Safety certification of airborne software: An empirical study

Ian Dodd a,1, Ibrahim Habli b,*

a Airservices Australia, Building 101 Da Vinci Business Park, Locked Bag 747 Eagle Farm, QLD 4009, Australia
b Department of Computer Science, University of York, York YO10 5GH, United Kingdom

1. Introduction

Commercial airlines provide one of the safest forms of public transportation [1]. This has partly been achieved by placing high safety-integrity targets on all aspects of the industry from aircraft design and maintenance to crew training and aircraft operation. To ensure that aircraft systems are designed and manufactured to the required targets, different countries have commissioned various organisations that are responsible for auditing these critical systems. Flight approval or certification authorities include the European Aviation Safety Agency (EASA) in Europe [2] and the Federal Aviation Administration (FAA) in the USA [3]. When aircraft systems are first developed or upgraded, it is the responsibility of the flight certification authorities to approve the system design before it is cleared for flight. This process is known as Type Certification, where the authorities approve one sample of the proposed system type for flight use. Any exact copy of that type is also approved for flight use as long as it meets predefined design and operational constraints.

In modern avionics, it is the norm that the functionality is implemented using a microprocessor running complex computer software. In many cases, the avionics is a safety-critical item and therefore must be designed and built to the highest levels of safety integrity. The overall safety integrity of the avionics, comprising both software and hardware, is typically specified quantitatively, e.g. in terms of failure rates. However, for software, it is widely accepted that there is a limit on what can be quantitatively demonstrated [4,5], e.g. by means of statistical testing and operational experience. To address this limitation, many aerospace software standards appeal instead to the quality of the development process to assure the dependability of the software. In the civil aerospace domain, DO178B (Software Considerations in Airborne Systems and Equipment Certification) is the primary guidance for the approval of airborne software [6]. Throughout the software process, the certification authorities are required to audit the development, verification and support activities. The audits are known as Stage of Involvement (SOI) audits. Each audit is positioned at strategic points in the lifecycle to reduce the risk of failing the final certification audit. An early indication of a potential certification failure is vital to ensure that the software process is not heading in the wrong direction. An audit failure will normally require that an artefact must be reworked before the audit can be repeated. The typical time...
duration between audits is four to six months. It is therefore important for the software and safety engineers to have an indication of how well the software process is adhering to the certification requirements before a SOI audit is performed. In aerospace software projects, it is common practice to collect metrics about the defects found during the lifecycle of the project and relate these defects to a common denominator (e.g. the number of defects found per lines of code). By relating these defect metrics to the requirements of the certification authorities, indicators can be generated to determine the readiness of the software for its next SOI audit.

In this paper, we identify a set of issues concerning the auditing process for the approval and certification of airborne software. These issues were generated from interviews with experienced independent software auditors. We then propose, and empirically evaluate, a statistical method for supporting software certification audits based collecting and analysing data about the software throughout its lifecycle. This collected data is first normalised and then weighted against certification factors such as the number and types of defects, which relate to system safety. The evaluation of our proposed method is based on an industrial case study covering data collected from 9 aerospace projects and comprising 58 software releases. In this work, we focus on two groups of stakeholders, namely certification authority auditors and development teams. Auditors could use the trend of the data over the history of a project lifecycle to identify software problems and possibly misleading information. The data could also be used by the development teams within aerospace companies to assess the readiness of a software project against the certification targets. As part of our evaluation, we present the advantages and limitations of our approach from the viewpoint of both the developers and auditors.

This paper is organised as follows. Section 2 reviews existing approaches to software certification and related work. Section 3 describes a set of auditing issues concerning the software certification process. Section 4 proposes a statistical method for addressing many of the auditing issues listed in Section 3 based on the concept of Statistical Process Monitoring (SPM). This method is empirically evaluated in Section 5 through an industrial case study based on a set of data collected from anonymous aerospace manufacturers responsible for the development of safety-critical airborne software. A detailed discussion of the case study is provided in Sections 6 and 7. The legal and ethical issues concerning our proposed method are discussed in Section 8 followed by conclusions in Section 9.

2. Background and related work

2.1. Software safety certification

Certification refers to the "process of assuring that a product or process has certain stated properties, which are then recorded in a certificate" [7]. Assurance can be defined as justified confidence in a property of interest [8]. Whereas the concept of safety and assurance cases [9–11] is heavily used in goal-based standards in critical domains such as defence [12,13], rail [14] and oil and gas [15], compliance with prescriptive standards tend to be the norm in the civil aerospace domain [16–18], particularly with regard to the approval and certification of airborne software [6,19]. In prescriptive certification, developers show that a software system is acceptably safe by appealing to the satisfaction of a set of process objectives that the safety standards require for compliance. The means for satisfying these objectives are often tightly defined within the prescriptive standards, leaving little room for developers to apply alternatives means for compliance, which might better suit their software products and processes. One fundamental limitation of prescriptive software standards lies in the observation that good tools, techniques and methods do not necessarily lead to the achievement of a specific level of integrity. The correlation between the prescribed techniques and the failure rate of the system is infeasible to justify [16,20]. In goal-based certification, on the other hand, standards require the submission of an argument, which communicates how evidence, generated from testing, analysis and review, satisfies claims concerning the safety of the software functions. Despite the advantages of explicit safety arguments and evidence, there are some concerns regarding the adequacy of the guidance available for the creation of assurance arguments, which comply with the goals set within these standards (i.e. lack of sufficient worked examples of arguments or sample means for generating evidence). Many studies have considered and compared these two approaches to software safety assurance [21–23], highlighting advantages and limitations of each and how they might complement each other [24].

