levels of abstraction and may end up with a higher-level requirement that lacks the detail necessary to detect the missing functionality.

Actually, this is not a problem for our RE. We have an oracle in this case – the interface specification. What if the system needs an amber (e.g. caution) and only warnings and “good” indications are coded. In this case, a good SME would immediately recognize that the system needs this extra functionality. So how would such a mitigation appear?

“The SME shall evaluate the reverse engineered functionality to ensure it contains all of the functionality needed to perform the SME’s level view of the system functions”.

However, consider that there may be several levels of interface specifications. Some are provided at the System level and would be understood by the SME. Other interfaces are local to the implementation (e.g. header files in the C programming language), and the SME may not understand the programming language and the interfaces at that lower level. For example, specifications of internal data structure formats. If the data structure is used consistently with no assumptions on the layout at the implementation level then the checks of the interfaces may be performed at the local level. For example – a data structure format could be used to represent a double-linked list that holds a value at each list element. The record could contain the components Next, Prev, and Value. Source code that used the Next and Prev by name would not
be affected adversely if the record type was changed to Prev, Next and Value. If some of the software functions assumed that Next was the first component in the structure, then the software could fail if the record type in the interface specification was changed.

This internal structure may not be visible at the level at which the System SME would notice. The problem is the same whether forward engineering or RE is being performed. Interface specifications are not typically reverse engineered, however it is important that as RE uses information flow from the LAL to the MAL, interface specifications may be used to produce higher levels of abstraction and may result in descriptions of behavior that may not match the system intent.

Assumptions may be made about the functionality of components that are integrated. For example, function ALLOCATE should return a location in heap memory which contains zeros. There are many ways of implementing this. Let’s consider three:

1. ALLOCATE obtains the required memory area and sets the contents to zero before returning a pointer to it.
2. The heap memory is set to zero, ALLOCATE obtains heap locations and de-allocations are not possible.
3. The heap memory is set to zero, ALLOCATE obtains heap locations and de-allocations cause the de-allocated memory to be reset to zeros.

The implementation of the ALLOCATE function may assume implementation 2 or 3. In this case, it would not contain the additional functionality to set the memory locations to zero. This functionality would be considered missing, if the intended behavior was option 1, but the assumed behavior was 2 or 3.

In a more abstract way this is stating that that there may be more or less in the LAL than expected in the MAL – are the LAL elements sufficient?

**Sufficient, NEcessary, CORrect** may be remembered by the word SUNECO.

C4. LACKING GUIDANCE FOR CERTIFICATION AUTHORITIES

*Issue Description*

The certification authorities currently do not have guidance that would allow a consistent evaluation of RE projects, which would result in over conservative interpretation and easily rejected applications

*Potential Mitigations*

The suggested guidance in the report needs to have the following attributes:

1. Addresses all the issues raised herein including all concerns in current published regulatory material (e.g. CAST position papers)
2. Provides rationale.
3. Will result in acceptable levels of variation in its application.
4. Is acceptable and defined by the certification authority.
5. Can be used directly by certification authorities.

This could be addressed through a short RE Job Aid that can be used when RE is called out in a PSAC. This will require collaboration with the FAA.

C5. INADEQUATE PROCESS

Issue Description, CAST Ref 4.1, 4.7

The RE process is not described or poorly described in the process documentation. For example RE is used but only the forward process is controlled via plans. This may mean that the certification authorities may not be aware of an RE approach before significant RE work has been completed.

Potential Mitigations

Need to distinguish between the Development and the Verification steps.

If a company has plans that describe the forward process only, then an additional ‘RE plan’ may be an appropriate place to describe the specific RE steps to be taken on this project. The project should not wait until the SAS to describe the deviations, but describe them up-front, and not require that the existing project plans be re-written. An approach may be to include a separate Reverse Engineering Software Plan (RESP) or have a separately identifiable part of existing plans as the RESP. This approach would then be described in the PSAC or PHAC, which would have the responsibility to introduce all life cycles used on the project along with the plans that are designed to support them.

If the certification authorities only see the PSAC or PHAC then it is essential that these documents describe the RE process directly or indirectly in the process plans.

If the development plans do not describe the RE process (the developer fails to develop standards, transition criteria, verification criteria etc, specific to the RE process), or if the development plans are not followed (and this is addressed in the SAS) then the certification authorities will not have visibility into the RE approach until late in the program.

A possible mitigation could be to repeat work using a RESP. The RESP need not contain the full company process planning information, only the processes supporting RE. The steps that are not changed still apply and can be referenced out.
C6. WRONG LEVELS OF ABSTRACTION BETWEEN LEVELS

*Issue Description CAST Ref. 4.2, 4.5*

The artifacts of RE cannot demonstrate the same level of technical and verification content as a more traditional process. For example, there may be insufficient abstraction between the different layers of development. An example would be where the HLRs are highly correlated in detail with the source code.

*Potential Mitigations*

The difference in the abstraction levels of the MAL and the LAL should strike a balance between the difference between each level of abstraction, and the number of levels of abstraction.

The MAL should provide a representation that specifies the intended behavior of the LAL:

1. A re-statement of the intended behavior at the same or similar level using a different notation is only useful to check that the transformation was correct, not the intended behavior.

2. If the abstraction level between the MAL and LAL is too large, there will be less confidence that the process is repeatable with the same or equivalent results.

This issue also applies to forward engineering, but there may be a temptation for engineers to develop LLRs from code that specify how the code works rather than the intended behavior. This may then result in the difference between the abstraction level of the LLRs and the HLRs to be too large.

C7. LACK OF FAMILIARITY WITH RE PROCESSES

*Issue Description CAST Ref. 4.2*

The use of RE compromises the integrity claims that could be made from a traditional forward software engineering approach.

*Potential Mitigations*

Forward engineering requires a development process where information is constructed based on given information and a creative process that has been taught or learned through experience and guidance. RE is not taught to the same extent as forward engineering at colleges and universities.

