The Practical Application of Safety Techniques on an Ada based Project

Rupert Brown

Logica UK Ltd., Byron House, Business Park 4
Randalls Way, Leatherhead
Surrey, KT22 7TW, England

E-mail : brownrw@logica.com

Abstract

Two techniques are used to analyse a computer based system from a safety perspective. The first identifies those user requirements which encapsulate the safety-related nature of the system. These requirements are tracked through analysis, design and coding, resulting in the identification of safety-related components throughout the project lifecycle. The second technique seeks to examine the effect of failures of system components as they emerge from the design activity. The results produced from both activities are compared and combined. Design changes are made to eliminate "weak points", and the degree to which components can affect safety is constantly monitored. Safety-related components are handled with greater care and subjected to more intense development and testing rigour than non safety-related components.

1. Introduction

Ensuring the safe operation of software based systems is becoming an increasingly important task for software developers. Typically, this task falls to those who develop using Ada, as the language is a natural choice for safety related systems. The Ada language and its use in safety related systems has been the subject of much study. An obvious example of this is the work which culminated in the SPARK subset[1]. The techniques used to determine whether a system, or parts of a system, are safety related have received less attention. This paper describes the practical application of safety analysis techniques on an Ada based project.

2. System overview

The Systeme pour Coordination, Traitement et Visualisation (SCTV) project is one of the systems currently being developed as part of the Year 2000 Investment Programme for the Guyana Space Centre (CSG), PICSG 2000. The main aim of

the programme is to provide the ground infrastructure for the Ariane 5 launcher programme for the next fifteen to twenty years.

The SCTV system is distributed across several sites and over several computers within each site. In total, SCTV is made up of fourteen DEC Alpha workstations, three DEC VaxStations and three DEC rmVAX real-time computers. The language of choice for all software is Ada, although some of the software is written in C and FORTRAN. Most of the FORTRAN software is existing code "industrialised" as part of the project. The system has been under development since November 1993 by an international consortium consisting of Logica and Sema teams in France, England, Italy and Spain. The French National Space Agency CNES, based in Toulouse, are managing the procurement of the system and the final users will be CSG in Kourou.

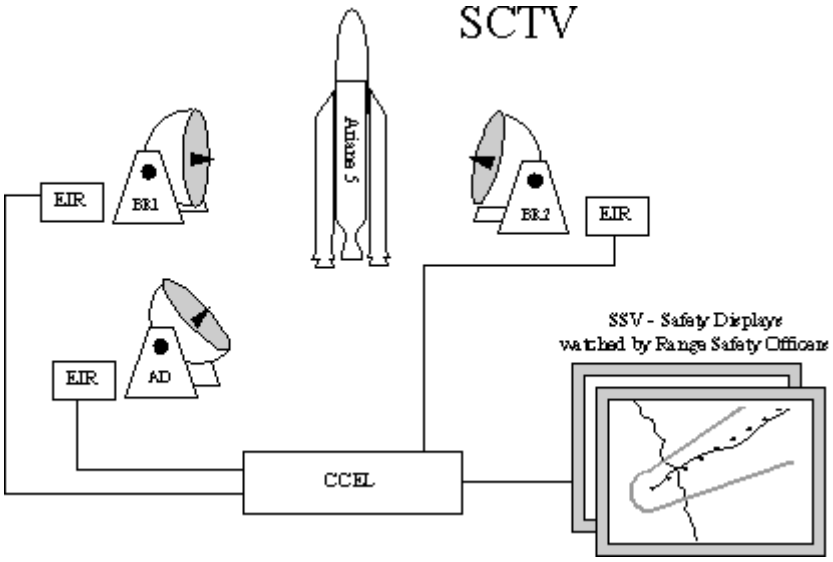

Figure 1. Overview of the SCTV system

The main purpose of SCTV is to monitor the trajectory of Ariane launchers throughout their flight. Telemetry systems and radars positioned around the launch site feed SCTV with data describing the launcher's position. This information is used to determine whether the launcher is behaving normally i.e. whether it is following a nominal and safe trajectory. If the launcher is shown to be deviating in a dangerous way, then the range officers, who monitor the information provided by SCTV, have the power to destroy the launcher in the air before it can cause loss of life or damage to people and equipment on the ground.

3. Safety levels

CNES have defined a series of high level fault trees for PICSG 2000. These fault trees show how failures within SCTV (called mission critical events) could lead to one or more "feared events" which sit at the top of the fault trees. Feared events include loss of human life, destruction of ground systems and incorrect destruction of a launcher. Comparison of these fault trees and the ESA safety standard for space systems and equipment[2] show that a failure of SCTV could not on its own lead to a catastrophic event. Catastrophic in this sense may be related to integrity level 4 of the IEC standard[3] (but note CNES have defined their own failure probability requirements). This comparison sets the overall level for safety analysis work.

