The Need for a Software Safety Assessment In Weapons and Munitions Systems

(OR: How to Byte Off More Than You Can Test)

Presented by:

Group Captain Bill Mayne,
Royal Australian Air Force
President, Australian Ordnance Council

Prepared by: ***Mr Arthur Ringer***
Technical Staff Officer - Explosives
Australian Ordnance Council

Synopsis

Weapon systems and their components often contain safety or arming features which are activated under software control; the control may be exercised through programmable timers, an electrically erasable programmable read only memory (EEPROM) or a microprocessor. However, software is just one part of a system and the assessment of software safety only becomes relevant in the system context: if the system moves into a hazardous state due to a software problem, then the software which caused that state is unsafe. This paper describes an approach to the assessment of the extent to which the software may be trusted. The approach includes traditional tools such as Hazard Analysis and Fault Tree Analysis together with Validation and Verification and other techniques for assessing code and indicates developments in Australia addressing these and related issues.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 1996		**00-00-1996 to 00-00-1996**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Need for a Software Safety Assessment In Weapons and Munitions Systems (OR: How to Byte Off More Than You Can Test)	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Australian Ordnance Council,Campbell Park Offices,Canberra ACT 2600 Australia,**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM000767. Proceedings of the Twenty-Seventh DoD Explosives Safety Seminar Held in Las Vegas, NV on 22-26 August 1996.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**6**	

Introduction

Professor Nancy Leveson of the University of Washington, widely acknowledged as a leading authority on software engineering, wrote the following in Risks Forum, an Internet publication on computer safety:

"Software engineers are causing the same accidents other engineers learnt to avoid years ago. I feel that unless (they) learn those basic safety concepts that have been accumulated over years by engineers, we are going to repeat the accidents of the past and kill thousands of people unnecessarily."

Some Notable Accidents

A320 Airbus ran off runway and crashed into earthen wall at Warsaw, 2 killed, 14 Sep 93. Aircraft landed long at high speed on wet surface, and commenced aquaplaning with only one wheel down. Braking logic controller prevented application of reverse thrust until weight on BOTH wheels

28 soldiers killed in a barracks by a Scud in Saudi Arabia during 1991 Gulf War. **Patriot missile** software blind to specific combination of altitude, trajectory and speed of target.
(IEEE Software, 1991)

Therac 25 X-ray machine. Between June 1985 and January 1987, six patients received severe overdoses of radiation while being treated on the Therac 25, medical linear accelerator. Two people died as a result of radiation exposure, two would have died from their radiation dose had they not died from the cancer for which they were being treated and two suffered various degrees of scarring and permanent disability. The problem may be summarised as a race condition created by the speed of entry of data by the operator; this is NOT "Operator Error" but the activation of a fault in the control software.

The London Ambulance Service disaster, a failure of a computer aided dispatch system.

Ariane 5 rocket, 6 Jun 96. Veered off course and destructed. Software testing inadequate.

Swedish JAS-39 Gripen crashed while landing in gusty conditions on 2 Feb 89 and then again on 8 Aug 93. In the first accident, the software requirements had been incorrectly specified with the result that stability margins were exceeded; in the second accident, a pilot induced oscillation was set up, forcing an unstable state (the designers had assessed the risk of this happening as very low).

Three USAF **F-117 stealth fighters** crashed under similar circumstances, diving straight into the ground. A failure of the autopilot system in what is an intrinsically unstable aircraft.
(Janes Defence Weekly, 4 Nov 95)

Safety Critical Software

Weapon systems and their components often contain safety or arming features which are activated under software control; the control may be exercised through programmable timers, an electrically erasable programmable read only memory (EEPROM) or a microprocessor. An assessment of the safety and suitability for service of such a system must therefore include an assessment of the extent to which the software may be trusted.

The Need for a Software Assessment

Software is just one part of a system and the assessment of software safety only becomes relevant in the system context: if the system is in a hazardous state, then the software which caused that state is unsafe.

Reliability. Software reliability is specified and defined differently from hardware reliability. Hardware reliability is measurable in terms of MTBF and the probability of random failures. Software reliability is much more complex. Software faults may be caused by undetected hardware errors such as transient faults causing corruption of data, security violations, human error or interface problems with other systems such as timing errors or bugs in the supporting system. The removal of all faults and the provision of a perfect execution environment is virtually impossible to attain; the trend, therefore, is to produce fault-tolerant software and accept that run-time failures are likely to occur. In situations where the cost of a failure is unacceptably high, special techniques must be used to make the system safe.

