

Static Code Analysis on the C-130J Hercules Safety-Critical Software

Eur Ing K J Harrison, BSc CPhys MinstP CEng MRaES MBCS; Aerosystems International, UK

Keywords: Software Safety

Abstract

The UK Ministry of Defense (MoD) has taken delivery of the first two C-130J Hercules aircraft both of which contain a significant amount of safety-critical software. The decision by the UK MoD to purchase new C-130Js with new avionics systems was made in late 1994. The UK MoD required that all the safety-critical software undergo third party assessment using Static Code Analysis (SCA). This had never been performed before on such a large scale. The C-130J safety-critical software is contained within 23 Line Replaceable Units (LRUs) and consists of approximately 500K Source Lines Of Code (SLOC) developed by 18 vendors in Ada, C, LUCOL, PLM and various assembler languages. In early 1995, a team comprising of Aerosystems International (AeI) and Lloyd's Register (LR) were selected to perform this assessment. At the time the most accepted tools to perform static code analysis were the MALPAS and SPARK toolsets. A process was developed and has since evolved to use these tools and perform SCA on several iterations of each safety-critical Line Replaceable Unit. This paper describes the process (including managing data, Goal Directed Approach, Static Code Analysis, Sentencing and Regression Analysis), difficulties and statistics of the project.

Introduction

In late 1994, the UK Ministry of Defence (MoD) made a decision to purchase 25 C-130J Hercules transport aircraft for the Royal Air Force (RAF) from Lockheed Martin Aeronautical Systems (LMAS), Marietta, Georgia, USA. The C-130J Hercules is a newer variant of the old C-130 aircraft. The significant difference is the addition of a whole new avionics system, which includes newly designed software controlled electronic LRUs.

In early 1995, the contract to carry out third party assessment of all the safety-critical software on the Hercules aircraft was awarded to Aerosystems International (AeI). AeI subcontracted part of analysis work to Lloyd's Register (LR). This paper reports on the process,

the difficulties and the accomplishment of the project.

At the time of writing this paper, all the software systems have been assessed at least once and in some cases as many as six times. The first two aircraft have been delivered to the UK and are undergoing flight trials at the Defence Evaluation and Research Agency (DERA), Boscombe Down.

Outline of Hercules C-130J SCA Project

Aircraft Safety Assurance

The UK MoD requires assurance that the aircraft is safe to operate and fly. The MoD is required to issue a "Military Certificate of Airworthiness" to the RAF before the aircraft is accepted. The MoD are assisted and advised by specialists at DERA, Boscombe Down.

DERA is responsible for making recommendations to the certifying authorities that the aircraft is safe. This recommendation includes the safe operation of the software.

The general view on safety-critical software is that it should be developed to the appropriate level using the principles of the RTCA/DO178-B guidelines. Furthermore, the delivered safety-critical software should be subject to Static Code Analysis (SCA).

Scope of Project

SCA was to be applied to all the software on the C-130J aircraft classified as RTCA/DO178-B Level A and B. All other software classifications were not deemed safety-critical and therefore not assessed in this way.

There are 23 Level A and B software systems on the C-130J ranging in size from 1,500 to 180,000 source lines of code. These software systems, known as Line Replaceable Units (LRUs), have been developed by 18 US/UK vendors (LRUVs).

The total amount of source code was about 500K lines, written in a number of languages: Ada, C,

LUCOL, PL/M and various assemblers. Over one third of the Ada was written in the SPARK safe subset of the Ada language.

Since SCA is a labour-intensive activity, Ael provided a large team of analysts peaking at a total of 50 staff.

Reporting of Assessment results

Any potential error discovered during the analysis process was raised as an “anomaly”. All anomalies were stored in a database to facilitate subsequent investigation and discussions. All anomalies were evaluated for safety impact via an assessment panel meeting attended by a Safety Manager and Chief Engineer.

The main purpose of the project was to identify any safety-threatening anomalies so they could be rectified by the LRUV prior to final delivery of the aircraft. This was a pre-requisite for MoD acceptance.

Anomalies not deemed safety threatening were also reported to the LRUV as a further aid to their product improvement.