It is important that engineers performing RE processes know and understand the intent, the risks, and the potential errors that can result if the process is done badly.

What could go wrong? If a working code base is reverse engineered, the fact that the code base is working could lead to a false sense of confidence that the software is correct which might result in less thorough investigation or care in developing the RE artifacts.
Additional training, work orders, mentoring, and oversight may be required to ensure that reverse engineered information is valid.

**C8. UNSTABLE PRODUCT LEADS TO INTEGRITY ISSUES**

*Issue Description* CAST Ref. 3.0

The object being reverse engineered is not a stable product, does not have an acceptable documented service history, and has not been developed to a standard that is comparable to DO-178B thereby having lower ability to demonstrate integrity assurance.

*Potential Mitigations*

The integrity assurance should be demonstrated by the Verification processes. We should be able to demonstrate that an unstable development process is fine providing it is understood and allowed for, e.g. Agile development will have many iterations during development. We must know and understand the intended states of all of the components.

If service history is used to obtain certification credit, then this should be addressed in guidance specific to service history.

Stability of the product will be discovered during the RE process. If the product is unstable, and many changes to the code base are required then the costs become very high. Developing requirements to software that is being updated uses up a lot of resources as work is repeated and through the spread of impact, the problem becomes larger and unmanageable. If the product is unstable, then the RE processes should be stopped and informal debugging steps should be taken before continuing the RE processes.

**C9. RE PROCESS IS NOT VIABLE**

*Issue Description* CAST Ref. 4.2

There is no justification of the viability of the RE process.

*Potential Mitigations*

The viability has been demonstrated on a large number of projects. 63 out of 162 respondents to the software survey reported that they had been involved in a project that used RE. The respondents were very experienced. Only one of the RE projects was not approved. There is a preponderance of use of RE use in the industry, and most of this has satisfactory conclusions. The viability of RE has been demonstrated.

**C10. ABSENCE OF SUBJECT MATTER EXPERTISE**

*Issue Description* CAST Ref. 4.3, 4.4

There may be no or inadequate resources (e.g. SMEs, developers, descriptive documentation) available to establish a given LAL, the basis for design decision, the intent of implementations,
the intended problem space that the LAL was intended to solve, etc. This would make it difficult to verify any MALs developed from the LAL.

Potential Mitigations

It is absolutely necessary to have access to SMEs. A single individual is unlikely to have all the required expertise. Typically, a group of experts will be used to ensure the necessary system domain knowledge and software expertise, particularly if the system is complex. A requirement is proposed that an organization identify such experts to the authorities.

C11. AIRBORNE SYSTEM REQUIREMENTS CANNOT BE TRACED DOWN

Issue Description CAST Ref. 4.5

Airborne system requirements cannot be correlated to the reverse engineered MAL (e.g. HLRs)

Potential Mitigations

The correlation between system requirements and HLRs depends very much on how the requirements at these two levels are written. Some of this correlation depends on the terminology used. For example:

System Requirement 25: Reverse Thrust shall be inhibited if there is no Weight on Wheels.

High-Level Requirement HR1.7: Discrete REVERSE THRUST shall be set TRUE when BIT 7 of ADDRESS X’2F33 is ZERO.

In the world of the system engineer, the system requirement may make sense, but in the world of the software engineer, there have been a number of assumptions made which need to be resolved. In a forward engineering process, the transition from the system requirement to the HLR would be a large transition because information was missing:

1. How is Reverse Thrust represented? (Discrete voltage on a pin?)
2. Is voltage High TRUE or FALSE?
3. Does TRUE represent INHIBITED?
4. Is Weight on Wheels represented by a bit value that can be read directly from Direct Access Memory?
5. Does the bit set to Zero represent Weight on Wheels?
6. How quickly will the bit settings represent the real world conditions?
7. Is there a possibility of a bounce?
8. Etc.
The system requirements may be underspecified, and the HLR developer would be forced to request more details before the software could be written.

In an RE process, the fear is that the HLR developer could assume that the software is correct because it appears to be working, and simply documents what the code is doing. When traceability between the system requirements and HLRs is established, then again, assumptions could be made that the requirements are correct and complete.

The HLR review must not make the same mistake that the HLR developer made. Reviewers should ask themselves if there is enough information in the system requirements and any other documented specifications to correlate to the HLRs.

In a forward engineering development, the translation of system requirements to HLRs is developed by a developer and reviewed by a verifier, i.e., information flows from the MAL to the LAL during development, which is checked by review. In a RE development, the information flows from an even lower level LAL to the LAL during development and the information flow from the MAL to LAL is checked by review.

Developers using an RE process should be trained to ask the same questions when establishing traceability between system requirements and HLRs, and fail the HLR if the requirements cannot be traced because information is missing or is incorrect.

C12. TRACEABILITY GAPS

Issue Description CAST Ref. 4.5

Forward and backward traceability is not fully established in RE projects. This could result in traceability gaps. This can also make it difficult to determine whether specific functions at one abstraction layer represent unwanted or undesirable functionality (e.g., dead code)

Potential Mitigations

When using RE, the traceability links are part of the development process and are established from the LAL to the MAL. The verification process should be independent, and should confirm the correctness of the MAL to the LAL.

Traceability records relationships between objects at different abstraction levels. If these relationships are based on incorrect assumptions, then review of the objects should fail. The reviews of the objects at the different abstraction levels should use information from the MAL when checking the LAL.

The developer establishing the traceability should use the information from the LAL to establish a traceability link to the MAL. The link is bi-directional. This will be confirmed during subsequent review processes.

The forward engineering process could also make assumptions and information could be missing when traceability is established. The verification process should find this.
If a large number of these traceability problems are found during the reviews, then this could indicate a deficiency in the development process, which results in the traceability being reviewed. The development process would need to be corrected.

C13. LACK OF CORRESPONDENCE BETWEEN LEVELS

**Issue Description CAST Ref. 4, 5**

Employing RE concepts in conjunction with more traditional (e.g. forward engineering approaches) can result in MALs that don’t correspond to LALs

**Potential Mitigations**

Lack of correspondence will be discovered during independent Verification between the levels.