Assessment of Munition Related Safety Critical Computing Systems

The AOC approach to the assessment of safety critical software embodies the following processes:

Hazard Analysis
Validation and Verification
Software Risk Management
Personnel Management

Software Hazard Severity Categories

The Software Hazard Severity Category indicates the consequence of a hazard being realised. The four categories are defined in terms of their most severe possible consequences. They are:

Catastrophic:	multiple loss of life.
Fatal:	loss of life.
Severe:	severe injury.
Minor:	minor injury

It is necessary to ensure that safety critical code is free of errors; this is established through a combination of mathematical proof and experimentation.

Verification. Safety critical software must be verified as being correct with respect to its safety critical function; it is sufficient to verify that the software satisfies the Software Safety Criteria. Verification is achieved by mathematical proof, which may be either formal or rigorous. Proof alone is not sufficient to demonstrate safety.

Validation. Safety critical software must be validated as being error-free with respect to experiments chosen to reflect this safety critical function; the process of checking that the Software Safety Criteria capture the safety requirements is a form of validation. Validation is achieved by experimentation, which can include reviews, simulations, tests and trials. Experimentation alone is not sufficient to prove safety.

Independent Verification & Validation. Proofs and experiments concerning safety critical software must be checked and ratified by parties independent of those who performed them originally.

Software Integrity Level. There are four SILs, namely S1, S2, S3 and S4. S4 is the most demanding of the four, requiring mathematically formal specification and development and IV&V. S1 is the least demanding, requiring merely sound software engineering practices. Two orthogonal parameters determine the integrity level viz the hazard severity and the system vulnerability to software failure.
Table 1 gives a description of each level and its relationship to the Safety Critical Computing System Function (SCCSF) and Table 2 indicates the SIL appropriate to each risk level.

Table 1. Description of Software Integrity Levels

Level	Description of SIL
S4	Utmost confidence is required that the SCCSF maintains the system's safety requirement. This entails maximum use of software engineering and safety analysis techniques.
S3	A very high level of confidence is required that the SCCSF maintains the system's safety requirement. This entails significant use of software engineering and safety analysis techniques.
S2	A high level of confidence is required that the SCCSF maintains the system's safety requirement. This entails moderate use of software engineering and safety analysis techniques.
S1	This represents the situation where sound software engineering practices will ensure that the SCCSF maintains the system's safety requirement.

Table 2. Software Hazard Criticality Matrix

SOFTWARE CONTROL	*SOFTWARE HAZARD SEVERITY CATEGORIES*			
CATEGORIES	CATASTROPHIC	FATAL	SEVERE	MINOR
Software decision and action	S4	S4	S3	S2
Software decision and action, operator decision and action	S4	S4	S2	S1
Software decision, operator decision and action	S4	S3	S2	S1
Operator decision, software action	S4	S3	S1	S1
	Multiple Loss of Life	Loss of Life	Severe Injury	Minor Injury

How To Byte Off More Than You Can Test

Limitations To Be Aware Of. There are some theoretical and practical limitations to software assessment that make the achievement of a safety assessment impossible to obtain for many products.

Testing All Paths. It is generally impractical to attempt to test the program with combinations of data paths representing all possible execution paths through the product; the number of combinations to be tested quickly exceeds the capability of the test bed to perform that number of tests in a reasonable time.

Testing All Data. Operating conditions often differ from test conditions. It is generally impractical to attempt to test the program with combinations of data representing all possible inputs; the number of combinations to be tested quickly exceeds the capability of the test bed to perform that number of tests in a reasonable time frame. A tailored approach to testing is therefore the only feasible option and a testing oracle is then needed to determine the correctness of the output for a given test input.

Formal Verification or Proof of Correctness. Proof of correctness is a collection of techniques that apply the formality and rigour of mathematics to the task of proving the consistency between an algorithmic solution and a rigorous, complete specification of the intent of the solution. The limitation to this approach is the dependence upon a correct formal specification that reflects the user's needs (needs such as performance, reliability, quality or safety). The lack of suitable powerful tools may also make the proof technique impractical.

Quotable Quotes on the *Benefits(?)* of Relying on Testing Software

Unlike the fairy tale Rumplestiltskin, do not think that by having named the devil that you have destroyed him. Positive verification of his demise is required.

System Safety Handbook for the Acquisition Manager, USAF

Software testing can only prove the presence of errors, NOT their absence.

Edward Dykstra

The (US FAA) administrator was interviewed for a documentary film on the Paris DC-10 accident. He was asked how he could still consider the infamous baggage door safe, given the door failure in the Paris accident and the precursor accident in Canada at Windsor, Ontario. The administrator replied, and not facetiously either:

'Of course its safe, we certified it.'

C.O. Miller
A Comparison of Military and Civilian
Approaches to Aviation Safety