2.2. DO178B

In the civil aerospace domain, DO178B is the primary guidance for the approval of airborne software. The purpose of the DO178B document is "to provide guidelines for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements" [6]. DO178B defines a consensus of the aerospace community concerning the approval of airborne software. To obtain certification credit, developers submit lifecycle plans and data that show that the production of the software has been performed as specified by the DO178B guidance. The DO178B guidance distinguishes between different levels of assurance based on the safety criticality of the software, i.e. how software components may contribute to system hazards. The safety criticality of software is determined at the system level during the system safety assessment process based on the failure conditions associated with software components. These safety conditions are grouped into five categories: 'Catastrophic', 'Hazardous/Severe-Major', 'Major', 'Minor' and 'No Effect' [25,26]. The DO178B guidance then defines five different assurance levels, which relate to the above categorisation of failure conditions (Levels A to E, where Level A is the highest and therefore requires the most rigorous processes). Each level of software assurance is associated with a set of objectives, mostly related to the underlying lifecycle process, e.g. planning, development and verification activities (Fig. 1). For example, to achieve software level 'C', where faulty software behaviour may contribute to a major failure condition, 57 objectives have to be satisfied. On the other hand, to achieve software level ‘A’, where faulty software behaviour may contribute to a catastrophic failure condition, nine additional objectives have to be satisfied—some objectives achieved with independence [27].

To demonstrate compliance with DO178B, applicants are required to submit the following lifecycle data to the certification authorities:

- Plan for Software Aspects of Certification (PSAC)
- Software Configuration Index
- Software Accomplishment Summary (SAS)

They should also make all software lifecycle data, e.g. related to development, verification and planning, available for review by the certification authorities. In particular, the SAS should provide
evidence, which shows that compliance with the PSAC has been achieved. The SAS should provide an overview of the system (e.g. architecture and safety features) and software (e.g. functions and partitioning strategies). It should also provide a summary of the potential software contributions to system hazards, based on the system safety assessment, and how this relates to the allocated assurance level. The SAS then references the software lifecycle data produced to satisfy the objectives associated with the allocated assurance level.

During the certification process, it is not uncommon for aerospace companies to have open issues with the software lifecycle data. This is acceptable as long as any remaining problems do not compromise aircraft safety. All of the known problems must be declared and explained to the certification authorities during the SOI audits. Each problem must be submitted with a categorisation that determines the potential safety risk to the aircraft [29]. Four different SOI audits are set throughout the software lifecycle. Each of these audits typically lasts up to five days and is scheduled at the following stages in the software lifecycle:

- SOI#1: Software development planning review
- SOI#2: Software design and development review
- SOI#3: Software verification review
- SOI#4: Final software certification approval review

2.3. Software metrics for certification

Software metrics provide powerful means for estimating and monitoring the cost, schedule and quality of the software product and processes [30]. Metrics are particularly important for the certification of safety-critical software, especially metrics related to problem reports and test coverage. Despite the availability of several software project monitoring and measurement processes (e.g. the Practical Software and Systems Measurement (PSM) [31], Goal Question Metric (GQM) [32], Six-Sigma [33] and CONstructive COst M0del (COCOMO II) [34]), very few studies have been published on how these approaches can be applied to the software certification processes. Basili et al. discuss how the GQM approach can be used to provide early lifecycle visibility into the development of safety-critical software [35]. This is achieved through a set of “readiness assessment questions”, which should be answered against predefined measures and models. Habli and Kelly use a similar approach to the assessment of safety-critical software processes through linking the verification evidence in the safety case to the processes by which this evidence is generated [36]. Weaknesses in these processes are then used as indicators of the level of confidence that can be allocated to the safety evidence. Similarly, Murdoch et al. report some results on the application of PSM for the generation of measures for supporting the management of safety processes [37,38]. Some of these measures are related to safety certification, e.g. completion against certification data requirements. Nevertheless, none of these studies provide empirical results that validate the effectiveness of these measurement techniques against industrial data generated from the software safety certification processes.

3. Problem statement

The current approach to auditing airborne software poses a set of issues to the certification authorities and aerospace companies. These issues are summarised as follows:

**Issue 1—Auditors only audit a snapshot of the software process.** Authorities spend a short period of time at the audit compared to the time spent on the whole software engineering lifecycle. It is difficult for the auditors to identify complex problems in the time taken to carry out an audit.

**Issue 2—Software safety standards are open to different interpretations.** It is likely for software and safety engineers to over- or under-engineer the software in their effort to fulfil the requirements of the standards. To this end, they may be over- or under-spending to achieve certification credit or under-spending and risking an audit failure. An aerospace company can seek advice from the certification authorities, but, to maintain their independence, the authorities are restricted in the advice they can offer.

**Issue 3—Companies might be deceitful:** This is very rare but nevertheless not unheard of. A software company might try to mislead the authorities at a SOI audit regarding the status of their software process. This might be due to financial or timescale factors, pressurising the company into making false declarations in order avoid a delay in project schedules [39]. Typically, this would be manifested near the time of an audit, when the company could realise that the project is not ready for a SOI audit.

**Issue 4—There is a lack of objective criteria for determining software status:** In order to assess the achievement of the
certification requirements, the certification authorities must understand the technical aspects of the software being developed e.g. the safety issues raised during the development. This is normally achieved by reviewing the technical reports of the project, sometimes during the short duration of a SOI audit. Absorbing the technical subtleties of a software project and making an objective assessment is a challenging task for the auditors, particularly when they are responsible for the auditing of multiple software projects from different companies.

**Issue 5—Same errors made again and again:** Many companies enter SOI audits and make the same mistakes they have made before, mainly due to poor understanding of the certification objectives that the authorities aim to assess. This can be due to lack of experience (perhaps due to turnover of staff) or lack of rigorous quality control within these companies. The introduction of new technologies could also contribute to this problem. Equally, this issue is a source of problems for companies developing airborne software for the first time.

The above issues are partly generated from interviews with experienced independent software auditors. The focus of the interviews was on the efficiency of the auditing process as well as the level of transparency in the relationship between the certification authorities and the development teams. Some insightful quotations from these auditors are documented in Table 1. Of course, the above issues form a subset of problems encountered during the auditing process. Development practices can vary across countries, companies and projects. For example, other issues include inappropriate reuse of certification evidence across different projects or immature deployment of novel technologies. However, the above five issues seem to be the most recurring difficulties facing the software safety auditors interviewed.

To this end, if the certification authorities had metrics, based on predefined criteria, reflecting the status of the software project before an audit commences, they could be better prepared for a SOI audit, particularly from the technical perspective. If these metrics are provided at regular intervals as a project develops, the certification authorities could use the trend of the data over the history of a project lifecycle to identify any misleading information. For example, one indication might be in the form of sudden improvements in progress graphs near the time of the SOI audit. Further, if the metric data is normalised across different software projects, it can also be used to compare the quality and progress of a project against other projects which have previously successfully passed the SOI audits. Equally, quality assurance departments within aerospace companies could use these metrics to determine the readiness of the software against the certification targets. Based on these metrics, focused training may be provided to aerospace companies on ways to improve their in-house capability and the quality of their certification material.