If the MALs do not correspond to LALs, then this should be found during the review process. If the problems are pervasive, then this would indicate failures in the development processes. These would need to be corrected. This problem could occur during forward engineering, RE, or a combination of them.

C14. MISSING FUNCTIONALITY

**Issue Description CAST Ref. 4, 5**

The reverse engineered product may not contain all of the operational and safety functionality required at the airplane system level. This may result in imprecise traceability between the software engineered product artifacts and the system and safety requirements.

**Potential Mitigations**

The absence of operational and safety functionality should be found during the reviews of the system requirements and HLRs.

Traceability is already covered by the sections C11, C12, and C13.

Correct Traceability should be established by the developers. If this cannot be established then the software developers should resolve this with the systems and safety engineers. Verification of the traceability should confirm that the traceability is correct and complete.

C15. DERIVED REQUIREMENTS MAY BE MISSED

**Issue Description CAST Ref. 4, 5**

The existence of derived requirements may be the result of not knowing the purpose of a specific element of an artifact that is an input to the RE process.

**Potential Mitigations**

As all derived requirements must be resolved, then these should be discovered.
A specific element may be inserted through an assumption and this assumption may be correct or incorrect. If an assumption is correct but is not documented then this is also a problem. At the lowest level that is manually reviewed (source code), code fragments may be found that may be the result of coding, design or requirement errors. If an RE process is used, then the code insertions may result in design and requirements at the MAL that correspond to the code errors.

Some code insertions may be valid, even if they do not have corresponding requirements. For example, an operating system function that expects a task pointer as a parameter may check to verify that the object actually passed is a task object, and the user has not passed in a semaphore or message object by mistake. Such checks verify consistency of the code and are robustness checks. Other checks may be more subtle, and the effects of their outcomes may propagate to the system interfaces, by perhaps causing a system state change (like a reset).

Any artifact that is added at any level must have its behavior documented through a requirement. The requirement may be traced to expected behavior at a MAL. If this cannot be accomplished, then the requirement must be documented and treated as a derived requirement in accordance with the requirements of DO-178B.

Derived requirements may be the result of a forward engineering or RE process.

**C16. MISSING INTERFACES**

*Issue Description CAST Ref. 4.6*

There will be missing interface descriptions that result when MAL are reverse engineered from LALs. These interfaces could be software-to-software interfaces, hardware to software interfaces, hardware-to-hardware interfaces, or the more abstract function to system interfaces.

*Potential Mitigations*

If descriptions are missing then they must be added. They could be the result of forward engineering and making assumptions, or RE and not questioning assumptions.

Information will be missing during any MAL to LAL review process. If such missing information is pervasive, then this will indicate a poor forward engineering or RE development process.

**C17. MISSING INFORMATION**

*Issue Description CAST Ref. 4.6*

User guides and porting guides may not completely describe the functions and protocols required.

*Potential Mitigations*

This may happen during the forward engineering or RE process. The review process should fail if information is missing.
C18. PLANNING FOR RE

Issue Description CAST Ref. 4.6

Certification authorities not being involved early in the planning process for RE projects can result in significant rework and the associated introduction of additional errors. This will be particularly true if the product has inadequate requirements, design, traceability, and verification documentation.

Potential Mitigations

A RESP should be developed early, or the development plan should address RE, for the project and provided to the certification authorities for review along with the other software plans.

C19. MULTIPLE STAKEHOLDERS IN RE PROJECTS

Issue Description CAST Ref. 4.6

Using multiple stakeholders in RE projects can result in errors being introduced due to inadequate requirements, design, traceability, verification documentation, communication gaps, configuration management shortfalls, and quality assurance shortfalls.

Potential Mitigations

The requirements for co-ordination of the stakeholders should be described in project documentation and followed.

Using multiple stakeholders may have advantages in the development and certification of software. However, errors may be introduced whether there are many stakeholders or few. The following criteria should be imposed on a multiple stakeholder project:

1. The areas of responsibility must be clearly identified
2. The processes used must be coordinated
3. The configuration management between all stakeholders must be described, coordinated, and verified.
4. Problem and fault tracking between all stakeholders must be coordinated
5. Information flow between the stakeholders must be controlled and tracked
6. The stakeholders must possess the necessary expertise to complete the processes they are responsible for
7. Communication channels between stakeholders must be unencumbered.
8. Etc.
Multiple stakeholders with expertise in their specific areas may make a better and safer product, provided their effort is performed in accordance with the criteria listed above.

Errors and failed projects using multiple stakeholders often result from inadequate training or knowledge of the DO-178B processes to be used, or inadequate DO-178B processes imposed on the project.

C20. ADDITIONAL FUNCTIONALITY REQUIRED BY SYSTEM

Issue Description CAST Ref.

The situation may exist where the Systems SME may not be aware of all of the desired functionality. The RE process may result in desirable MAL functionality that is discovered by the RE of the LALs and a process for the recognition and validation of such functionality should exist.

Potential Mitigations

The RE process should ensure that the additional functionality is labeled as derived requirements and passed through to the System Safety assessment process.

A system may be sufficiently complex for an SME to not fully understand the requirements in detail at the time of requirements development.

Below is a fictitious example to illustrate this point.

Consider a system where a pulse is sent to a door latch followed by another pulse to open a door. The requirement could be stated as:

A pulse on door latch followed by a pulse on door actuator shall cause the door to swing open.

Such a system was implemented using slow actuators, and the system worked perfectly. The system is then re-implemented with a newer faster computer with a smaller more sensitive latch. During testing, the programmer finds that the latch is not releasing for long enough and the door cannot swing open. The previous latch had a greater moment of inertia, which held the latch open long enough for the door to swing open. The new latch, being more responsive, opened, but closed again before the door could be opened. To solve the problem the programmer added a delay loop so that the latch was held open for a longer time.