It is necessary to employ safety analysis techniques to ensure that the delivered system offers as little risk of failure in operation as possible. It must be accepted that perfect safety is not possible. Given that fact, safety analysis has two important goals to achieve. Firstly it must ensure that the overall safety of the system as designed meets the safety thresholds mandated for the project (the ALARP principle[4]). Secondly, by identifying the "safety drivers" of the project it allows a greater proportion of the available project resources to be focused on those elements. This holds out the prospect of improved levels of safety being achieved within the finite resources available to the project. Safety analysis requires conscientious pragmatism.

4. Safety requirements

Safety requirements are expressed as a set of high level safety related functions; functions that must be performed in order that the range safety officers can fulfil their role during launches. The safety related functions on SCTV are a small subset of the total required functionality.

In addition, a set of high level "mission critical events" (MCE) are defined for SCTV. These MCEs are the shown on the high level fault trees mentioned in section 3. The design of SCTV should ensure that the risk of one of these mission critical events actually happening is acceptably low.

Functions can fail in different ways, and combinations of failures can lead to MCEs. The following sections describe the techniques employed on SCTV to determine how these failures could occur and to mitigate both the chances of occurrence and their consequences.

5. Analysis through the lifecycle

SCTV follows a conventional project lifecycle. Yourdon and SADT were used to perform the functional analysis, describing system functions derived from user requirements. These functions were translated into operations within objects, captured using HOOD. Software components were produced from the objects, and the software was subjected to verification and validation.

Safety analysis develops through the lifecycle as the project develops. At the beginning of the lifecycle, safety analysis concentrates on requirements, because that is all that is available. As the project progresses, safety analysis examines and influences the design and then goes on to determine the level of development and testing rigour appropriate for individual software (and hardware) components. SCTV uses two primary safety analysis techniques :

1. the identification, allocation and tracking of safety related functions,
2. the examination of the effects of failure modes.

These two techniques cross-reference and support each other in a manner which is described below.

6. Identifying and tracking safety related requirements

As described in section 4, safety related functions are identified. A process that continues throughout the lifecycle is the identification and tracking of the functions, then objects and finally software and hardware components that realise these safety related functions. This identification and tracking process ensures that components which realise safety related functions are handled with an appropriate degree of rigour throughout the lifecycle.

7. Examining the effects of failure modes

A parallel analysis activity determines the effect of failure modes of functions, then objects and finally software and hardware components. The Failure Modes, Effects and Criticality Analysis (FMECA) is used to determine failure modes (Fig 2). Fault Trees (Fig 3) combine failure modes to reveal possible outcomes of those failures, and how combinations of failures can lead to MCEs. This analysis feeds back to influence the project at the next stage of the lifecycle. For example, as the design becomes more detailed it is possible to identify and modify parts of the design to remove single points of failure and other areas of potential weakness. In this way risks are assessed and action taken to mitigate those risks. This is an iterative process.

Failure Modes, Effects and Criticality Analysis

Subsystem _____ Prepared by: _____ Date: _____

Item	Failure Modes	Cause of Failure	Possible Effects	Severity	Mitigation Actions
[Name of software component]	[In what ways could this software fail?]	[What would cause the failure?]	[If the failure occurred, what would happen?]	[Just how dangerous are the consequences of failure?]	[What can be done to reduce the chances of the failure occurring and/or its effects?]

Figure 2. An example of a FMECA table.

8. Combining the techniques

The identification and tracking of safety related functions results in some of the Yourdon functions being marked as safety related. The objects that are allocated these functions are then also marked as safety related. The software produced from these objects is considered to be safety related software. Not all software is marked as safety related. Safety categories are allocated to software components. This is discussed later. At each stage a check is made to ensure that the safety related functions are being properly identified and tracked at the transitions between different requirements and design representations (e.g. Yourdon to HOOD).

At the same time, the "effects of failure modes" analysis is performed. The results of this analysis should confirm that functions, objects and software components, marked as safety related, can have a significant impact on the successful operation of the system. If it becomes obvious that the failure of an object, which is marked as safety related, would have no impact on the system whatsoever, then the object is re-examined to see if it can be marked as not safety related. Conversely, an object that has been ignored in safety terms but which, through a failure mode, could cause the whole system to fail, would have to be marked as safety related.

In this way the two techniques cross check each other. The "effects of failure modes" analysis should also show where redesign would help mitigate risk of failure. The way and extent to which the risk is mitigated determines the cost to the project. A compromise has to be reached between acceptable risk and affordable cost.

It is vital to start this kind of analysis work as early as possible. The later safety related problems necessitate design changes, the more expensive to the project the change becomes.