### 4. Data collection and analysis for certification audits

In this section, we define a statistical method for collecting and analysing live data from aerospace software projects to assess the readiness of a project for a SOI audit by comparing the project’s data against historical data collected from past projects, which successfully passed the SOI audits. We first discuss the types of data which should be collected and justify why they are important for the certification process. We then define weighting factors for categorising the collected data against their importance for the satisfaction of the certification objectives. We finally specify a detailed four-step process for statistically analysing a project’s data against data collected from past successful projects. It is important to note at this stage that what we propose in this section is a method for assessing the readiness of a project for a certification audit and as such offers process evidence rather than product evidence. This process evidence merely relates to compliance with the DO178B guidance and does not directly relate to confidence in the safety claims concerning the software product.

#### 4.1. Data collection

As part of the software certification process in the civil aerospace domain, different types of lifecycle data are produced, which relate to planning, development, verification and support. Many of these types of data are produced based on compliance with the aerospace software guidance DO178B and are summarised in Table 2.

In this study, based on the lifecycle data shown in Table 2, we defined 15 metrics, which should be collected as part of the software certification process. These metrics are refined and encoded using the Goal Question Metric (GQM) technique [32]. GQM is a top–down measurement framework for defining goals related to products and processes. A goal is interpreted using a set of questions whose answers are associated with objective or subjective metrics. A goal is specified relative to four elements: Issue (e.g. reliability or safety), object (e.g. software, platform or process), viewpoint (e.g. independent or internal auditing) and purpose (e.g. failure rate reduction). Questions are derived from these elements and subsequently their answers estimate the achievement of the top goal using primitive metrics (e.g. faults detected, failure classifications or objectives satisfied). These primitive metrics provide measurable references against which the analysis mechanisms can be performed. Table 3 document the GQM for Software Requirements Specification (SRS). This GQM will be used throughout this paper to illustrate the various steps for data collection and analysis.

As shown in the metrics of the GQM in Table 3, a key factor in the assessment of the readiness of the software for certification audits is the number and types of defects associated with each
lifecycle artefact (e.g. problem reports associated with SRS). However, feeding raw defect data is not enough to determine if a project is ready for a SOI audit. Collected data must first be pre-processed (filtered) to draw out the issues that are important for certification during the SOI audits and amplify these issues during data analysis. To amplify the relevant factors, collected data should be normalised and then weighted against certification factors such as the number and types of defects, which relate to system safety.

When a defect is identified during the software process, information about the defect is stored in a problem report. Problem reports act as a repository for the issues that should be corrected. They also contain a rating that defines the impact of the identified problems on the safety of the aircraft. The requirements for software problem reports are predefined in certification guidance documents. One of these documents is provided by EASA and titled “Certification Review Item CRI – T8 – The Management of Software Open Problem Reports” [29]. This document specifies five categories of problem reports, which are listed in Table 4.

Table 4
Problem report classifications [29].

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE 0 or CAT 0</td>
<td>A problem whose consequence is a failure, under certain conditions, of the system with a safety impact.</td>
</tr>
<tr>
<td>TYPE 1A or CAT 1A</td>
<td>A failure with a significant functional consequence; the meaning of significant should be defined in the context of the related system and its specific application.</td>
</tr>
<tr>
<td>TYPE 1B or CAT 1B</td>
<td>A failure with no significant functional consequence.</td>
</tr>
<tr>
<td>TYPE 2 or CAT 2</td>
<td>A fault which does not result in a failure (i.e. no system functional consequence, fault not detectable by the crew in foreseeable operating conditions).</td>
</tr>
<tr>
<td>TYPE 3A or CAT 3A</td>
<td>A significant deviation whose effects could be to lower the assurance that the software behaves as intended and has no unintended behaviour.</td>
</tr>
<tr>
<td>TYPE 3B or CAT 3B</td>
<td>A none significant deviation to the methodology (plans) that does not affect the assurance obtained.</td>
</tr>
</tbody>
</table>

Table 2
DO178B Lifecycle Data.

<table>
<thead>
<tr>
<th>Plan for software aspects of certification</th>
<th>Design description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software development plan</td>
<td>Source code</td>
</tr>
<tr>
<td>Software verification plan</td>
<td>Executable object code</td>
</tr>
<tr>
<td>Software configuration management plan</td>
<td>Software verification cases and procedures</td>
</tr>
<tr>
<td>Software quality assurance plan</td>
<td>Software verification results</td>
</tr>
<tr>
<td>Software requirements standards</td>
<td>Software life cycle environment</td>
</tr>
<tr>
<td>Software design standards</td>
<td>Configuration index</td>
</tr>
<tr>
<td>Software code standards</td>
<td>Software configuration index</td>
</tr>
<tr>
<td>Software accomplishment summary</td>
<td>Problem reports</td>
</tr>
<tr>
<td>Trace data</td>
<td>Software configuration management records</td>
</tr>
<tr>
<td>Software requirements data</td>
<td>Software quality assurance records</td>
</tr>
</tbody>
</table>

lifecycle artefact (e.g. problem reports associated with SRS). However, feeding raw defect data is not enough to determine if a project is ready for a SOI audit. Collected data must first be pre-processed (filtered) to draw out the issues that are important for certification during the SOI audits and amplify these issues during data analysis. To amplify the relevant factors, collected data should be normalised and then weighted against certification factors such as the number and types of defects, which relate to system safety.

When a defect is identified during the software process, information about the defect is stored in a problem report. Problem reports act as a repository for the issues that should be corrected. They also contain a rating that defines the impact of the identified problems on the safety of the aircraft. The requirements for software problem reports are predefined in certification guidance documents. One of these documents is provided by EASA and titled “Certification Review Item CRI – T8 – The Management of Software Open Problem Reports” [29]. This document specifies five categories of problem reports, which are listed in Table 4.

Table 4
Problem report classifications [29].