SCA Tools

Available SCA Tools

In principle, static code analysis may be carried out by many software tools, including some compilers, but most only cover control flow and data use analysis. The C-130J Hercules was required to carry out a more detailed analysis, namely, syntax analysis (control flow, data use and information flow analysis) and semantic analysis (comparison of path function against design). The analysis of the SPARK further required formal proof of correctness of the code against a semi formal specification. At the time, there were only three toolsets that were acceptable to the UK MoD and DERA. They include:

- MALPAS MALvern Program Analysis Suite, available from TA Consultancy Services
- SPADE Southampton Program Analysis and Development Environment, available from Praxis (Bath);

- SPARK SPade Ada Kernal, available from Praxis (Bath).

The SPARK examiner was used to analyse the SPARK with the SPADE automatic simplifier and interactive proof checker to complete the analysis. All other software was analysed using the MALPAS tool.

MALPAS Toolset

The MALPAS suite of tools reads software written in a standard language known as Intermediate Language (IL). Hence, any software to be analysed using this tool first needs to be translated into an IL model. This is very flexible since MALPAS can then be used to analyse any software language. On the other hand the need for translation can make it more difficult to relate assessment results to the original source code. Furthermore, because the scope of the IL model is restricted to analysable features, translation is not an exact process (5.4.1). It is both expensive and error prone.

MALPAS comprises of the following analysers:

- Translation
- Control-flow
- Data-use
- Information-flow
- Semantic (path function analysis)
- Compliance (not used on the C-130J SCA project)

SPARK

The SPARK examiner reads SPARK code directly giving control flow, data use, information flow and Verification Condition (VC) and Run Time Check (RTC) information. The SPADE automatic simplifier is then used to simplify the resulting VCs and an interactive proof checker is then used to prove the VCs and show that the source code matches the semi formal specification.

Loop assertions, pre and post annotations are added to the source code to represent the formal specification. These annotations are used to generate the VCs.

The SPARK examiner was only used on the SPARK subset of the Ada language. All other

Ada on the project was analysed using MALPAS.

Organisation

The major associated functions to manage such a large project is illustrated in figure 1 below. They include:

Project Management Function, a Project Manager with responsibility for the completion

The safety function is also responsible for performing a Safety Review and for advising the assessment and analysis function on the safety implications of anomalies found.

This function also has the important role of interfacing with the customer safety function, and is involved in providing support to the producers of the overall aircraft or system safety case.

Assessment and Analysis Function, lead by a

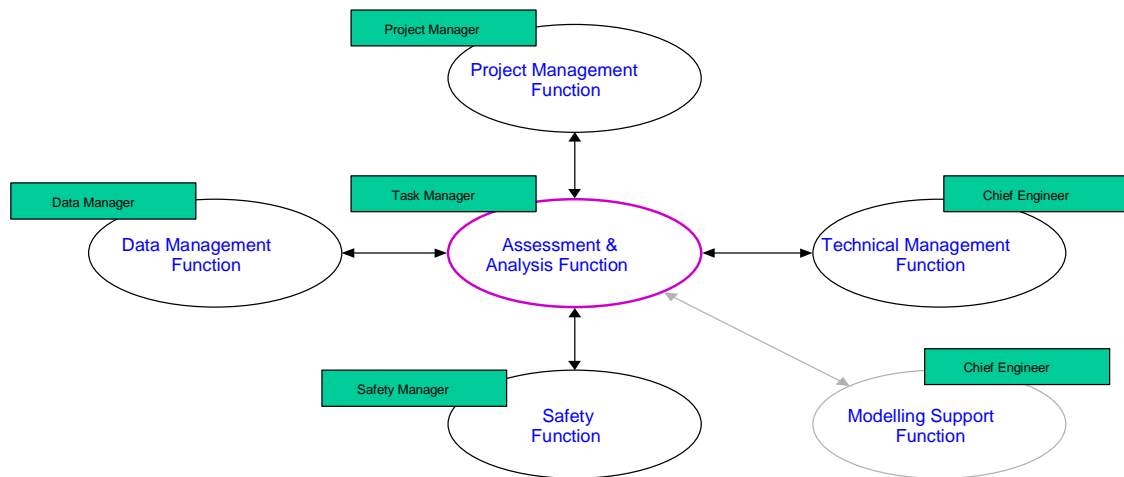


Figure 1 - C-130J Functional Organisation

of all assessment activities and the primary interface to the customer, providing regular reports to both the customer and Ael management.

Technical Management Function, lead by a Chief Engineer with responsibility for all technical aspects of the project. This function will also ensure that work is carried out to the required technical standards, by participating in review activities.