This new functionality solved the immediate problem but was not reported back to the SME, and was deemed to be a requirement to make the latch work correctly.

Unfortunately, the increased current in the wire connected to the latch, caused local heating which was not expected, and the heat build-up caused a fire hazard.
The problem stemmed from the addition of functionality that was not requested by the system requirements. The fix by the programmer was reverse engineered. Functionality was added not using information flowing from the MAL.
Validation of the proposed framework, through the review of the results, the performance of completeness checks, and the execution of case studies, demonstrates the applicability and efficacy of the proposed framework for software.

D1. VALIDATE FRAMEWORK

A project that had already been accepted by the FAA was evaluated and assessed against the framework. The plans applied to the standards were examined, as were the requirements and design artifacts that had been reverse engineered. They were compared with the framework presented in the Phase 1 report.

D2. BACKGROUND

GNAT is an open source compiler for the Ada programming language, which forms part of the GNU Compiler Collection (GCC). GNAT is licensed under the GPL. ARTE forms part of the GNAT tool chain. The authors of GNAT founded a company called AdaCore, which maintains GNAT and distributes a supported version called GNAT Pro.

Verocel reverse engineered DO-178B [1] lifecycle data for ARTE version 5.04a3, now known as ARTE/Cert, for use with the Wind River Systems Platform for Safety Critical version 1.8. This work was carried out on behalf of Smiths Aerospace (now GE Aviation)\(^1\), with the full support of AdaCore. ARTE/Cert provides a reduced Ada runtime environment capable of supporting safety critical Ada applications hosted on an Integrated Modular Avionics (IMA) platform. ARTE/Cert is now offered by AdaCore as part of the GNAT Pro High Integrity Edition. ARTE/Cert was accepted by the FAA for use on the Boeing 787 Dreamliner, supporting applications up to and including DO-178B Level A.

D3. DELIVERABLES

There were a small number of project-specific plans:

1. Project Profile
2. PSAC
3. Software Test Plan (STP)

The work performed by Verocel used the standard company processes, which are described in a set of company-wide plans and standards:

1. Software Development Plan (SDP)
2. Software Verification Plan (SVP)
3. Software Configuration Management Plan (SCMP)
4. Software Quality Assurance Plan (SQAP)

\(^1\) Verocel negotiated exploitation rights to use and sell Certification package, thus Verocel is able to put some of the certification materials in the public domain.
5. Software Plans Addendum
6. Software Requirements Standard
7. Software Design Standard

Other than the planning documents listed above, the main deliverables were as follows:

1. Software Requirements Specification
2. Software Design Document
3. Derived Implementation Requirements
4. Software Life Cycle Environment Configuration Index (SECI)
5. Software Configuration Index (SCI)
6. Software Accomplishment Summary (SAS)
7. Requirements Traceability Document
8. Software Vulnerability Analysis
9. A Digital Versatile Disk – Read-Only Memory (DVD-ROM) containing:
   a. Software Verification Cases and Procedures
   b. Software Verification Results
   c. PRs
   d. Software Configuration Management (SCM) Records
   e. Software Quality Assurance (SQA) Records

Finally, a number of white papers written by Verocel are relevant to this project:

1. MC/DC Coverage Analysis Using Short-Circuit Conditions
2. Control-Coupling Verification With VerOLink
3. Use of Requirements-Based Testing, Requirements Coverage Analysis, Linkage Analysis And Structural Coverage Analysis To Confirm Data And Control Coupling
4. Verification Of Traceability
5. Evaluation Of GNAT Coding Style vs. DO-178B Section 11.8
6. Certification of Elementary Functions in the GNAT/ARTE Cert Project

D4. PROCESS

ARTE/Cert is a general-purpose Ada runtime environment capable of supporting a variety of applications. There are therefore no avionics-based system requirements.

The HLRs for ARTE/Cert are based on the:

- Ada 95 Reference Manual
- GNAT Pro High Integrity Edition Ada Language Profile for DO-178B Level A/B Certified Applications
The software architecture and LLRs were reverse engineered from the source code.

The Verocel Software Requirements Standard defines the criteria for the specification and review of software requirements for a High Integrity Software (HIS) application or component. The standard distinguishes two types of requirements:

1. Development requirements describe the expected behaviors of the HIS. Development requirements may be either HLRs or LLRs.

2. Implementation requirements relate to the structure of the underlying source code. Implementation requirements are LLRs. Implementation requirements are very detailed, specifying constraints, robustness, and boundary conditions.

The Verocel Software Design Standard specifically describes how to reverse engineer design information from source code.

D5. TOOLS

Verocel used its own tools, which were qualified as necessary:

1. VerOCode. This is a structural coverage analysis tool, qualified as a DO-178B software verification tool.

2. VeroTrace. This is a requirements traceability tool, which managed the traceability between:
   a. HLRs and LLRs
   b. LLRs and source code
   c. Software requirements and test cases
   d. Test cases and test procedures
   e. Test procedures and test results

Two components of VeroTrace are qualified as software verification tools:

   a. Link Verification of Traceability
   b. Artifact Verification of Traceability

3. Test Harness. This automated test harness controls the execution of the tests and the capture of the test results. It is qualified as a DO-178B software verification tool.

4. VerOLink. This tool assists with analysis of control coupling. It is qualified as a DO-178B software verification tool.
D.6.1. SOFTWARE DEVELOPMENT PROCESSES AND SEQUENCE DEPENDENCIES

The project plans do not use the term, reverse engineering. The PSAC does use the term, re-engineering, a number of times:

1. “The re-engineering of the certification evidence will be performed using Verocel’s plans and procedures that are central to Verocel’s processes” (section 4).

2. “Where possible, the life cycle data materials are reused from prior certification packages, or produced as new development (requirements, design capture, source code, test procedures, and analysis) and re-engineered from the existing AdaCore data materials. As the use of re-engineering does not assure that the derived materials match the original materials, reviews by Verocel and AdaCore personnel verify that the derived materials adequately represent the intent of the ‘as built’ product” (section 5.2).