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE 0 or CAT 0</td>
<td>A problem whose consequence is a failure, under certain conditions, of the system with a safety impact.</td>
</tr>
<tr>
<td>TYPE 1A or CAT 1A</td>
<td>A failure with a significant functional consequence; the meaning of significant should be defined in the context of the related system and its specific application.</td>
</tr>
<tr>
<td>TYPE 1B or CAT 1B</td>
<td>A failure with no significant functional consequence.</td>
</tr>
<tr>
<td>TYPE 2 or CAT 2</td>
<td>A fault which does not result in a failure (i.e. no system functional consequence, fault not detectable by the crew in foreseeable operating conditions).</td>
</tr>
<tr>
<td>TYPE 3A or CAT 3A</td>
<td>A significant deviation whose effects could be to lower the assurance that the software behaves as intended and has no unintended behaviour.</td>
</tr>
<tr>
<td>TYPE 3B or CAT 3B</td>
<td>A none significant deviation to the methodology (plans) that does not affect the assurance obtained.</td>
</tr>
</tbody>
</table>

These issues cast some doubt over a simple weighting metric that only considers TYPE 0 problem reports. For example, if a software system had one TYPE 0 problem report and 50 TYPE 1A or TYPE 1B problem reports, an auditor might raise a concern regarding the number of the TYPE 1A and TYPE 1B problem reports and potentially prevent the release of the software in its current state to flight. In practice, the categorisation of the problem reports is complicated and can be error prone. This can be attributed to some of the following factors:

- Problem reports are often poorly written, leading to errors during the severity assessment and categorisation.
- The impact of some problems is often too complex for a single engineer to understand. The full scope of the impact of a problem report is not truly understood by the engineer making the assessment.
- The categorisation of problem reports changes as the development and verification activities progress, e.g. the resolution of one problem report may influence the categorisation of other problem reports.
Table 5
Data on errors with Problem Report classification on a large aerospace project.

<table>
<thead>
<tr>
<th>Problem Reports raised on the project before the first SOI#4 Issue</th>
<th>Percentage of total number of Problem Reports (%)</th>
<th>Number of Problem Reports this percentage relates to</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrongly categorised TYPE 0 Problem Reports that should have been TYPE 1A or TYPE 1B</td>
<td>0.03</td>
<td>2</td>
<td>Experienced engineer found these issues late in the programme as he was assessing very high-risk Problem Reports.</td>
</tr>
<tr>
<td>Wrongly categorised TYPE 1A and 1B Problem Reports that should have been TYPE 0</td>
<td>0.1</td>
<td>6</td>
<td>Problem Reports missed by initial assessment but were later found to be incorrect after accidental re-reviews of the Problem Reports.</td>
</tr>
<tr>
<td>Wrongly categorised TYPE 2, 3A and 3B Problem Reports that should have been a functional Problem Report</td>
<td>4</td>
<td>240</td>
<td>Inexperience of Problem Report reviewers led to many incorrect assessments.</td>
</tr>
</tbody>
</table>

Data analysis

Data analysis in our proposed method is centred on comparing problem report data from a live project with historic data captured from projects that successfully passed the SOI audit. As a prerequisite, normalised data from these projects should be collected in order to generate a normal distribution for these projects. The live project data can then be compared against this normal distribution. If the project is within the ‘maximum allowable deviation’ on this normal distribution (discussed in more detail in Section 5.1), it can be deemed as ready to be submitted to the SOI audit and vice versa. If the project was deemed to be ready for the audit and passes the audit, the data from the project should be added to the data set that generates the normal distribution for the successful projects. Consequently, the database generating the normal distribution increases in size and capability. However, if the project fails the SOI audit, an investigation must take place to understand why this failure had not been identified prior to the audit. Projects which have failed the audit, when they had been expected to pass, should be used to identify patterns in the data that show why a project may fail an audit. Once a pattern is identified, it should be fed back into the main weighting metrics.

4.3. Statistical Process Monitoring (SPM)

At the heart of the data analysis stage is the concept of Statistical Process Monitoring (SPM) using a normal distribution curve [41]. SPM provides the ability to statistically analyse similar procedures and produce results, which are normally distributed around the mean value. This is directly applicable to the aims of this paper. Certification data items are produced from similar software engineering processes, which are driven by the same guidelines, i.e. DO178B guidelines, and are within the same domain, i.e. civil aerospace. Although software engineering processes may differ between aerospace companies, certification audits should be carried out in a consistent manner using a common methodology and standard. If the data collected from different projects is normalised to account for the different size and complexity of the projects, SPM can effectively be used to analyse such data. In this paper, we pre-process the raw data before it is plotted on a normal distribution to amplify the particular issues that are of concern to the certification authorities, based on the weighting factors described in Section 4.2.

When SPM is used, variability (standard deviation) in the data collected from the projects that have successfully passed the SOI audits can be identified. On the one hand, if there is a large standard deviation, we can see that the variation between the SOI
audit passes is considerable. This might reveal that the audit results are unacceptably subjective and that the auditors might not have a common theme, mandate or training. This may also be used to identify certain anomalies, such as overly strict or lenient auditors. On the other hand, if the standard deviation is small, it might show that the auditors are applying common practices and rules.

4.3.2. Data analysis steps

Driven by the concept of SPM and the weighting factors discussed in Section 4.2, Fig. 2 depicts an overview of our proposed data collection and analysis process, comprising the following four steps:

**Step 1—normalising the data:** Raw defect data for all certification artefacts in a software lifecycle step for a project is collected (e.g. all the SRS’s in the project). This raw defect data details the number of problem reports for each problem report type (TYPE 0, TYPE 1A, TYPE 1B, TYPE 2, TYPE 3A, TYPE 3B). Next, the total number of problem reports for each of the six problem report types is calculated. The six resulting sums are then normalised using a selected normalisation factor (e.g. the total...
number of requirements in all the SRS for a project). All of these normalised problem report types are then summed to produce the 'lifecycle stage total normalised sum'. This normalisation step is repeated for all of the projects in the data set (e.g. repeat for all of the SRS's in the projects that are in the data set).

