



# Planning for Software Certification in a Schedule Driven Environment

**ADF Software Symposium 2010**

**FLTLT Kristian Cruickshank**

**SCI-DGTA**





# Introduction

- “A good plan today is better than a perfect plan tomorrow” – General G. Patton.
- Upfront planning is often the first casualty when schedule pressures apply to projects. Plans for software certification even more so.
- This brief is intended to show how to strike a balance between a “good” plan and a “perfect” plan for certifying aviation software.
- Only speaking to certification of safety-related\* software, however principles applying equally to mission-related software, security-related software, etc.
- If I lose anyone in SCI “software assurance speak” please pull me up.

\* The term ‘safety-related’ encompasses all levels of software contributing directly or indirectly to system hazards.





# Overview

- **Background to Software Certification**
- **Why plan for software certification?**
- **The perfect plan**
- **The *good* plan**
- **The new software regulations – TAREG 2.2.12 and 3.5.3**
  - PSACs and SCFPs
- **Practical scenarios**
- **Further Information**





# Background





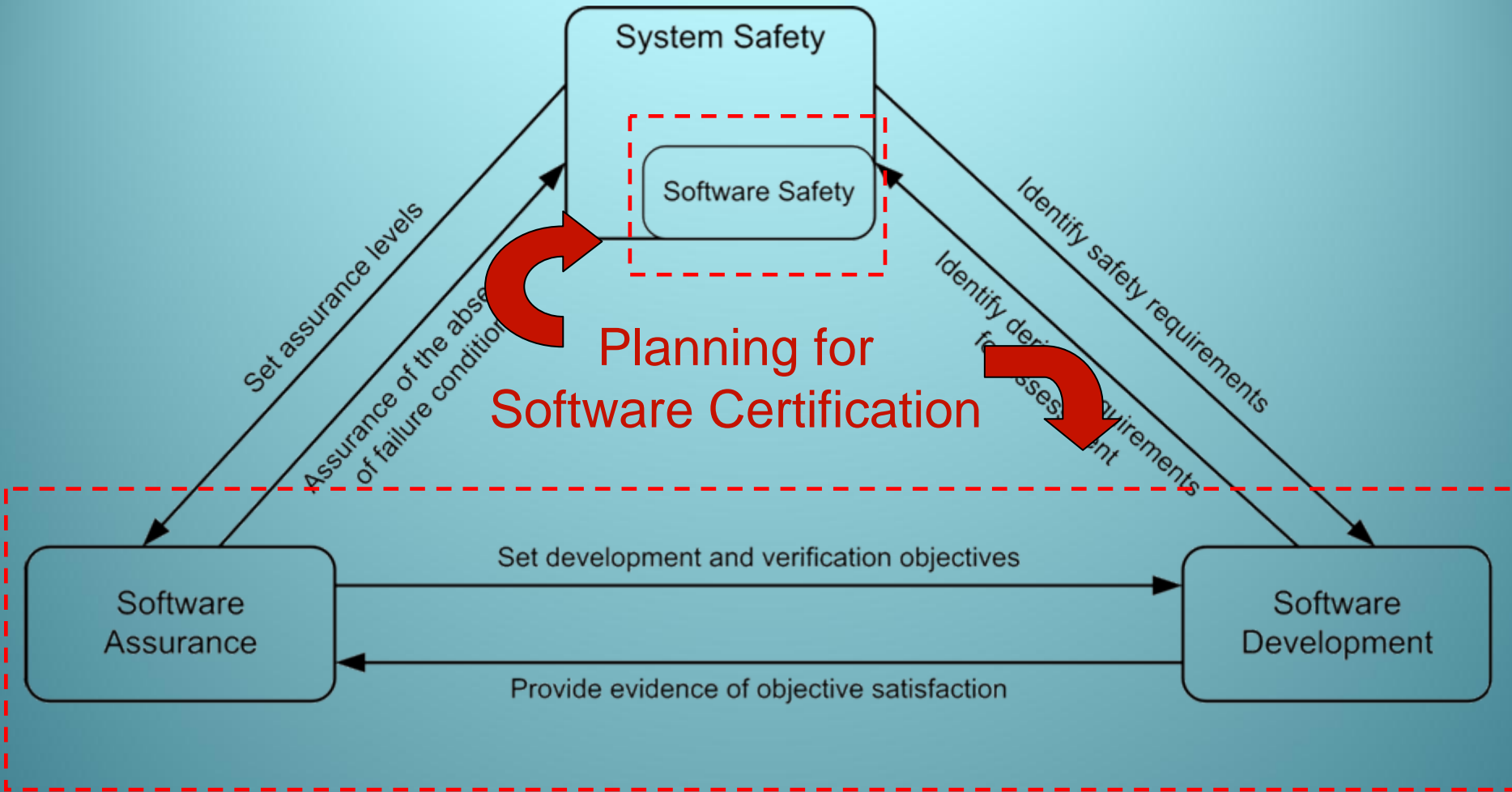
# Background

- **Certification of software takes into account more than an assurance standard.**
- **The software must be designed with appropriate behaviours for the system context, and it must be shown to comply with the designed behaviours.**
  - System/Software Safety, and
  - Software Assurance.
- **Appropriate behaviours determined through System/Software Safety Programs.**
- **Confidence that software complies with designed behaviours is achieved through Software Assurance.**





# Background





# Background

- **Software safety is a topic in its own right, and therefore is not covered in depth through this presentation.**
- **Software safety is generally dealt with in one of two ways:**
  - Through a stand-alone Software Safety Program, or
  - As part of the System Safety Program.
- **DGTA's recommended standard for stand-alone Software Safety Programs:**
  - IEEE 1228 – *Standard for Software Safety Plans*
- **This presentation will focus on the software assurance and development aspects of planning for software certification.**





# Background

- **Certifying aviation software does not occur by coincidence!**
- **An ad-hoc approach to developing aviation software will inevitably result in software that:**
  - Exhibits unsafe behaviours, and
  - Does not function according to requirements.
- **Software in it's own right is complex, therefore plans for it's management will not be simple.**





# Background

*Certifying Aviation Software  
requires Planning*





# Why plan for Software Certification?





# Why Plan for Software Certification?

- **Contrary to popular belief, the aim of developing software certification plans is not to:**
  - Gain TAA and OAA endorsement of the plan,
  - Comply with regulations\* ,