3. “Engineering is responsible for re-engineering the life cycle data materials from the data and source code delivered by Smiths Aerospace for the Ada runtime environment” (section 6.2.5).

4. “The software requirements for ARTE/Cert will be re-engineered from existing specifications, user documents, and the Ada Language Reference Manual” (section 6.3.2.1).

Verocel’s White Paper on “Certification of Elementary Functions in the GNAT ARTE/Cert Project” makes a number of references to reverse-engineering:

1. “We begin with the specific problems inherent to our reverse-engineering approach to software certification” (section 1).

2. “In a reverse-engineering approach to certification, the requirements for a software function can usually be deduced from the software and available documentation” (section 3).

3. “The reverse-engineering approach does not allow us to deduce a precision requirement for the elementary functions” (section 4).

Verocel’s Software Requirements Standard does not mention reverse engineering. Verocel’s Software Design Standard does make many references to reverse engineering:

1. “Verocel’s design capture is a reverse engineering activity that uses the code and any existing materials to capture the design of the HIS software” (section 1.1).

2. “Because Verocel’s Design Capture Process is performed on existing software, the approach taken is primarily a bottom-up, or reverse engineering approach. In this context, reverse engineering means that an understanding of the software components at
the lowest level is built up first, followed by and understanding of how the software components are used together to provide higher-level functionality, thus providing the building blocks that are used by higher-level functions. The reverse engineering process records this understanding in a way that a reviewer would comprehend, and provides the software design components (directories, source files and functions)” (section 3).

3. “If the LLRs are developed through a reverse engineering process, then the existing software associated with the target-dependent LLRs (and design components) has already been demonstrated to function with the target computer” (Table 6-1).

4. “Where the data does not exist, then the reverse engineering process is used to document the architecture and HLRs” (Table 6-1).

5. “For a reverse engineering project, the software associated with the target-dependent LLRs (and design components) already exists, and has been demonstrated to function with the target computer” (Table 6-1).

In conclusion, the Software Design Standard seems to deal with RE adequately, but the Software Requirements Standard appears to assume a forward engineering process.

D.6.2. COMPATIBILITY OF REGULATORY GUIDANCE WITH RE PROCESSES

ARTE/Cert has been approved by the FAA for use on the Boeing 787 Dreamliner to support applications up to and including DO-178B Level A.

D.6.3. ROLES

This report identifies seven roles:

1. SME
2. Requirements developer
3. Architecture developer
4. Source code developer
5. Verification role
6. Configuration manager
7. Quality Assurer

The Verocel plans and standards do not describe the use of SMEs. Although the PSAC does not use the term Subject Matter Expert, it does briefly describe the involvement of AdaCore personnel:

- “As the use of reverse engineering does not assure that the derived materials match the original materials, reviews by Verocel and AdaCore personnel verify that the derived materials adequately represent the intent of the ‘as built’ product” (section 5.2).

The role of requirements developer (author) is described in the Verocel Software Requirements Standard.
The role of architecture developer (designer) is described in the Verocel Software Design Standard.

The role of source code developer was carried out by AdaCore. The source code was developed using the GNAT Coding Style — A Guide for GNAT Developers.

The verification role is described in the Verocel Software Verification Plan.

The configuration manager role is described in the Verocel Software Configuration Management Plan.

The quality assurer role is described in the Verocel Software Quality Assurance Plan.

D.6.4. PROCESSES

The following Integration Definition for Function Modeling (IDEF0) diagram, extracted from the SDP, outlines the Verocel approach to the software life cycle for a project. Figure D-1 shows five major activities:

![Figure D-1. Lifecycle Overview](image-url)
D.6.4.1. Develop Certification Materials

Of these five activities, “Develop Certification Materials” is the most relevant to this case study. This activity is expanded in the following figure D-2. Figure D-2 shows four phases:

**Figure D-2. Develop Certification Materials Activity**
D.6.4.2. Establish Requirements

The purpose of the Establish Requirements phase is to establish a requirements baseline and document that baseline in the Software Requirements Specification (SRS). In general, the requirements are derived from:

- Production specifications
- Software baseline
- System requirements

The Verocel Software Requirements Standard defines the criteria for the specification and review of software requirements for a HIS application or component.

In the specific case of ARTE, the requirements were derived from:

- Ada 95 Reference Manual
- GNAT Pro High Integrity Edition Ada Language Profile for DO-178B Level A/B Certified Applications
- Software baseline

Figure D-3 provides a representation of the Establish Requirements phase:
D.6.4.3. Capture Design

The purpose of the Capture Design phase is to create a Software Design Document (SDD) that captures the software implementation decisions shaping the product. In general, the design is derived from:

- Requirements baseline
- Customer software baseline
- Product specifications, if available

The Verocel Software Design Standard defines the level and the type of information in documenting the design of HIS.

In the specific case of ARTE, the design was derived from:

- Ada 95 Reference Manual
- GNAT Pro High Integrity Edition Ada Language Profile for DO-178B Level A/B Certified Applications
- Software baseline
Figure D-4 provides a representation of the Capture Design phase:

Figure D-4. Capture Design phase

D.6.4.4. Code Review

The purpose of the Code Review phase is to review the HIS source code. When review of the source code is complete, the system may be built (compiled and linked) to form an executable software baseline.
Figure D-5 provides a representation of the Code Review phase:

Figure D- 5. Code Review phase
D.6.4.5. Verify Product

The Verify Product phase produces evidence that the HIS has been verified against the requirements and other artifacts.

Figure D-6 provides a representation of the Verify Products phase:
D.6.4.6. Analyze Results

The test results obtained from testing are analyzed against the expected results for functional tests and structural coverage analysis.