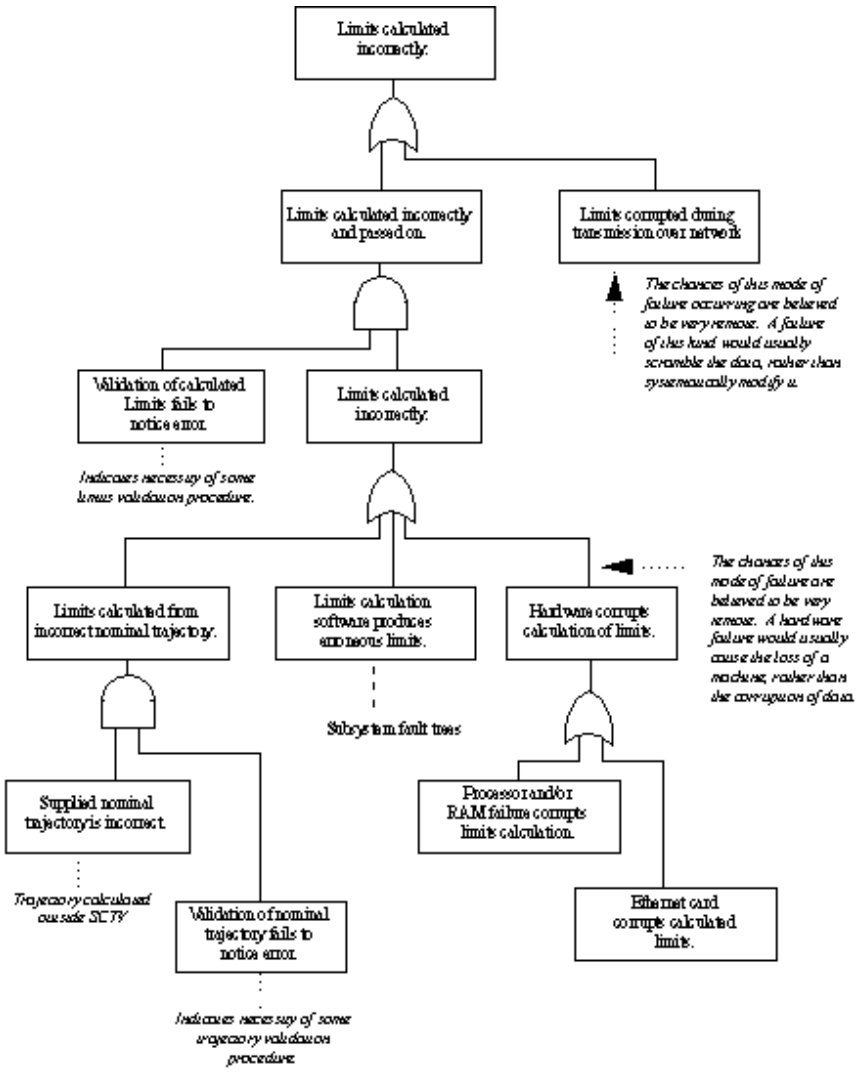

Figure 3. An example of a Fault Tree.

9. Safety categories

The techniques described identify both those components which pose the greatest risk to the system should they fail, and those whose failure would represent a loss in functionality, but which might not lead to a serious situation. This means that components have different levels of significance with regards to safety. It follows that it is possible to categorise components depending on their different levels of safety significance.

On SCTV, we have defined three levels of safety related significance, category 1 being the least safety significant, category 3 being the most significant. Software components that are identified as being more significant in safety terms, are being developed and tested in a more rigorous way; in a way that increases confidence in their ability to function correctly. Rigour is enhanced through the use of :

1. greater controls during detailed design and coding - specific rules for safety software contained within the Ada coding standard,
2. the use of automated standards checking (AdaTest),
3. formal inspections (Fagan Inspection Technique),
4. enhanced levels of testing (for example 100% condition coverage as opposed to simply statement coverage).

The result is that software realising safety related functions should be less susceptible to failures than software in other parts of the system.

10. Conclusion

The application of these techniques results in the creation of a trace from high level safety functions to the safety categories assigned to individual software components. The trace continues through inspection and testing records demonstrating that the appropriate level of rigour has been applied to each component.

The techniques discussed represent a two-part approach to the safety related software developed as part of the SCTV project. Firstly, a thorough analysis of risks allows weaknesses to be designed out of the system. Secondly, development and testing rigour is determined by the safety category allocated to each software component. The visibility and ownership of this analysis by the project team enables thorough checking of the developed system against project standards throughout the project lifecycle. It also ensures the audit trail of implementation standards is clearly marked. The approach taken thus maximises project management and client confidence that the system will meet its safety requirements.

References

[1] B A Carré and T J Jenning, 'SPARK - The Ada Kernel'
[2] 'System safety requirements for ESA space systems and associated equipment', ESA PSS-01-40 Issue 2 September 1988.
[3] IEC 65A WG9 'Draft International Standard for Software for Computers in the Application of Industrial Safety-related Systems'.
[4] 'Guidelines on Risk Issues', The Engineering Council and Lloyds Register.

This paper was published in the Abstracts journal of the Ada in Europe 1995 conference, held 2nd to 6th October 1995 at the Marriott Hotel, Frankfurt, Germany.