Definition of Safety-critical system

A safety-critical failure effect is one that could result in loss of life, serious injury, illness or serious environmental damage.

A system is safety-critical if a malfunction could result in a safety-critical failure effect. This includes active protection systems.

Software Intensive System Engineering
Getting the requirements right (from a safety perspective) is about identifying the appropriate hazard and putting a system in place to detect and mitigate that hazard. Frequently these requirements have to cover 'failures' (i.e. those operations for which the system was not intended to be operated) emanating from humans (the operator), designers (failures manifest from latent errors, or system deficiency) or underlying component failure. Safety Systems often have additional expectations with regards 'availability' (i.e. continue to work in presence of failure) and 'integrity' (i.e. always give the same result in operation, and possibly in failure). This Systems Engineering approach has to cover development activities, but also cover operation and maintenance tasks.

Do you have a process for specifying (and acquiring) product and test rigs?

You need to be an intelligent customer, understanding the safety implications of a defined system and ensuring that the Tier 1 or 'turnkey' supplier understands your expectations in ensuring that the system composition and integration achieves the safety goals from design through acceptance and operation and in through-life maintenance.

One procurement model does not fit all. In some cases there is only one bidder for a complex test/experiment rig. In this case, the do/buy, and make/buy could well show that it best to bring it in-house and take more control of the risks. E.g. A rig has only one bidder from outside Europe who does not understand EU regulations. They are a small company and cannot take on full liability for the cost of failure. Is it reasonable to take special measures to assure the investment?

"Reasonably foreseeable mis-use" is not often taken into consideration in risk assessments. One area related to "software" that should consider "reasonably foreseeable mis-use" is operator interfaces which are often GUI or combinations of GUI and traditional controls/lamps. E.g. This is evident from examining the system design of Operator Controls on various rigs/plant automation.

Change control and maintaining design intent post development, installation and commissioning.
This is an area that is often forgotten after handover of the facilities and the technical strength with knowledge of the design intent evaporates away from the facility to other projects. You need to have a mature understanding of the role of the "Duty Holder" in UK safety management arrangements.

The final point in this area is maintaining the effectiveness of control measures to ensure the facility remains ALARP. Control, when used in safety, requires sensors and actuators, both need to be maintained, checked, calibrated, etc, to maintain effectiveness. Layers of protection can easily be eroded and this has an impact of both safety and availability.

Prompting Questions for soliciting evidence in support of appropriate HS&E approaches

Systems Engineering for Safety in Programmable Systems (Supplier Questions – Show me....?)

- *Your approach to Systems Engineering*
- *How you specify the 'duty' for the system (safety requirements)*
- *How you have assessed the Safety or business continuity (or commercial impact)*
- *The Safety Case, including arguments, Fault Trees, Common Cause Analysis, etc*
- *The Requirements, Design and Failure Analysis (FMEA, FMECA, FHA, Fault Tree)*
- *The Supplier assessment and oversight (fit for 'duty') - for each supplier/sub-supplier*
- *The mechanisms of Failure Detection and robustness (Signal Integrity and Validation, Redundancy, Diversity)*
- *The intended Failure response and the Validation of failure effects*
- *Industrial 'confidence' in solutions (Electronics, Programmable Hardware, Software) – Simplicity*
- *Human Factors*
- *Management of system modification (pre- and post- delivery)*
    - *How System changes including (particularly, but not uniquely) Software changes cause the Safety Case to be re-assessed*

Electronic Engineering (Supplier Questions – Show me....?)

- *Your assessment of the mechanisms of failure (both Permanent and Transient)*
- *The mechanisms for detection of these failures*
- *How the system is designed to respond to a detected failure*
- *How the design limits propagation of failure*
- *Your assessment of Environmental triggers for the above failure e.g. Stress (Electrical, Thermal, Shock and Vibration, Ageing), Electrical Noise (EMC) and mechanisms to mitigate or alleviate them.*

Software Engineering (Supplier Questions – Show me....?)