  - Get in the way of doing the real work,
  - Blow out budgets and schedules,
  - Make the regulator happy, etc.

***The aim of a software certification plan is to identify the safety benchmark and show how it is intended to be satisfied. This contributes to SAFETY!***

**\* It just so happens that, usually, regulations are aligned with a valid aim. If complying with regulations does not result in value adding to the project, then it is not valid.**





# Why Plan for Software Certification?

- **A plan for software certification is an agreement with the certification authority that the provisions of the plan are sufficient to support approval of the end product.**
  - Particularly useful for new or novel approaches to development or verification.
- **It provides the certification authority with a level of confidence that the software will be of sufficient rigour.**
- **It provides the applicant with a level of confidence that the software will be of sufficient rigour.**





# Why Plan for Software Certification?

- **It enables the applicant to systematically evaluate their own development and verification practices against the certification authority's requirements:**
  - Determine requirements if not already known.
  - Ensure that software development and verification processes do not conflict with requirements.
  - Ensure that appropriate evidence will be produced throughout the process to satisfy requirements.
  - Identify when, where, and how certification authority will access and review evidence.





# The Perfect Plan





# The Perfect Plan

- **Most widely recognised benchmark for aviation software assurance is RTCA/DO-178B.**
- **This is the ADF's preferred software assurance standard.**
- **For the purpose of this presentation, the “perfect plan” is modelled off RTCA/DO-178B.**
  - *Plan for Software Aspects of Certification.*





# The Perfect Plan

- **A DO-178B PSAC is broadly broken into the following sections:**
  - System Overview (including software)
  - Certification Considerations
  - Software Life Cycle and Data
  - Schedule
  - Additional Considerations
- **The PSAC should be developed as early as possible in the project, but typically not before drafts of other more detailed plans (development, test, etc).**





# System Overview - PSAC

- **The System Overview section of the PSAC provides context for all that follows in the plan.**
- **It is an important section, but should not be the focus of the plan.**
- **The Overview should summarise the functionality of the system and software separately.**
- **System functionality should only be discussed where software exhibits some form of control or input.**





# System Overview - PSAC

***The aim of the System Overview section is to allow the Certification Authority to understand the context of the software operating within the system.***

- No more, no less.
- Recommend that summaries of system descriptions be provided rather than references to in-depth technical documents.