Data Management Function, a Data Manager with responsibility for liaising with the software supplier during the course of the analysis process and for receipt and distribution of all the LRUV data and queries.

Safety Function, lead by a Safety Manager with responsibility for providing expert input on system safety issues pertinent to the assessment.

Task Manager with responsibility for the assessment of a LRU. The Task Manager lead an analysis team, with the necessary skills and experience to perform SCA. Analysts required experience in the relevant analysis techniques and tools, in the application, and in software development and verification.

Modeling Support Function, lead by the Chief Engineer with responsibility for the MALPAS modeling strategy.

The majority of the effort on the project was spent running the C-130J software through the SCA tools and interpreting the results. However, this was only a part of the overall SCA process.

Preparation

Figure 2 details the preparation required to perform SCA.

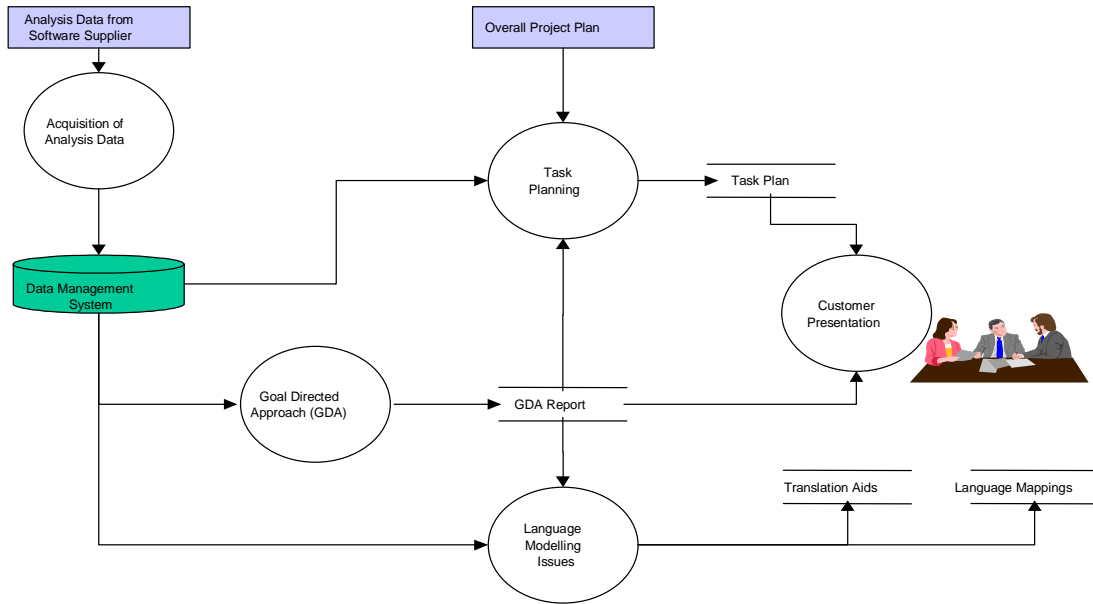


Figure 2 - Preparation

Data Acquisition

The success of project required support and co-operation from the suppliers. This required timely provision of design information, source code and responses to data queries. The commercially sensitive nature of this information meant that the assessment team not only had to maintain confidentiality, but also to be seen to do so. A Data Management function was set up to ensure safe and secure handling of suppliers data. The Data Management function is responsible for:

- Liaising with the supplier to resolve commercial issues such as Technical Data Licenses and Confidentially Agreements;
- Initial receipt and storage of supplier data by the project;
- Checking its completeness and suitability for analysis;
- Supplying the information to the appointed Task Manager for analysis;
- Handling queries about the supplier data;
- Return or controlled destruction of supplier data following analysis.

The software was delivered normally after thorough testing. In addition to the source code, the supplier provided some or all of the following documents:

- Software Requirements Specification;
- Software Design Document;
- Interface Design Documents;
- Version Description Documents.

These were used to support the assessment process.

The project experienced various problems with supplier data:

- Software and documentation arrived later than expected. This sometimes made it difficult to organise effective resourcing.
- Software was sometimes further modified after initial delivery. Although it was beneficial to have early code versions so that the analysts could make an immediate start on the assessment, this was offset by the need to carry out regression analysis on a later version.
- Software and documentation were sometimes “out of step”. Much time was spent investigating whether the anomalies were caused by software error or documentation error.