Figure D-7 provides a representation of the Analyze Results activity:

![Analyze Results Activity Diagram]

**Figure D-7. Analyze Results activity**

D.6.4.7. Assessment of Process against Framework

The requirements and design were reverse engineered from the source code, the Ada 95 Reference Manual, and the GNAT Pro High Integrity Edition Ada Language Profile for DO-178B Level A/B Certified Applications. The tests were forward engineered from the resulting requirements baseline. The approach taken was therefore compliant with the proposed framework.

D.6.5. RE ASPECTS OF SOFTWARE VERIFICATION

ARTE was developed by AdaCore. As stated in the PSAC, reviews by AdaCore personnel verified that the reverse-engineered materials accurately represent the intent of the “as built” product.
D.6.6. GENERIC RE SME VERIFICATION PROCESS

As stated above, reviews by AdaCore personnel verified that the reverse-engineered materials accurately represent the intent of the “as built” product. Any issues raised by AdaCore were recorded as PRs. It is recommended that the Verocel Plans and Standards should be updated to formalize the use of SMEs, rather than leave this to the project specific PSAC.

D.6.7. ACCEPTANCE CRITERIA

The ARTE requirements and design artifacts exhibit a satisfactory level of abstraction from the source code. The requirement review checklist appears to be based on the normal DO-178B criteria, e.g., “each requirement is accurate and unambiguous”, and does not reflect the special needs of RE suggested by the proposed framework. It is recommended that the Verocel requirements review checklist be updated in accordance with the framework.

D.6.8. RECOMMENDATIONS

The Certification Package was completed, assessed through four SOI audits, approved by several DERs, and signed off by the DER of record. Analysis of the planning documents and standards show that while adequate procedures were used during the verification process, some elements were done as a “good software engineering practice”, even though they were not formalized in the plans and standards. The following recommendations are made as a result of assessing the ARTE case study against the proposed framework:

1. The Verocel Software Requirements Standard should be updated to define how RE techniques should be used, when appropriate.

2. The Verocel process plans should describe and formalize the use of SMEs.

3. The Verocel requirements review checklist should ensure that the LLRs show some level of understanding of the intent of the source code, and are not too close a match to the source code.
E.1. VALIDATE FRAMEWORK

This project was controlled by the International Traffic in Arms Regulations. No details of the specific project, functionality, customer, and aircraft are disclosed, but the approach to certification is analyzed as it pertains to RE. The project was described in the PHAC as follows:

SCOPE

This document is the Plan for Hardware Aspects of Certification (PHAC) for the XXX of the ‘Identified Aircraft’.

The XXX is comprised of a number of Interface Control Units (ICUs) that are interconnected using a communication network. Each ICU contains a number of devices arranged on circuit boards. Each board contains a microcontroller, memory, a COTS device to control the lower levels of the communications protocol, and other devices developed specifically to support the ICU and XXX. These devices are interconnected using various information busses. The devices work at different speeds and have their own specific communication protocols. A Programmable Logic Device (PLD) is used to manage the interconnections between the devices and connections to various discrete I/O pins. This PLD is a Field Programmable Gate Array (FPGA) that will be programmed specifically to support the design of the ICU target board. The PLD will be developed and verified to ensure that the target board supports the functional operations of the ICU and the XXX.

This plan addresses the engineering and management processes and practices to be used during the development and verification of the PLD program. These processes and practices will result in life cycle data for the PLDs\(^2\) as a complete certification data package to support approval under the objectives of DO-254 [2] for DAL B.

This plan is modeled after the software certification guidelines of RTCA/DO-178B [1] and complies with RTCA/DO-254. It describes both the hardware development life cycle activities used by ‘Customer’ in the development of the PLD(s) for the XXX, and the verification activities performed by Verocel, Inc. on behalf of ‘Customer’.

This plan is intended for use by the certification authorities and their designees to allow a review of the intended development and verification processes, and to allow determination of the adequacy of the processes used.

\(^2\) Two separate PLDs were used on the platform: one implemented the high-speed bus, and the other implemented the analog/digital (A/D) converters, discrete signals and low speed busses.
The initial PHAC was written by ‘Customer’ and Verocel, with a clear delineation of the work to be performed and the responsibilities. An initial diagram was developed to capture the interface between the development and verification processes:

Figure E-1. Summary of Hardware Life Cycle Activities, between Customer and Verocel

The diagram above describes the processes and interactions between the companies as planned in the PHAC. Note that the ‘Detailed Design’ was the Verilog representation of the implementation. The intent was that the development process was performed by the ‘Customer’ and Verocel would perform the verification.