# Certification Considerations - PSAC

- **The Certification Considerations section provides a basis for the level of assurance required.**
- **Propose the certification basis for software assurance:**
  - DO-178B or alternate (DEF-STAN 00-55, TAR approved software assurance matrix, etc)
- **Propose software design assurance levels for each CSCI and provide justification (through System Safety Program).**





# Certification Considerations - PSAC

***The aim of the Certification Considerations section is to identify the proposed level of software assurance.***

- **In the ADF, this may include assurance through standards other than DO-178B.**
- **The System Safety Program will determine the level of assurance.**
  - A complete PSAC assumes the SSP is appropriately mature.





# Software Life Cycle and Data - PSAC

- **THE MOST IMPORTANT PART OF THE PLAN!!**
- Identifies *how* the objectives of the chosen assurance standard will be satisfied throughout the lifecycle.
- Identifies organisations involved throughout and their responsibilities.
- Provides a description of how the applicant intends to show how each objective will be satisfied.





# Software Life Cycle and Data - PSAC

***The aim of the Software Life Cycle and Data section is to describe how the assurance standard will be satisfied.***

- This section is the cornerstone of the PSAC.
- More effort required here than any other part.
- Software Life Cycle Data used for evidence of satisfaction should be identified.





# Additional Considerations - PSAC

- **Next to the Software Life Cycle section, this is the second most important section.**
- **A catch-all section for those aspects of certification that are not covered by other sections.**
- **The most likely “additional consideration” is Tool Qualification.**
  - A topic for another symposium, but essentially where a software tool reduces or automates engineering effort and contributes to satisfaction of an assurance objective – it must be developed to an appropriate level of assurance.





# The *Good Plan*

- Combined with the guidance given in previous slides and RTCA/DO-178B Section 11, a “perfect” plan can be derived.
- What can be considered a “good” plan?
- Depending on the DGTA’s involvement and understanding of the program, it may be possible to reduce the amount of effort applied to each section in the PSAC.





# The *Good* Plan

- Unlike the FAA, DGTA are usually well aware of upcoming projects, their status, and issues they may be facing.
- Through interaction between the project and DGTA, sections of the PSAC may be well understood without providing significant detail.
- Obviously the level of understanding established will be different for each project. The minimum requirements for a “good” plan would be assessed on a case by case basis.





# The *Good* Plan

- Broadly speaking If all supporting plans and programs are sufficiently mature, the PSAC sections should be attacked in the following order:

- Software Life Cycle and Data
  - Additional Considerations (e.g. Tool Qual)
  - Certification Considerations
  - Schedule
  - System Overview
- Bare  
Expected  
Reasonable





# The *Good* Plan

- **Bare Minimum:**

- If DGTA is acutely aware of the systems functionality, software's involvement in the functionality, the System Safety Program, and the project schedule through other plans and documents, it may be possible to develop a PSAC only detailing how assurance objectives will be satisfied.
- For completeness, the other documents and plans that contribute to the Certification Authority's understanding should at least be referenced by the PSAC.





# The *Good* Plan

- **Expected:**
  - Where DGTA have not necessarily had oversight of the System Safety Program to a level where the justification of software levels is obvious, or where the software levels determined by the System Safety Program are somewhat unique (i.e. not the norm for certain functionality), the PSAC should include a summary of the justification determined by the System Safety Program.
  - Including this summary in the PSAC is expected since it consolidates the reasoning for the level of assurance with details on how the objectives for that level will be satisfied.





