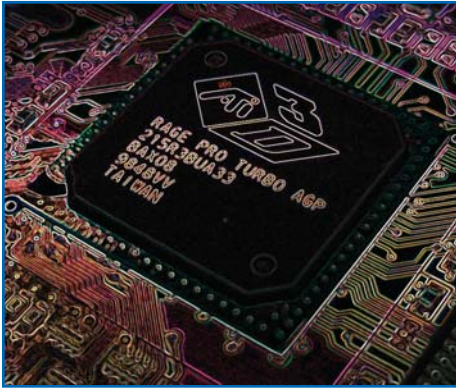


Affordable Software Safety - More for Less



Software, Software, Everywhere

Software is increasingly being used within systems which perform functions important to safety, both for new facilities and in the replacement of existing hardwired systems which are coming to the end of their life. Such systems include fire control and reactor control panels.

This transition from hardwired to software based equipment, fuelled by cheaper processing power, is often accompanied by improved functionality or flexibility. The knock-on effect is that hardwired components typically required for safety system functionality, such as simple closed loop control, alarm and indication functions, are uneconomic to produce.

Software Safety Headache

From the safety perspective, hardwired components offer the advantage that they have a limited number of predictable failure modes and there are widely accepted techniques for estimating their likelihood of failure. This is good news for those tasked with producing the safety justification.

Box 1 IEC 61508

IEC 61508 deals with functional safety and advocates the application of a safety process during system development, the rigour of which is directly related to the required Safety Integrity Level (SIL) of the system. Four SILs are recognised, designated SIL 1 (low integrity) to SIL 4 (high integrity).

In the case of an equivalent software based safety system, it is usually more difficult to produce an equally robust safety justification following the requirements of IEC 61508 (Refer to Box 1).

This is particularly true if software is partially or wholly proprietary, when the source code and specification are normally unavailable. Very often, the implied time, cost and project risk of producing a safety case may force designers to adopt a costly bespoke hardware solution, even though this may provide inferior functionality. More worryingly, in some instances software may be implemented without proper safety justification.

Testing, Testing

When software safety cases are produced, they typically comprise a number of legs (see Box 2), one of which is verification and validation (V&V) testing. This may take the form of functional testing which is applied to provide product safety assurance evidence, but is often poorly targeted. Furthermore, this V&V testing may ignore any additional functionality provided as a feature of the proprietary software, which, while unused in the safety system, could result in its hazardous failure.

A New Approach

A new approach, developed by Risktec, makes software safety justification a more practical and affordable proposition. The key is to adapt classical safety techniques such as HAZOP and Functional Failure Analysis, to derive the safety requirements for software in its specific safety application.

As this approach is applied at the functional level, it does not necessarily rely upon access to the source code, and also takes into account any effects on system functionality when the software is integrated with hardware. The result is a comprehensive and auditable set of safety requirements.

Box 2 Software Safety Case

The certification of a software based safety system relies on a combination of the following types of evidence:

Process Evidence - Evidence produced during the software development process, which demonstrates compliance with the appropriate safety process (e.g. software specification, design documentation).

Operational Experience - Previous operational experience with the system or with one or more of the software components, where shown to be directly relevant in terms of the software version and operational envelope.

Independent Certification Safety - Certification performed by an independent third party.

Product Evidence - This may include formal proofs or functional V&V testing, in addition to that undertaken during system development.

When V&V testing is focused by these safety requirements, the outcome is a compelling demonstration that the requisite level of safety can be achieved. This, coupled with the ever increasing speeds of testing, means that a comprehensive evaluation can be undertaken in realistic timescales.

Affordable Software Safety

As the use of proprietary software continues to grow and the option to use hardwired systems diminishes, the desire for affordable software safety justification begins to strengthen. At the same time the amount of available process evidence is substantially lessening. Offsetting a lack of evidence of sound safety process during development with a much greater emphasis on safety-led testing offers a practical means of assuring product safety.

For further details, contact Kevin Charnock (Warrington).