- *Your assessment of the Safety Integrity Level (SIL) required of the system (the need) (N.B. This may be different for various components of the system)*
- *How you have assessed the system/software process needs for each part having a different SIL above*
- *How the process you have defined/followed relates to the needs and the standards (e.g. assessment and selections based on IEC 61508)*
- *How those processes and relevance to need are documented in a Software Development Plan*
- *Your Software Quality Plan and Audit evidence of compliance*
- *Your approach (system, recording and tracking) of*
    - *change control (tracking of change request through to validated solution to change) and*
    - *configuration management (component versions and their incorporation into baseline builds) (N.B. Particular "watchpoints" may be components that use the same part number even though the software build is not the same – this is not acceptable)*
- *How that approach is documented in your Software Configuration Management Plan*
- *How your approach (above) applies to*
    - *new developments (including new requirements/ user requested changes etc)*
    - *error correction for both:*
        - *Errors found on this system/installation ('system of interest')*
        - *Errors found on 'other systems' (similar installations sharing components / ancestry)*
            - *How the impact to the 'system of interest' from changes originating from 'other systems' was assessed*

Software Engineering is essentially a human endeavour and suffers from the failings of humans – we make errors. Development processes are about minimizing these in origination, and detecting them (near to where they are originated, most cost effectively).

Software Development approaches for safety

There are three approaches commonly used in the engineering of software-based safety critical systems to ameliorate (but not entirely mitigate) errors in the product. All of these approaches improve the software quality in safety-critical systems by testing or eliminating

manual steps in the development process, because people make mistakes, and these mistakes are the most common cause of potential life-threatening errors.

1. *Process engineering (e.g. DO178B for Aero, 61508 or its derivatives for other process industries) and management.*
2. *Using the appropriate tools and environment for the system. This allows the system developer to effectively test the system by emulation and observe its effectiveness.*
3. *Legal and Regulatory requirements, (e.g. Aero regulator, Nuclear regulator etc). By setting a standard for which a system is required to be developed under, it forces the designers to stick to the requirements.*

BS EN/ IEC 61508 (Functional safety of electrical/electronic/programmable electronic safety-related systems) and its derivates (e.g. 61511, 61513)

The standard approach is to carefully code, inspect, document, test, verify and analyze the system. This relies on yet more human endeavour... so is a) imperfect, b) dependent on competence and c) subject to programme management pressure.

Another approach is to certify a "production system", e.g. a pictures to code compiler, and then generate the system's code from specifications. This is commonly called a 'qualified toolset' approach – and is rare because of the expense of creating the qualification evidence... which usually only pertains to a single context.

Another approach uses "formal methods" to generate mathematically supportable "proofs" that the code meets requirements. This translation to a formal mathematical base for all but simple logic is VERY expensive, usually used by those demanding the highest levels of SIL (e.g. SIL4 in 61508) on very small pieces of software.

Key System and Software Development Indicators

- *Management, Metrics, Measures and closed-loop Feedback*
  - *Mine the Configuration Management system for trend evidence*
  - *Understand effort levels, balance of programme, sensitivity of components and architectural significance*
- *Estimation, Size, Programme, Complexity*
- *Re-use, Provenance*
  - *Understand the value (and cost) of re-use, strategic, planned re-use like Product Line techniques etc*
  - *How much and what has changed (hidden and obvious)*
- *Skills and Competence*
  - *Competency frameworks (Industry standards), assessment of engineers and deployment against skill type*
    - *Look for 'certified' engineers or assessments to international standard frameworks*
  - *Active training solutions for skill and competence development*
  - *Professional development support*
- *Capability Maturity*
  - *CMMI type assessments – internally or externally assessed*
- *Safety and Quality organisations and their ethos*
  - *Part of product development – but independent authority*

Attributes of typical Safety Critical Control Solutions

- *Safety Case and their composition from Safety Arguments*
- *Composite solutions and sub-tier suppliers*
- *Inspection in support of Safety (not just finished article, but development assurance)*
- *Acceptance tests (including safety criteria)*

Safety Critical Software Systems in Operation

- *Human Factors issues with Software (e.g. Graphical User Interfaces)*
  - *Graphical User Interfaces, 'context sensitive' displays, live or frozen.*
- *Human factors with Automation in Safety Critical situations*
  - *No skilled operator with regular practice, yet controls revert to manual when automation unable (usually when faced with significant failure).*

Systems Engineering for Security in Programmable Systems

- *Malicious intent and Security threats*
- *Opportunities for security breach throughout the development lifecycle*
- *Mechanisms of potential vulnerability*

References/Suggested reading material:

Nancy Leveson "Safeware: System Safety and Computers" http://sunnyday.mit.edu/book.html