# The *Good* Plan

- **Reasonable:**

- DGTA will always have at least some understanding of the project.
- In addition to the above sections, to provide a reasonably detailed PSAC, the applicant should include:
  - An accurate schedule identifying when DGTA should be involved in software certification activities.
  - A very brief overview of the system with referral to more detailed system and software description documents.





# The *Good* Plan - Summary

- Since the PSAC is an agreement between the Applicant and Certification Authority, the level of detail required for any “Good” PSAC rather than “Perfect” PSAC, will depend on the Certification Authority’s understanding of the project.
- The previous slides provide guidance on what DGTA would consider suitable in a schedule driven environment.
- Any proposed PSAC’s containing less than the guidance provided are unlikely to be approved.





# TAMM Software Regulations

- Besides all the possible benefits, developing a PSAC will still divert my resources from other engineering effort. Why should I develop a PSAC?
- TAREG 3.5.3.d.(3).(i): The procedures to assure software integrity of new or modified software shall ensure a PSAC, or equivalent document, is submitted to and approved by the TAR prior to commencement of development if the worst credible failure condition of the software is more severe than Minor (AC25.1309) or Marginal (MIL-STD-882C).





# TAMM Software Regulations

- **Why? In short:**
  - Because the TAR says so.
- **The long version:**
  - SCI has experienced an increasing trend for aviation software to be developed poorly.
    - No identification of suitable benchmark,
    - Lack of evidence that benchmarks are satisfied,
    - Unorthodox development methods for the aviation industry,
    - Poorly managed software development and assurance programs, etc.
  - Developing a PSAC is one method to avoid the increasing trend.
  - Unlike most others, this regulation is not the result of an ADF mishap.
    - The ultimate aim is to develop safe systems!





# A Practical Guide

- **Scenario 1 – A project is about to be established to acquire an Autopilot System replacement (the current is no longer supportable) for an ADF platform. The failure conditions of the software are found to be Hazardous. How do I go about compiling a PSAC?**
- **Scenario 2 – I've just been posted into AIR XXXX and I've had a poor handover/takeover. What should I look for to gain some confidence that the software design is being appropriately managed?**





# A Practical Guide - Scenario 1

- **First things first – Make sure the contract includes the requirement to assure the software in accordance with RTCA/DO-178B to the level determined by the System Safety Program. DO-178B requires a PSAC to be developed.**
  - Alternatively – make sure the contract includes “equivalent” requirements for a PSAC.
  - For “equivalent” requirements use DO-178B as a guide, this presentation as a guide, and talk to SCI1-DGTA.
- **Second – Make sure the contract includes access provisions to the data used in the PSAC!**





# A Practical Guide - Scenario 1

- **A PSAC cannot survive on its own!**
  - Many other requirements of the contract need to be satisfied to a reasonably mature state to develop the PSAC.
    - System and Software Descriptions
    - System Safety Program
    - Software Safety Program
    - Software List
    - Etc





# A Practical Guide - Scenario 1

- **As described in previous slides – start with the most important parts first.**
  - Software Life Cycle and Data
  - Additional Considerations (e.g. Tool Qual)
  - Certification Considerations
  - Schedule
  - System Overview
- **Assumption at this point is that all required “enabling” information is complete.**
  - Assurance level is known and justified.
  - Software List is complete (to the extent that all major CIs are identified – including software tools).
  - System description and software description are sufficiently detailed.
  - Software management plan is documented.





# A Practical Guide - Scenario 1

- **The PO should review the PSAC before submitting to the TAR.**
  - **If the PO is not willing to stand behind the product, why should DGTA be willing to review it?**
  - Typically the first go at a PSAC will require some rework.
  - The review should compare the draft PSAC with the requirements of the assurance benchmark.
    - Does it explain how assurance objectives will be met?
    - Does it justify the level of assurance?
    - Does the system and software overview provide sufficient context for the reviewer, i.e. are all system functions accounted for in the justification for level of assurance?
    - Do you expect DGTA to agree with the proposed approach?