The activities performed by each company and the interaction activities were described in detail, and a summary is presented in the table below:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Company</th>
<th>Process</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify System and Derived</td>
<td>‘Customer’</td>
<td>‘Customer’ to allocate system and derived requirements to the ICU hardware. Refer to Hardware Development Plan (HDP) sections 3.2 and 4.1.1</td>
<td>Requirements are captured in VeroTrace and allocated to hardware configuration item #1 (HWCI-1).</td>
</tr>
<tr>
<td>Activity</td>
<td>Company</td>
<td>Process</td>
<td>Artifact</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Capture Hardware Requirements</td>
<td>‘Customer’</td>
<td>Develop the hardware requirements for HWCI-1 from the system and derived requirements. Allocate the hardware requirements to the applicable programmable devices. Refer to HDP 4.1.2</td>
<td>Draft Hardware Requirements Specification (HRS)</td>
</tr>
<tr>
<td>P1</td>
<td>‘Customer’</td>
<td>Capture the requirements in VeroTrace and establish traceability to system requirements.</td>
<td>HRS Requirements Review Checklists</td>
</tr>
<tr>
<td>Review HW Requirements</td>
<td>‘Customer’</td>
<td>Review the hardware requirements and document the review actions. Ensure all action items are closed. Establish the requirements baseline for HWCI-1. Issue the approved HRS. Refer to HDP 4.1.2, 5.2.1</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>‘Customer’</td>
<td>Capture the requirements reviews in VeroTrace.</td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>Verocel</td>
<td>Extract requirements from VeroTrace for the programmable devices.</td>
<td>Requirements-based functional tests.</td>
</tr>
<tr>
<td>Develop Requirements-Based Verification Tests</td>
<td>Verocel</td>
<td>Develop software tests to execute on the ICU that verify the functionality of the programmable devices. These tests are driven by the requirements for each device.</td>
<td></td>
</tr>
<tr>
<td>V2</td>
<td>Verocel</td>
<td>Place the requirements-based functional tests under configuration management. Update the traceability data in VeroTrace.</td>
<td>Test Review Checklists</td>
</tr>
<tr>
<td>Review Verification Tests</td>
<td>Verocel</td>
<td>Review the requirements-based tests against the device requirements.</td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td>Verocel</td>
<td>Capture the Test reviews in VeroTrace.</td>
<td></td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>‘Customer’</td>
<td>Generate the design specifications for the high-level functional entities (including programmable devices), drawings, preliminary parts lists, and Interface Control Drawings (ICDs). Refer to HDP 3.2, 4.1.2, 5.2.2</td>
<td>Design specifications. ICDs Parts Lists</td>
</tr>
<tr>
<td>P3</td>
<td>‘Customer’</td>
<td>Update the requirements traceability in VeroTrace to reference the applicable design artifacts. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>Preliminary Design Review</td>
<td>‘Customer’</td>
<td>Review the high-level architecture, design specifications, and other identified artifacts. Conduct Preliminary Design Review (PDR).</td>
<td>Design Review Checklists</td>
</tr>
<tr>
<td>P4</td>
<td>‘Customer’</td>
<td>Capture preliminary design reviews in VeroTrace. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>Software Based Verification Testing</td>
<td>Verocel</td>
<td>Perform the functional requirements-based testing for the programmable devices on the ICU. Review the functional test results (FTR) and generate the FTR checklists.</td>
<td>Functional Test Results FTR Checklists</td>
</tr>
<tr>
<td>V4</td>
<td>Verocel</td>
<td>Place the functional test results and associated reviews under configuration management. Update the traceability data in VeroTrace.</td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td>Company</td>
<td>Process</td>
<td>Artifact</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>‘Customer’</td>
<td>Generate Verilog representations for each programmable device. Update parts lists, update diagrams, and schematics. Refer to HDP 3.2, 4.1.3, 4.1.4, 5.2.2</td>
<td>Verilog source files Updated Parts lists Updated diagrams and schematics</td>
</tr>
<tr>
<td>P5</td>
<td>‘Customer’</td>
<td>Place the Verilog representation under configuration management. Update the traceability data in VeroTrace.</td>
<td></td>
</tr>
<tr>
<td>P8</td>
<td>‘Customer’</td>
<td>Capture detailed design reviews in VeroTrace. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>V5</td>
<td>Verocel</td>
<td>Extract Verilog representations for the programmable devices.</td>
<td></td>
</tr>
<tr>
<td>Develop MODSIM Test Cases</td>
<td>Verocel</td>
<td>Develop modeling and simulation (MODSIM) based test cases and scenarios to exercise the Verilog representations.</td>
<td>MODSIM Test Cases and Scenarios</td>
</tr>
<tr>
<td>V6</td>
<td>Verocel</td>
<td>Place MODSIM based test cases and scenarios under configuration management. Update the traceability data in VeroTrace.</td>
<td></td>
</tr>
<tr>
<td>Review MODSIM Test Cases</td>
<td>Verocel</td>
<td>Review the MODSIM based test cases and scenarios, consider ability to demonstrate structural coverage of the Verilog representation.</td>
<td>Test Review Checklists</td>
</tr>
<tr>
<td>V7</td>
<td>Verocel</td>
<td>Capture test reviews in VeroTrace. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>‘Customer’</td>
<td>Provide parts lists to Procurement.</td>
<td></td>
</tr>
<tr>
<td>Procurement</td>
<td>‘Customer’</td>
<td>Control parts lists, manage receiving process, release drawings to ‘Customer’s Production Release system</td>
<td>Parts lists and other released data related to programmable devices.</td>
</tr>
<tr>
<td>Verify Device Design using MODSIM</td>
<td>Verocel</td>
<td>Execute MODSIM test cases and scenarios against baselined Verilog files. Capture and annotate MODSIM simulation results.</td>
<td>MODSIM simulation results</td>
</tr>
<tr>
<td>V8</td>
<td>Verocel</td>
<td>Capture MODSIM simulation results in VeroTrace. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>Produce Structural Coverage Analysis</td>
<td>Verocel</td>
<td>Use MODSIM results to verify coverage of the Verilog. Document coverage analysis results. Document review of structural coverage.</td>
<td>Structural coverage analysis results Structural coverage analysis Review Checklists</td>
</tr>
<tr>
<td>V9</td>
<td>Verocel</td>
<td>Capture coverage analysis results and associated review checklists in VeroTrace. Place generated artifacts under configuration management.</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>‘Customer’</td>
<td>Provide any revised data to Procurement that affects purchased or manufactured components.</td>
<td></td>
</tr>
<tr>
<td>P9</td>
<td>‘Customer’</td>
<td>Provide final design package to programmable device manufacturers for fabrication.</td>
<td></td>
</tr>
<tr>
<td>P10</td>
<td>‘Customer’</td>
<td>Release manufacturers to begin fabrication.</td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td>Company</td>
<td>Process</td>
<td>Artifact</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>First Build</td>
<td>‘Customer’</td>
<td>Produce sufficient hardware assets to support the remaining development and verification activities. Refer to HDP 3.2, 4.1.5</td>
<td>Released Functional Test Plan</td>
</tr>
<tr>
<td>P11</td>
<td>‘Customer’</td>
<td>Update the Technical Data Package (TDP) contents in VeroTrace and configuration management.</td>
<td></td>
</tr>
<tr>
<td>P12</td>
<td>‘Customer’</td>
<td>Extract the TDP in preparation for transition to production.</td>
<td>Updated TDP</td>
</tr>
<tr>
<td>Production Transition</td>
<td>‘Customer’</td>
<td>Complete hardware/software integration, Perform airworthiness qualification testing, Verify the TDPs, update, and release production test procedures. Refer to HDP 3.2, 4.1.6, 4.1.7, 5.2.3, 5.2.4</td>
<td></td>
</tr>
<tr>
<td>V10</td>
<td>Verocel</td>
<td>Extract requirements and traceability data from VeroTrace. Extract baselined artifacts from configuration management.</td>
<td>Completed CD</td>
</tr>
<tr>
<td>Generate Certification Data Package</td>
<td>Verocel</td>
<td>Generate Compact Disc (CD) image of the certification data package (DO-254 compliant) for the programmable devices.</td>
<td></td>
</tr>
</tbody>
</table>