Step 2—weighting the data: For each project, the normalised raw defect data items for each of the six problem report types (calculated in the previous step) are multiplied by the weighting fraction defined in Table 8. This produces a set of six weighted normalised data items for each problem report type for a project (e.g. all the SRS's from one project). This is repeated for all of the projects in the data set (e.g. repeat for all of the SRS's in all of the projects that are in the data set). After the data has been multiplied by the weighting values, it reflects the authorities' importance criteria and it can now be used to produce a normal distribution, from which the mean and the standard deviation ($\sigma$) can be identified.

Step 3—plotting the normal distribution: The normal distribution for all of the 'lifecycle stage total weighted normalised sum' in the data set (e.g. for all of the SRS's in the projects that are in the data set) is calculated. The normal distribution for the 'lifecycle stage total weighted normalised sum' is then plotted.

Step 4—comparing the artefacts from one project against other projects: The normal distribution can now be used to compare a single project's software lifecycle certification artefacts (e.g. all SRS's from one project) against this normal distribution to determine if they are within the maximum allowable deviation limits. The maximum allowable deviation limits can either be assumed or predefined, such as in a Six Sigma scheme [33], or calculated as illustrated in the next section.

5. Industrial case study

In this section, we evaluate the statistical analysis method defined in the previous section using a data set which was provided, in an anonymous format, by two large avionics manufacturers. These manufactures are responsible for the
5.1. Calculating the maximum allowable deviation from the mean

As a prerequisite for applying the four-step process proposed in the previous section, the maximum allowable deviation from the mean of the normal distribution should be defined. In order to calculate the maximum allowable deviation, factors such as Natural Variance (NV) and Assignable Variations (AVs) should be considered. This will allow current projects to be judged against a calculated rather than an assumed (3σ envelope) acceptable tolerance envelope. The NV observed in SPM is the variance in the recorded data due to natural fluctuations in how items are produced. For example, for findings identified during an SRS review, some of the results could be attributed to issues that are not readily controllable such as reviewers having personal problems affecting their review judgements. On the other hand, variations that occur in a process due to conscious technical or managerial reasons are known as Assignable Variations (AVs). Generally, NV is difficult to reduce, as it is an inherent part of the process. Of course, the data has been captured from only two avionics manufacturers. The data was solicited using a questionnaire to extract the figures. The questions were derived from the analysis of data from previous SRS reviews that indicated where there may be weaknesses in the SRS production and review process. It is important to note that during the development of airborne software, there is an amount of AV that could be reduced, but is agreed as being an acceptable deviation. This is as a result of project management decisions to release the software to the SOI audit with known problems, defined in open problem reports. NV and AV are analysed and calculated in Sections 5.1.1 and 5.1.2 and are then used in Section 5.1.3 to determine the maximum acceptable variation or ‘acceptable tolerance envelope’.

5.1.1. Calculating the Natural Variance (NV)

In this section, metrics are collected from the SRS developers and reviewers in order to calculate the NV. Table 10 shows a set of data collected during the SRS development and review from the two avionics manufactures. The data was solicited using a questionnaire to extract the figures. The questions were derived from the analysis of data from previous SRS reviews that indicated where there may be weaknesses in the SRS production and review process. Of course, the data has been captured from only two
### Table 10
Natural Variance (NV) calculation.

<table>
<thead>
<tr>
<th>Item</th>
<th>Source</th>
<th>Description</th>
<th>COMPANY 1 estimated variance in terms of number Problem Reports raised per 100 Problem Reports</th>
<th>COMPANY 2 estimated variance in terms of number Problem Reports raised per 100 Problem Reports</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reviewer</td>
<td>Reviewer unknowingly does not understand the parent requirement and raises a Problem Report due to his lack of knowledge</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reviewer</td>
<td>The Reviewer is trained but inexperienced and unknowingly misunderstands the review requirements and raises unnecessary Problem Reports</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reviewer</td>
<td>The Reviewer unknowingly does not understand the content of the SRS requirements being reviewed</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reviewer</td>
<td>The Reviewer concentrates too much on syntax and not technical content and raises Problem Reports to cover typos</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reviewer</td>
<td>The reviewer is not sure if there is a problem so raises a Problem Report</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Author</td>
<td>Parent requirement is poorly defined and is unknowingly implemented incorrectly causing a Problem Report to be raised</td>
<td>1.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Author</td>
<td>The Author is trained but inexperienced and therefore unknowingly writes bad requirements causing a Problem Report to be raised</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Author</td>
<td>Conflicting requirements force a fault to be reported as a Problem Report</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Author</td>
<td>Problem Report raised twice for the same finding but not identified by the Problem Report originators</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Total number of Problem Reports raised for every 100 Problem Reports due to Natural Variance caused by a team reviewing the SRS's</td>
<td>14</td>
<td>9</td>
<td>Sum items 1–9 and derive the % This is then the % of Problem Reports raised that are due to the NV</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>For a typical project that passed SOI#2 audit the average number of Problem Reports raised in total is</td>
<td>908</td>
<td>908</td>
<td>Average across sampled projects</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>For a typical project from this company the average number of Problem Reports raised due to the Natural Variance is</td>
<td>127</td>
<td>82</td>
<td>Item 10/100 × Item 11 (not yet normalised)</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>For a typical project that passed SOI#2 audit the average total number of requirements on analysed projects is</td>
<td>2631</td>
<td>2631</td>
<td>Average across sampled projects see</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Natural variance of Problem Reports raised for SRS normalised by requirements</td>
<td>0.048</td>
<td>0.031</td>
<td>Item 12/Item 13 Natural Variance (Normalised)</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>AVERAGE NV of two project shown (NOT WEIGHTED)</td>
<td>0.04</td>
<td></td>
<td>Average of Item 14 for both projects</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Median of weighting values</td>
<td>0.25</td>
<td>Refer to Table 8</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>AVERAGE NV of two project shown (WEIGHTED)</td>
<td>0.01</td>
<td></td>
<td>Item 15 × Item 16</td>
</tr>
</tbody>
</table>

### Natural variance of SRS faults Pre-requisites to analysis

Data gathered from SRS writing and review of DO178B Level ‘A’ software

<table>
<thead>
<tr>
<th>Item</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Reviewer</td>
<td>The documents presented for Reviewer are the correct version and applicable to the review</td>
</tr>
<tr>
<td>19</td>
<td>Reviewer</td>
<td>The Reviewers are trained to carry out DO178B reviews</td>
</tr>
<tr>
<td>20</td>
<td>Author</td>
<td>The Authors are trained to prepare a DO178B document</td>
</tr>
<tr>
<td>21</td>
<td>Author</td>
<td>The author and reviewer activity are localised to the one system and are not part of the global aircraft systems integration</td>
</tr>
</tbody>
</table>
companies, which limits the accuracy of the calculation. To improve the accuracy of the NV, more reference points would be required. However, this limited data set demonstrates the principle. The data was captured after both companies had already taken mitigation actions.