Task Planning

Detailed planning was necessary for all LRUs, since the assessment of even the smallest system

required several analysts. Furthermore, the breakdown of the activities allowed progress to be monitored against plan.

It should be noted that the unit of analysis in MALPAS is the procedure/function. Also, MALPAS must be applied to the software in a “bottom-up” fashion, in that any procedure called must already have been analysed.

The assessment of one or more procedures was known as a “subtask”. All subtasks within a particular LRU were identified at the outset, although the assignment of a subtask to an analyst was normally carried out dynamically. This meant that overall progress was less constrained by unexpected variations in code complexity or individual analyst productivity rate.

Goal Directed Approach

The Goal Directed Analysis (GDA) stage of the process has several objectives:

- To identify LRU software safety properties which might cause the aircraft-level hazards;
- To give analysts an understanding of the system;
- To focus upon safety;
- To ensure that a useful analysis was conducted, regardless of the level of quality of documentation;
- To define which parts of the LRU system and software can contribute to the hazard.

The starting point for the GDA was the resultant systematic hazard analysis at the aircraft level, carried out by a separate “safety” team as part of the overall aircraft safety case for the Hercules aircraft. This was used to define a hazard table for each LRU. Two specific techniques were used during the GDA process:

- Fault Tree Analysis (FTA). A tree was produced with each LRU hazard as the top event. These were refined into sub-events, each related to a particular software function. A function that did not contribute to a hazard was not deemed to be safety-critical.
- Functional Block Analysis (FBA). The software was examined at the architectural level in order to identify software functional areas. The effects of certain standard failures

(function occurring too early, too late, being interrupted or having a coding error) were then considered on all the functional areas. The results of the FTA were used to describe the system level failures.

The findings of the GDA were summarized in a report, which provided a logical connection to the SCA assessment team and the safety case team

Language Modelling

Early assembler and C LRUs were generally translated by individual analysts at the subtask level, using the translation aids described previously. The resulting IL was then reviewed in order to confirm that it was a sound model of the original source code.

Ada translation was carried out in bulk by a specialist team, using a set of standard pre-translation edits prepared for the LRU. It was still necessary for individual analysts to make post-translation edits, but this strategy was more efficient the process of individual translation. The resultant IL model for functions was still reviewed. Later the assemblers of LRUs were also translated this way.

Translation strategy and language issues

Since MALPAS only accepts software written in IL, it was necessary to translate the original source code prior to analysis. For some languages, automatic translators were available. Translator tools were procured for Ada and 8086 assembler. However, it soon became clear that a translator is not in itself sufficient to produce the most suitable IL model. Furthermore, some of the C-130J software was written in 80186 and 80386 assembler, which incorporates instructions not available for the 8086. Thus it was necessary to spend a significant amount of time considering the translation strategy for each language.

Strategy for Ada

The MALPAS “automatic” Ada translator was found to be suitable for the majority of (non-SPARK) Ada, but it was necessary to modify the Ada code in areas such as the following:

- Generics, including
UNCHECKED_CONVERSION;