# A Practical Guide - Scenario 1

- **Post DGTA approval:**
  - Make sure that assurance objectives are satisfied in accordance with the plan.
  - Any minor departures from the plan are likely to be tolerated as long as the changes are documented and DGTA is informed.
  - Major departures from the plan will need to be discussed with DGTA as soon as practical.
  - A Software Compliance Finding Plan (in accordance with TAREG 2.2.12) should be developed as soon as practically possible following PSAC approval.
  - The SCFP should be modelled around the PSAC – looking at evidence of compliance that was identified in the PSAC.





# A Practical Guide - Scenario 1

- **Post DGTA approval (cont):**
  - The result of the SCFP and PSAC should be a Software Accomplishment Summary (SAS).
  - The SAS may be included as part of the broader design Accomplishment Summary (or equivalent), however it is often easier to break it out into its own document.
  - The SAS should show how assurance objectives have been satisfied.
    - Essentially PSAC changed to past tense and referring to the evidence of satisfaction.





# A Practical Guide - Scenario 1

## Note on PSACs and the TAMM

- TAREG 3.5.3.d.(3).(i) only requires a PSAC be **submitted** to the TAR where the worst credible failure condition is more severe than Minor.
  - Scenario used 'Hazardous' failure condition.
- The TAREG does not prevent the DAR from requiring a PSAC where failure conditions are less than Minor.
- In this case, the DAR would be responsible for approving the PSAC.





# A Practical Guide - Scenario 2

- *I've just been posted into AIR XXXX and I've had a poor handover/takeover. What should I look for to gain some confidence that the software design is being appropriately managed?*
  - Assumption is that the project is at the stage where a PSAC has been developed or is at least somewhat mature.
- **Find the PSAC:**
  - Look for approvals from DAR and TAA
  - Look to see any review recommendations have been incorporated, or justification as to why not.





# A Practical Guide – Scenario 2

- **System and Software Overviews** should provide a good introduction of the functionality and design that impacts safety.
- The **Certification Considerations** section should enable you to understand how safety critical the software is (through required assurance levels), and will point you to the appropriate certification basis assurance standard.





# A Practical Guide – Scenario 2

- **The Software Lifecycle and Data section will describe to you the contractor's plan for satisfying the assurance standard objectives.**
  - This section will be the focus for a large portion of software activities following the initiation of development.
  - The results of each 'planned' activity for showing satisfaction should be recorded against each objective and will form the basis of the Software Accomplishment Summary.
  - Any major deviations from the 'planned' activities should be noted and discussed with DGTA.





# A Practical Guide – Scenario 2

- **Don't forget the Additional Considerations section!**
- **Inevitably Tool Qualification will be required (usually).**
  - Sometimes can be satisfied through a Tool Qualification Plan – essentially a PSAC for software tools – all the same considerations as a normal PSAC except scoped to the requirements of tool qualification.
  - Progress against objectives should be documented similar to SAS.
  - Tools should be assured before being used for development.





# A Practical Guide – Scenario 2

- **Other aspects of Additional Considerations likely to be encountered:**
  - Previously developed software
  - Formal methods
  - Multiple-version dissimilar software
  
- **Most likely Additional Considerations for ADF:**
  - Tool qualification
  - Previously developed software





# Further Information

- RTCA DO-178B Software Considerations in Airborne Systems and Equipment Certification
- RTCA DO-248 Final Report for Clarification of DO-178B
- FAA Order 8110.49 Software Approval Guidelines
- AAP 7001.054 Airworthiness Design Requirements Manual Section 2 Chapter 7 *Aviation Software*
- AAP 7001.053 Technical Airworthiness Management Manual Section 2 Regulation 3.5.3 *Software Integrity Management*





# Summary

- TAREG 3.5.3 requires a PSAC be approved by the TAR (for worst credible failure conditions above Minor or Marginal).
- RTCA/DO-178B provides the requirements for a “perfect” PSAC (requirements are equally applicable under different assurance standard).
- There is some latitude for a “good” rather than “perfect” PSAC – only applicable where other project/contractor documents supplement the PSAC.
- Documenting a PSAC increases the likelihood that software will be appropriately assured, and that the Certification Authority (DGTA) will endorse applications for ADF airworthiness instruments.





# Questions?