5.1.2. Calculating the Assignable Variance (AV)

The AV comprises two types of variation:

- Variations that can be reduced through improving the engineering processes;
- Variation due to project-based decisions to leave some problem reports open at the time of the SOI audit.

If we assume that the open problem reports capture the development process issues, then the AV can be attributed to the problem reports left open at the time of the SOI audit. Therefore, the average number of problem reports left open at the time of the SOI#2 audit, across our sample projects in this case study may be used to calculate the AV. The examination of the data set used in this case study shows that the average number of SRS Problem Reports that were left open for our sample projects at the time of their SOI#2 audit was 137. The assignable variance is therefore equal to 137, which will be normalised by dividing it by the average number of requirements. Table 11 shows how the AV is calculated using the average number of problem reports left open at the SOI#2 audit and the average total number of requirements.

5.1.3. Calculating the maximum tolerance envelope

The Maximum Variance is calculated by adding the AV to the NV. Next, we must weight this value using the MEDIAN of the weighting values (as calculated in Table 8). This value reflects the Maximum Variance for the weighted normal distribution (as shown in Table 11) which is the maximum number of normalised defects allowed in the data set or the maximum allowable tolerance. The MEDIAN of the weighting values is used because the weighted data is biased towards the TYPE 0 problem report. Further, there is a non-linear relationship between each of the problem report types. Most of the problem reports that remain open for the SOI are actually Type 2 or Type 3. Therefore, if the mean of the weighted data values is used, it would unfairly bias the maximum tolerance weighting (due to the weighting placed on Type 0 Problem Reports), making it larger than it should actually be. The maximum variance will now be referred to as the 'maximum acceptable tolerance'.

5.2. Preparing and plotting multiple project SRS defect data

In the section, we apply the four-step process proposed in Section 4.3.2. The data has been collected from 53 software releases. The defect data for each of the releases reflects the software status when it was presented to the SOI#2 audit. All of the 53 software releases passed the SOI#2 audit successfully. The data shows the number of problem reports that were raised during the SRS development, which have been grouped against six categorisation types. The following is a summary of the application of the four-step process:

1. For each project, the total number of SRS findings of each problem report categorisation type has been normalised by dividing it by the total number of requirements in the SRS. All of the normalised values for each problem report criticality type have then been summed (known as the ‘lifecycle stage total normalised sum’).
2. For all the projects, a normal distribution has been derived from the ‘lifecycle stage total normalised sum’ and the data is plotted and shown in Fig. 3.
3. The normalised data has then been weighted according to Table 6. After the data has been weighted, all the weighted normalised values for each problem report type have then been summed (known as the ‘lifecycle stage total weighted normalised sum’).
4. For all the projects, a normal distribution has been calculated from all the ‘lifecycle stage total weighted normalised sums’ and the data is plotted as shown in Fig. 4.

5.2.1. Preliminary analysis

Fig. 3 shows the normal distribution plot for the un-weighted ‘lifecycle stage total normalised sums’ raw defect data for the 53 software releases that successfully passed the SOI#2 audit. Further, the plot shows the ‘acceptable tolerance envelope’ that was determined using the maximum variation, which was calculated in Section 5.1.3. The plot also shows that 3 of the projects plotted actually breach the ‘acceptable tolerance envelope’. This indicates that the SRS for those projects should have been reworked so that they fall within the target tolerance value. It must be noted here that these 3 projects successfully passed the SOI#2 audit.
On the other hand, Fig. 4 shows the normal distribution plot for the ‘lifecycle stage total weighted normalised sums’ for the same 53 software releases that successfully passed the SOI#2 audit. This is a more important plot as it shows the weighted data plotted with respect to the issues, which are important for the certification authorities during the audit. The plot also shows the ‘acceptable tolerance envelope’ that was calculated in Section 5.1.3. All of the projects are within the ‘acceptable tolerance envelop’, which is in contrast to the plot in Fig. 3.

This demonstrates the difference between collecting and analysing raw defect data in isolation and performing the analysis against weighted data that reflects the auditing needs of certification authorities during the audit. If decisions had been made based on the raw defect plot (Fig. 3) without considering the weighting criteria, additional work may have been performed, which would have delayed the SOI#2 audit without having any effect on the outcome of the audit. If the project decision makers had been presented with Fig. 4, they could have made a balanced decision between taking the risk of failing the audit or passing it without reworking the SRS. This can be useful as it allows calculated risks to be taken, which is central to safety-critical projects in which delays can be costly and create damaging publicity.

5.3. Evaluation against five random projects

In this section, we evaluate the analysis method and the figures generated in the previous section against data from five different projects. Interviews were conducted with the project managers of these projects. The objective of these interviews was to understand the background behind the data in order to be able to interpret the outcome of the evaluation, particularly when the data is plotted against the successful SOI#2 audit project data normal distribution. It is important to note that none of these projects were included in the data used to derive the successful SOI#2 audit normal distribution shown in Figs. 3 and 4. As such, this provides a clean baseline against which these projects could be compared. Four of these projects failed their SOI#2 audit and only one project passed the SOI#2 audit.

5.3.1. Un-weighted data plotted against the successful SOI#2 audit normal distribution

Fig. 5 shows a plot of the un-weighted data of the selected five projects. Four of the test project data points lie outside of the ‘acceptable tolerance envelope’ and therefore give the initial indication that there are problems with the SRS artefacts. In all four cases, a synopsis from the project history (extracted through interviews with the project managers) indicates that all of the projects were in need of recovery and that there were serious quality issues associated with the SRS development. There were clearly unacceptable numbers of problem reports raised against the SRS artefacts. This is clearly reflected in the un-weighted data.
2. The same decision-making approach discussed above with regard to ‘failed project 3’ could be taken with the ‘failed project 2’. However, this clearly will carry much more risk of failure as it is further outside of the ‘acceptable tolerance envelope’. It is acknowledged further work is needed to address borderline cases. These cases could be calibrated using further risk analysis that allows the level of risk to be understood and a more informed decision to be made.