For all other languages including C and non

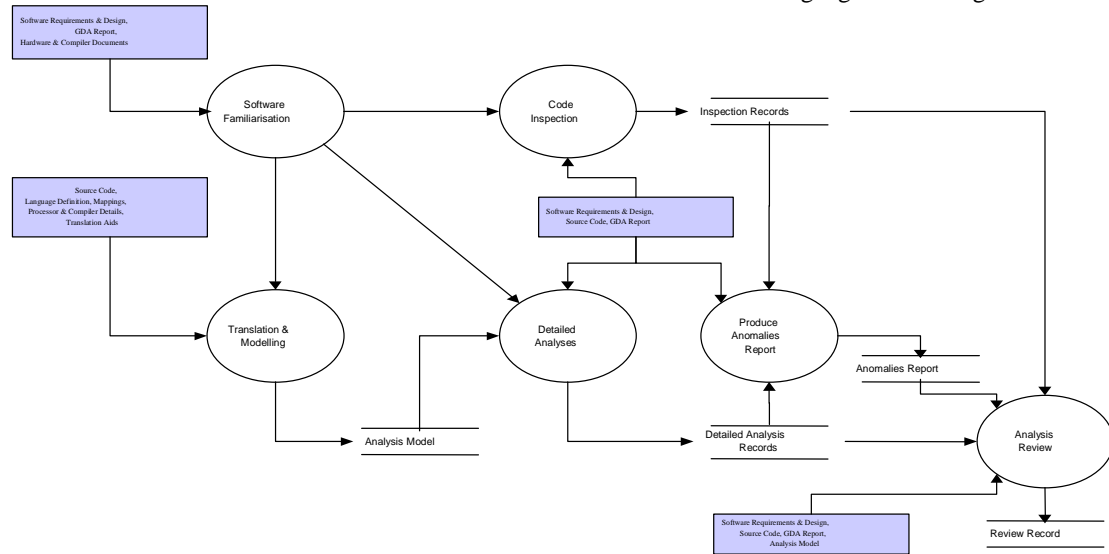


Figure 3 - Analysis Process

- Tasking;
- Addressing;
- Separates;
- Pragmas;
- Interface with other languages.

After carrying out standard “pre-translation” edits for these Ada features, the code was readily accepted by the Ada translator, which produced the IL model corresponding to the original Ada. After this it was generally necessary to perform “post-translation” edits such as the following:

Add file inclusion directives in order to access previous analysis functions;
 Add or modify the access modes of IMPLICIT parameters (non-locals);
 Add IL declarations of types or constants;
 Model untranslatable code;
 Rectify translation errors.

From the above it should be concluded that even an “automatic” translator is in reality no more than a translation aid, since it is impossible to formalise the translation from a large language such as Ada. However, the “automatic” Ada translator was found to be extremely useful, in that it improved efficiency of translation as compared to translation by hand.

Strategy for C and other languages

8086 assemblers a translation aid was written using either LEX/YACC or DCL command procedures. Although some were fairly primitive, they served their purpose in providing a better and faster translation than could be achieved by hand.

C pointers were abstracted in the IL model, whereas assembler language registers and flags were modeled exactly.

Customer Presentation

The GDA report and modeling and analysis approaches were presented to the MoD and DERA. This detailed any issues relating to a particular LRU and any necessary deviation from the procedures.

Static Analysis Process

Figure 3 details the analysis process for both MALPAS and SPARK.

Software Familiarisation

This was the first task in the SCA process. For each procedure to be analysed, the allotted analyst reviewed all available information (including the safety properties identified in the GDA) to gain familiarity in the subtask. The analyst then performed a code walkthrough to

compare the code against its design information, noting any discrepancies for further investigation in subsequent SCA.

This was a very useful activity, which found approximately on average 60% of the anomalies. Furthermore, the code walkthrough provided a good basis for further familiarisation during the analysis stages (6.3.1 and 6.3.2).

Code Inspection

This activity was performed for each package or source file to be analysed, and for each procedure or function contained within them. It consists of a systematic checklist-based inspection of the code against its design information and requirements, noting any anomalies or discrepancies found. An essential element of the code inspection is the provision of evidence of what has actually been checked.

Detailed Analysis

MALPAS detailed Analysis

Syntactic analysis and preliminary PROCSPEC

At this stage in the process the analyst ran the “syntax” or flow analysers (control flow, data use, information flow) on the translated IL model and examined the results.

Particular attention was paid to sets of “possible errors” identified by the data-use and information-flow analysis. The analyst was required to annotate each observation, either reporting an anomaly or explaining why there was no problem.

The results of data-use and information-flow analysis were used to construct the preliminary specification for each procedure (PROCSPEC, in MALPAS terminology).

Early on in the project there would be a review at this stage in the process. This would allow the Task Manager to check that the analysis team were following the correct approach. Later in the project, this was found to be unnecessary.

Semantic Analysis and definitive PROCSPEC

The analyst now ran the translated code through the semantic analyser, in order to convert the sequential logic of the program to parallel logic.

For each loop-free region of the procedure, the semantic analyser lists the predicates required for each path, together with the assignments taking place. The analyst was required to annotate each path, preferably with a reference to the documentation describing its action.

The analyst also checked for dynamic halts and run-time errors. The Ada translator was particularly useful in the latter respect since any possibility of constraint errors was indicated by an assignment of a specific translator-generated variable (e.g. `cons_err_1`).

The analyst refined the preliminary PROCSPEC so that it could be used during analysis of any ensuing procedures which might call it. The PROCSPEC was also annotated to describe its function and tested.

SPARK Detailed Analysis

The SPARK assessment is very similar to the MALPAS process, however, the code needs to be readable by the SPARK examiner it can be analysed.