The system requirements, HLRs, and preliminary design were developed by ‘Customer’, and reviewed by ‘Customer’ for credit.

The expectation was that the HLRs and preliminary design would provide the requirements that would form the basis for the verification work performed at Verocel.

As the project progressed and the HLRs and code were delivered, initial review revealed that the requirements were inadequate. At the customer’s direction, Verocel was given the responsibility for the development of LLRs. The project became a ‘reverse engineering’ certification effort. HLRs, preliminary design, and Verilog code were imported into the VeroTrace database and Verocel configuration management system. Verocel obtained detailed specifications of all devices that were being interconnected (A/D converters, serial communication devices, etc.). These specifications provided connection details as well as the communication protocols. Reverse engineering of LLRs from the available information and Verilog code was performed using the processes typically used for software.

The code was reviewed informally, LLRs were written by Verocel engineers, and traceability was added as the information was placed into VeroTrace. The LLRs were then reviewed against the HLRs, interface specifications, and preliminary design (which was treated as a high-level design). This review was performed both at Verocel and by SMEs at ‘Customer’ site.

LLR-based tests were written in Verilog and executed on a simulator. A test scripting mechanism was written to make sure that all of the Verilog tests could be run and results captured in a single test run. The same testing script and tests were then used to run the tests in coverage mode. Coverage metrics were captured along with the test results.

This process followed the Verocel company-wide verification plans used for software, but adaptations were made as necessary for the Verilog/simulation approach used for testing.
‘Customer’ retained responsibility for integration and system testing. Verocel was not involved in those processes.

The final Hardware Accomplishment Summary was written by ‘Customer’ using information provided by Verocel and referencing lifecycle data provided by Verocel. The lifecycle data was produced using the Verocel DVD-ROM process where all of the lifecycle data is extracted from configuration management and transferred to a DVD-ROM. Additional traceability files are loaded which provide hyperlinked traceability between all artifacts as required by DO-254. A standard browser may then be used to navigate between all lifecycle artifacts. In addition, traceability information is extracted to tables that are stored on the DVD-ROM as well, to allow information to be easily searched, sorted, and analyzed.

E.2. THE RE EXPERIENCE

The PHAC was not originally written with RE in mind. The intent was to follow a waterfall development process, followed by a waterfall verification process. ‘Customer’ decided that a working prototype was more important, as there were many other applications and systems activities that were dependent on having a functional platform. The platform was developed before the rest of the system by developing the platform components, programming the PLDs and integrating the components together. The RE approach was then applied to the platform software and PLD devices:

1. The PHAC was not changed to describe the RE approach adopted part way through the program. However, the Verocel processes which address RE were called out, and the change was described in the Hardware Accomplishment Summary (HAS).

2. Access to domain experts was provided, and was very useful at the start of the project. However, part way through, the expert left, and was replaced by someone who was not as familiar with the system. By this time, Verocel had mastered much of the intent of the platform and were able to suggest changes to the platform using their expertise. System level experts were available, but they were not ‘platform’ experts, and did not fully understand the nuances of some platform behaviors.

3. The Verification Reviewers were much more risk averse than the original developers. While the system worked well, the verification engineers were much more concerned with the timing margins programmed into the PLD design. If a pair of signals was expected in sequence, then the code was written to respond correctly if a specific signal appeared before the other (e.g. data, followed by data ready). However, if they arrived at the ‘same time’ the processing was performed as if they were in very close sequence. Because the signal times could deviate slightly based on temperature, and component tolerances, the review engineers requested a predefined time margin, which would accommodate potential variations and would improve reliability. As the system worked well in the engineering laboratory, there was some reluctance to make any changes. Verocel had to insist, refusing to support DER audits unless the problems were addressed.
E.3. ASSESSMENT OF THE RE FRAMEWORK FOR AEH

RE was not planned initially, and the PHAC accurately reflected the intent. As RE development and verification were adopted, it was decided to document these deviations in the final Hardware Accomplishment Summary.

Access to SMEs was good at the start of the project, but unplanned personnel changes at the ‘Customer’ site part way through the project meant that access to SMEs became poor. Access to the overall system experts was available throughout the project, but specific expertise on platform and board level design became scarce. Fortunately, this was mitigated by the verification engineers becoming experts in the requirements and design of the platform.

The verification materials were produced and traceability was established for the devices. The consistency between all life cycle artifacts was established with independence, even though the project had a Design Assurance Level of B.

The Verocel company process plans support a RE verification approach. These plans were referenced in the PHAC for the project. RE was followed even though the PHAC did not describe a RE approach. The Hardware Accomplishment Summary did make the appropriate adjustments.

Following the recommendations of this report, it would have been better to make the use of RE much more explicit, by changing the PHAC.

The following recommendations are made as a result of assessing the AEH case study against the proposed framework:

1. The Verocel Software Requirements Standard should be updated to define how RE techniques should be used, when appropriate.

2. The Verocel process plans should describe and formalize the use of SMEs.

3. The Verocel requirements review checklist should ensure that the LLRs show some level of understanding of the intent of the Verilog code, and are not too close a match to the code.

4. The Verocel document review checklist used to review the PHAC documents should be updated to ensure that RE is addressed if used.