3. The ‘Failed Project 4’ is clearly outside of the ‘acceptable tolerance envelope’. The SRS problem reports are indicating that the SRS artefacts have quality issues. This project is clearly not ready for a SOI#2 audit and the decision of submitting the project to a SOI#2 audit would be wrong.

4. The ‘Passed Project 1’ has clearly moved into the ‘acceptable tolerance envelope’ and is indicating that the project is ready for and SOI#2 audit (whereas this project was outside of the ‘acceptable tolerance envelope’ in Fig. 5). This is again supporting the reasoning regard to ‘failed project 3’ could be taken with the ‘failed project 2’.

5. The ‘Failed Project 1’ introduces a new issue to the analysis. The project appears to be within the ‘acceptable tolerance envelope’. Yet, the project has failed the SOI#2 audit. The reason was that a Type 0 problem report was discovered at the actual SOI#2 audit. The problem report was deemed to pose an
The event has the following criteria [42]:

1. Rarity: it is a surprise;
2. Impact: it has an extreme impact;
3. Retrospective predictability: after the fact, it is rationalised, “making it explainable and predictable”.

By further examining the history of ‘Failed Project 1’, each of the above criteria is clearly satisfied:

1. The event was not identified by the software engineers and was a surprise to them. However, the event was identified by the certification authorities as being severe enough to fail a SOI#2 audit. The software engineers believed that the project was ready for the audit. It was not a project management decision to press on with the SOI#2 audit regardless of the existence of the problem report.
2. The event had a major impact. The audit did actually fail which caused a major delay to the project and embarrassment to the software engineers.
3. After interviewing the project manager, he declared that after the authorities had identified the issue, it was obvious that the problem report should have been identified as being a major safety issue and should have been resolved before the SOI#2 audit was performed.

The hidden impact of this “Black Swan” event is the professional embarrassment factor in that the certification authorities have identified this problem and not the software engineers. Although as with many findings of this nature, there were warning signs that there was an issue with the software, but these warning signs were not acted upon.

6. Discussion

The case study in the previous section shows that the use of the data collection and analysis method proposed in Section 4 has some benefits, particularly for monitoring the SRS quality and the readiness of the software artefacts for SOI#2 audits. If the raw un-weighted SRS defect data is analysed in isolation, the data can easily give the false indication that a project is not in a suitable state to pass a SOI#2 audit and that the defective SRS artefacts must be reworked before the audit. As illustrated in the previous section, this is not always the case. When the data is weighted to account for the certification considerations, it reveals that some of the uncertainties concerning the quality of some SRS artefacts do not pose high risks to the success of the SOI#2 audit. The opposite is also true. The data has demonstrated situations where the SRS quality is believed to be acceptable, but is actually unacceptable due to safety-related problem reports, which pose a high risk of a SOI#2 audit failure. This is because safety related problem reports are a key consideration for the certification authorities. Of course, a weakness of the analysis method was revealed by evaluation against a “Black Swan” project. This evaluation is a difficult test for most statistical methods to pass. However, as suggested by Taleb [42], “Black Swan” events could be addressed by factoring a number of robustness measures into the project. However, to effectively reduce the number of “Black Swan” projects, it is not only the analysis method that will need to change, but also the development processes that feed the problem report data to the method. In the rest of this section, we discuss a number of observations relating to the selected weighting criteria and accuracy limitations of the case study and the SPM normal distribution.

6.1. Weighting criteria

The weighting criteria defined in this paper were generated from discussions with various aerospace auditors and the experience gathered by the first author from various SOI audits. The calculation of the weighting criteria presented demonstrates how the weighting factors can be calculated to reflect the needs of the certification authorities. However, to improve accuracy, the calculation of the weighting factors should be discussed at length, and agreed on, by experienced certification auditors and then discussed and shared with the aerospace companies. Concerning the precision of the weighting criteria, we have carried out a sensitivity check on the weighting factors. Needless to say, this check has revealed that the precision of the weighting factors has little effect on the final conclusions of the analysis method (i.e. is the project ready for a SOI audit or not?). What is more important, however, is the relative weighting between the different problem report types.

6.2. Accuracy limitations of the case study

The data used in the case study is difficult to extract from aerospace companies due to confidentiality issues. The lack of more data sets might be a weakness. Specifically, there are two areas where this weakness is manifested:

- The calculation of the ‘acceptable tolerance envelope’ is an area of uncertainty. In this paper, the calculation of the NV relies on only two sets of data from two companies. This is obviously too small to be reliable. This could be ignored if the NV were insignificant. However, at present, the NV is 43% of the ‘acceptable tolerance envelope’. As such, a shift in the calculated NV will influence the envelope significantly. To this end, data from more companies is needed to improve the accuracy of the calculated AV, ideally from different nations throughout the world.
- Another area of uncertainty is the amount of data used to produce the SOI#2 audit Normal Distribution graphs for the successful projects, as shown in Figs. 5 and 6. Although this is a reasonably large set of data (53 software releases), the ideal situation would be that such an important graph should be produced from thousands of data points collected from a variety of manufacturers and projects across the world. This would allow for a global view of the successful SOI audit projects criterion. Using a larger data set may change the shape of the normal distribution bell curve, which could change the number of projects that fall inside the ‘acceptable tolerance envelope’.

6.3. SPM normal distribution

When different projects and different lifecycle data are considered from around the world, there will undoubtedly be a difference between the mean and the standard deviation calculated for each individual project data set. This is because there are a number of factors that could influence the number of problem reports found on different projects, which would clearly skew the normal distribution. Examples of these issues are as follows:

- If two projects were compared where the code on one project was developed manually and the code on the other project was automatically generated, then there would most likely be less
7. Overall evaluation

Apart from one person with a basic understanding of statistical modelling, most engineers and auditors should be able to use the approach presented in this paper without the need for any further calibration of the models. It is important to note that our case study focuses on the review of SRS at SOI#2 audits. However, our approach is designed to be used continuously through the auditing process. As the software is developed, live defect data will be analysed in the SPM model and the overall state of the artefacts in the lifecycle stage can be evaluated. If at the measurement point, an artefact is deemed to be unsuitable for certification, the artefact should be reworked to improve its quality and mitigate identified defects. This rework will then result in improved defect data being fed into the SPM model, which could indicate that the software has reached a suitable level of quality for submission to the certification authorities. This approach may be used live and on a continual basis if the model is integrated with the problem report database. In the rest of this section, we discuss the advantages and disadvantages of using our data collection and analysis method from the point of view of the two key stakeholders, namely the certification authorities and the aerospace companies.