SPARK Annotation

The SPARK Ada may need to be annotated for proof checking. The code will need, if not present, loop assertions, pre and post conditions added to facilitate the proof checking.

Control Flow Analysis

The SPARK examiner will check for and report on any control flow anomalies. The analyst needs to collate these anomalies and add them to the Anomaly Report (AR). Due to the nature of the SPARK Ada language data use and information flow errors cannot occur.

Proofs and Run Time Checks

Proofs are performed by showing that the Verification Conditions (VCs) generated by the simplifier satisfy the semi-formal specification. Run Time Checking (RTC) is based on semantic analysis and format proof concepts and used to show the absence of errors due to overflow and exceeding the range of data types and subtypes.

Produce Anomalies Report

All anomalies were given an initial classification by the analyst, using the following categories:

Category	Meaning
1	Code anomaly which could threaten safety in a direct manner
2	Code anomaly which could threaten safety in the presence of other errors/failures
3	Code anomaly which have no bearing on safety
4	Documentation or code commenting anomaly (minor typos were generally ignored)
5	Lack of documentation precludes effective analysis
6	Unclassified – when the analyst could not decide on the category

Anomalies in categories 3 and 4 comprise around 51% of the total.

Because anomalies were discussed by project staff who were not involved in the analysis of the LRU, it was important that they were recorded clearly. Except for categories 3, 4 and 5 anomaly descriptions were partitioned into, summary, effect (on safety) and details.

In addition, analysis of source code was frequently facilitated by the use of assumptions, which were categorised as follows:

Category	Meaning
AS1	Where incorrectness of the assumption might threaten safety
AS2	Where incorrectness of the assumption had no bearing on safety

The GDA results were used by each analyst to support the above classifications.

Analysis Review

Reviewing was performed at all major stages of the process.

Completion Process

The analysis completion process is shown in figure 4.

Assessment Panel Review

When SCA for each subtask was complete, the LRU Task Manager collated the anomalies, with the aid of a database. After this, there was a meeting of the Assessment Panel (AP), comprising the Task Manager, Chief Engineer, Safety Manager and Project Manager (or appropriate deputy).

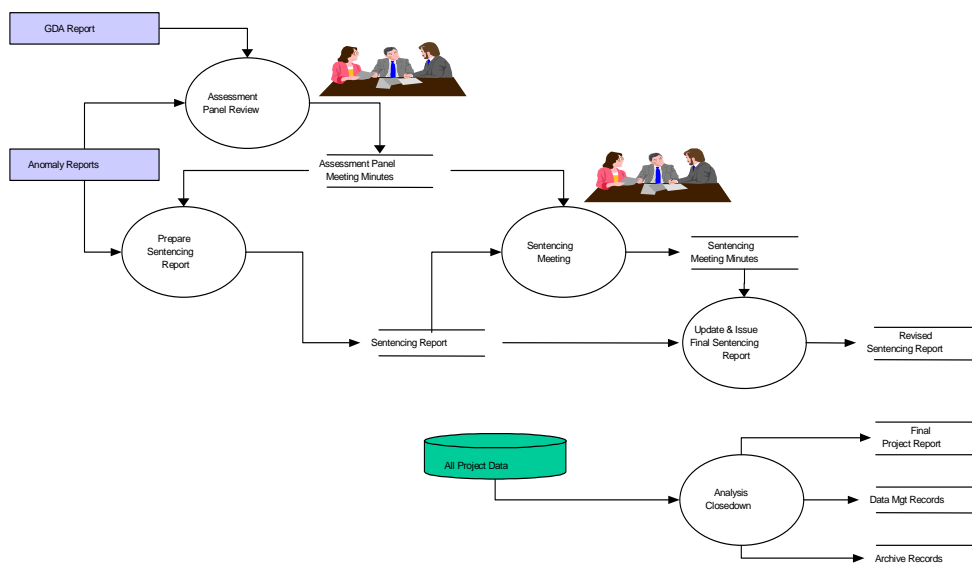


Figure 4 - Analysis Completion Process

Each anomaly classified as 1, 2, 5 or 6 and each assumption classified as AS1 were reviewed and placed in one of the following categories:

- Potentially safety threatening anomaly;
- Maintenance issue;
- Other LRU issue;
- Assumption to be confirmed by LRUV;
- None (when it was agreed that safety could not be affected).