7.1. Advantages

The data collection and analysis method offers the following advantages with regard to the auditing process:

- The history of the software development can be maintained and tracked over the lifecycle of a project. The engineers and auditors can assess how the quality of the product is improving as it progresses through its development phase into the in-service phase and then through its operational life. The amount of operational issues observed can then be used to gather statistics regarding the number of failures in service, with respect to the number of related problem reports that were raised during the development phases. This may also help in crash investigation if the causes of a crash are related to the software [47].

- If software project data is provided at regular intervals as a project develops, the certification authorities could use the trend of the data over the history of a project to acquire more exposure to the software process weaknesses. They no more need to make important decisions based on a small snapshot of the software lifecycle data. Further, it will also be more difficult for companies to mislead the authorities at the audits regarding the quality of the software. That is because the authorities will have the opportunity to follow the recorded problem report history throughout the lifecycle of a project.

- The collected data will help the certification authorities in auditing their own auditors. The authorities can then see if there is a trend occurring, where an auditor may be too strict or too lenient. It will also allow the certification authorities to identify where their auditors may be lacking in experience and training. As such, the certification authorities can focus the training of their auditors on the parts of the lifecycle, which can identify the extent to which a project is meeting the certification objectives.

- The metrics and the issues raised above allow the certification authorities to be audited by their governing bodies. Having data about the audits, the quality of the auditors and the quality of the projects can help these governing bodies in determining if the certification authorities are fulfilling their role as an independent certification agency.

- The uncertainty of whether a project will pass the SOI audit becomes an issue with respect to the required independence of the certification authorities. Confidentiality can be gained by the software management team that the project is heading in the right direction and issues can therefore be corrected before the audit. This is very important considering the large development costs and multiple stakeholders involved in aircraft development. The early detection and correction of problems can reduce the costs significantly.

7.2. Disadvantages

The data collection and analysis method also has a number of disadvantages, including the following:

- DO178B is a guidance document and if the certification authorities monitor software development at such a close level, they might be close to mandating the development activities that need to be completed and monitored. This also becomes an issue with respect to the required independence of the certification authorities.

- If the certification authorities overuse the metrics, they could easily be misled by a good set of data that might actually be hiding poor software processes. The analysis method is obviously open to abuse if the companies supply incorrect data. As such, the auditors should only use the data as a guide. They should still use their auditing skills to identify if the processes behind the development are of the required quality and are being followed.

- The certification authorities are not responsible for stating that the software is safe. This responsibility lies with the aerospace companies. If the certification authorities become too involved...
in the detail of the development, then the aerospace companies may claim in the event of an accident that the certification authorities were fully aware of the issues, thereby implicating the certification authorities by making them indirectly accountable for assuring the software’s safety.

- As part of the data collection and analysis method, commercially sensitive data regarding a company’s development processes is used. This could be damaging if this data is leaked to a competitor. An example scenario is the development of a joint venture aircraft. Although the different companies are working together, one company might discover that another company within the joint venture is behind schedule and use this information to their advantage in a ‘game of brinkmanship’ to see which company declares late delivery first and hence pays the late penalty costs.

8. Legal and ethical issues

In an ideal application of the approach presented in this paper, the certification authorities will be responsible for holding the data for a large number of companies. As discussed in the previous section, this data is sensitive and may be damaging if it is leaked to competitors or interested parties. As such, companies will not release their data without assurance that the data is secured from other competitors or interested parties. Further, the certification authorities will normally have confidentiality agreements in place with the aerospace companies to cover this type of confidentiality. However, it is important that independent security and integrity audits are carried out to ensure that the data is held securely and protected from industrial espionage. The certification authorities will be given the privilege of accessing more of a company’s ‘dirty laundry’ than they would normally view. They must respect this arrangement and not abuse this trust. This is a privilege that can also be abused by the aerospace companies. These companies have an insight into what the authorities are looking for at the audits and therefore they could falsify the data to improve how the project quality appears to the outside world. However, it is for the benefit of all the stakeholders that this data collection and analysis approach is successful as it allows for clear and open communication channels between both parties.

The certification authorities must also be careful about their legal obligations. As previously discussed, they may become too involved and implicated if an accident occurs. The certification authorities must clearly define their roles and responsibilities with respect to the collected data and its use. Their independence must be maintained and it must be made clear that they are using the data in an oversight role only. If an accident occurs, then this data would clearly be admissible in a court of law and therefore the aerospace companies may be even more reluctant to provide this data to the certification authorities, for fear that the data might be used against them.

9. Concluding remark

Although the safety categorisation of the problem reports is an important issue, safety-critical software must be maintained and be serviceable for a long period of time. From a maintenance and economic point of view, it would be unwise to undermine the impact of non-safety related issues purely for the sake of certification. The maintenance costs of a poorly documented software package will rise dramatically if the number of problem reports is not kept to a minimum. Problems that are not resolved early in the development lifecycle will have a knock-on effect in the downstream lifecycle stages, particularly in the maintenance phases. It is therefore important to use the method proposed in this paper as part of a balanced approach, where the certification issues are addressed to allow the project to pass the SOL audits and the quality issues are addressed to ensure that the software system is economically viable and maintainable throughout its life.

Acknowledgements

The authors would like to thank Steve Willoughby, Ian Hopkins and the various aerospace project managers and auditors who supported this work and provided the data to allow this research to be tested and validated. The authors would also like to thank Rob Alexander for reviewing and providing feedback on this paper.

References


[6] RTCA. DO178B—Software Considerations in Airborne Systems and Equip-


[23] Society of Automotive Engineers (SAE). Aerospace recommended practice 4754: certification considerations for highly-integrated or complex aircraft systems. SAE; November 1996.