For each safety threatening anomaly or assumption, the Task Manager prepared written queries to the LRUV, requesting clarification of the issue or confirmation of any assumption.

Prepare Sentencing Report

Following the response of the queries, any remaining issues would be written in a Sentencing Anomalies Report (SAR). This document (a main deliverable) was sent to all interested parties, namely LMAS, DERA and MoD.

Because of the late pressures on the program, the above activities were considered in a number of stages. Furthermore, not all queries were answered prior to the delivery of the SAR. This resulted in teleconferences with the LRUV, sometimes only hours before the SAR was discussed.

Sentencing Meeting

Approximately 2 weeks after the issue of the SAR for a particular LRU, there was a Sentencing Meeting (SM), attended by LMAS, DERA, MoD and relevant project staff. The purpose of this meeting was to consider the results of the LRU analysis. All issues relating to safety, maintenance and other LRU issues (detailed in the SAR) were discussed in detail, together with a sample discharged from the assessment panel meetings. The term "sentence" was given to the act of deciding whether an anomaly should be highlighted in a Sentencing Completion Report (SCR) or not. All attendees of the meeting agreed the final classification of any remaining issues.

Update & Issue Final Sentencing Report

The result of the meeting was reproduced in the SCR (final deliverable) and distributed to all parties including the LRUV.

Analysis Closedown

This activity involves the formal return and/or destruction of supplier proprietary data (including all copies of this data) provided by the software supplier(s), subject to project specific arrangements.

- Data Management records were updated accordingly.
- The Data Manager is responsible for archiving analysis results and associated data for the agreed time.

Regression Analysis

Regression analysis iterations will repeat some or all of the steps of the original analysis, based on an initial assessment of the impact of change. Activities will, in general, follow the same methods as those used originally. Key elements of a single iteration are:

- Input Data Management;
- Impact Assessment;
- Re-Planning Analysis Activities;
- Regression Analysis;
- Sentencing Process.

Conclusions from the impact assessment would normally have taken one of the following forms:

- No Re-Analysis Required
- Inspection Analysis Required
- Partial Re-Analysis Required
- Full Re-Analysis Required

8.1 No Re-Analysis Required

Where the changes to the system were deemed to be non-safety-related or cosmetic in nature, the only required action is to record that the changes to the system (or part of the system) have been examined, and that no safety impact had been identified.

Inspection Analysis Required

Where the changes to the system were deemed simple in nature, regression analysis was based on inspection of the changes and their impact on related code, and on the documentation. Note:, it is important that the justification is sufficient to

accept that the inspection can be performed with minimum technical risk.

The objective of re-analysis by inspection is to confirm that the changes have been implemented correctly, are self-consistent and are semantically correct. The analyst must also check whether any anomalies raised on the original release were cleared.

Partial Re-Analysis Required

Where there was a change in software structure then the relevant SCA process for MALPAS or SPARK was applied, making use of all the previous SCA results. In practice, the SPARK regression analysis had to re-run all the proofs.

Full Re-Analysis Required

Full Re-Analysis was required to be performed on new functions or if there was a major change in structure. The relevant assessment process was re-applied to the new/altered code. Complete re-analysis of an old function made use of all previous work, but it had to be re-validated before being used.

Difficulties

One of the major difficulties in the program was the acquisition of data from the suppliers.

- Readability of data;

- Documentation;
- Helpfulness of suppliers;
- Timely deliveries.

In the early stages of 1995, the suppliers were told by a SCA “road show” (visiting each supplier and presenting what SCA was all about and what was required) that their software was to be assessed by a third party. Most of the suppliers were not convinced of the usefulness of this analysis. Over the project life time most of the suppliers now treat the assessment process as a useful addition to their certification activity. Many now log the queries raised by AeI as an input to their internal fault diagnosis process.

The other difficult aspect of the process was the translation process needed for the MALPAS toolset. The use of pointers in C and generics etc in Ada required a lot of effort to produce a convincing model of the original source code.

Statistics

There are currently 11,590 entries in our anomaly database. These anomalies are the original analysis results from the teams of analysts on all LRUs over a number of iterations.

The anomaly classifications have all been collected and figure 6 shows a graph of totals for each category. It can be seen that more than 50% of the anomalies are category 3 and 4. Categories 1 and 2 amount to only 6%.

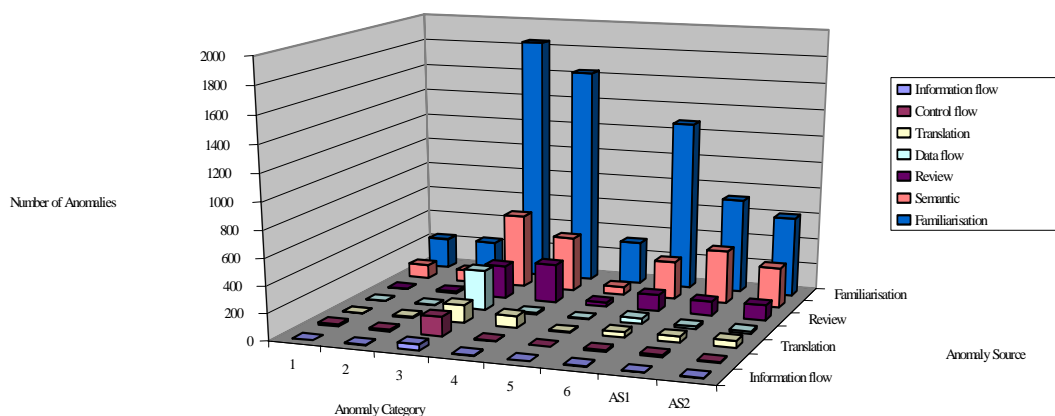


Figure 5 – Anomaly Category by Analysis

The source of the anomalies found is shown in figure 5. It shows that most of the anomalies were found during familiarisation. These were mostly the category 3s and 4s. The semantic analysis found a significant number of anomalies which, were considered to be safety related.

Benefits

Although the SCA of the safety-critical software for the C-130J Hercules aircraft has taken a lot of time and effort to perform. It has given many benefits to LMAS, MoD, DERA and the LRUVs. They are:

- Given LMAS an independent view of their LRU vendors software development process.
- Increased assurance to MoD and DERA.
- Significantly more paths through the software have been addressed.
- Errors found during the analysis process can assist the LRUVs in their development process.
- Found apparent safety anomalies with the software before they reached the aircraft.

Conclusions

The analysis of the safety-critical software was performed with the assistance of the C-130J safety case team. Any future project would require the involvement of a similar interface to the safety case of the aircraft.

Although the SCA of the C-130J safety-critical software was performed post development it added value to the development and safety process.

However, a far better approach would be to implement the SCA approach into the development life-cycle. This would then integrate the safety process at an earlier stage. This could be achieved within the budget for verification and validation.

The analysis was only effective with the involvement of the aircraft prime contractor and all the LRUVs – only they know the whole avionics system context.

One final conclusion from the program is to state that code walkthroughs are very powerful if used in a rigorous process.

Biography

K J Harrison, Project Manager, Aerosystems International, West Hendford, Yeovil SOMERSET, BA20 2AL United Kingdom, Telephone 44 1935 443000, facsimilie 44 1935 443111, e-mail keith.harrison@aeroint.com

Keith Harrison is a Project Manager at Aerosystems International and has been involved with the C-130 SCA project from the beginning. He has spent more than 12 years in testing and analysis of safety-critical software on various projects. He is also a Chartered Physicist and a Chartered Engineer.

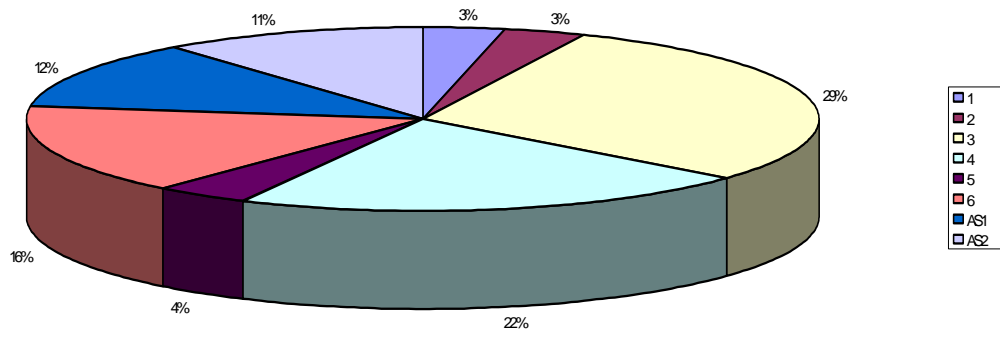


Figure 6 - Total Anomalies by Category