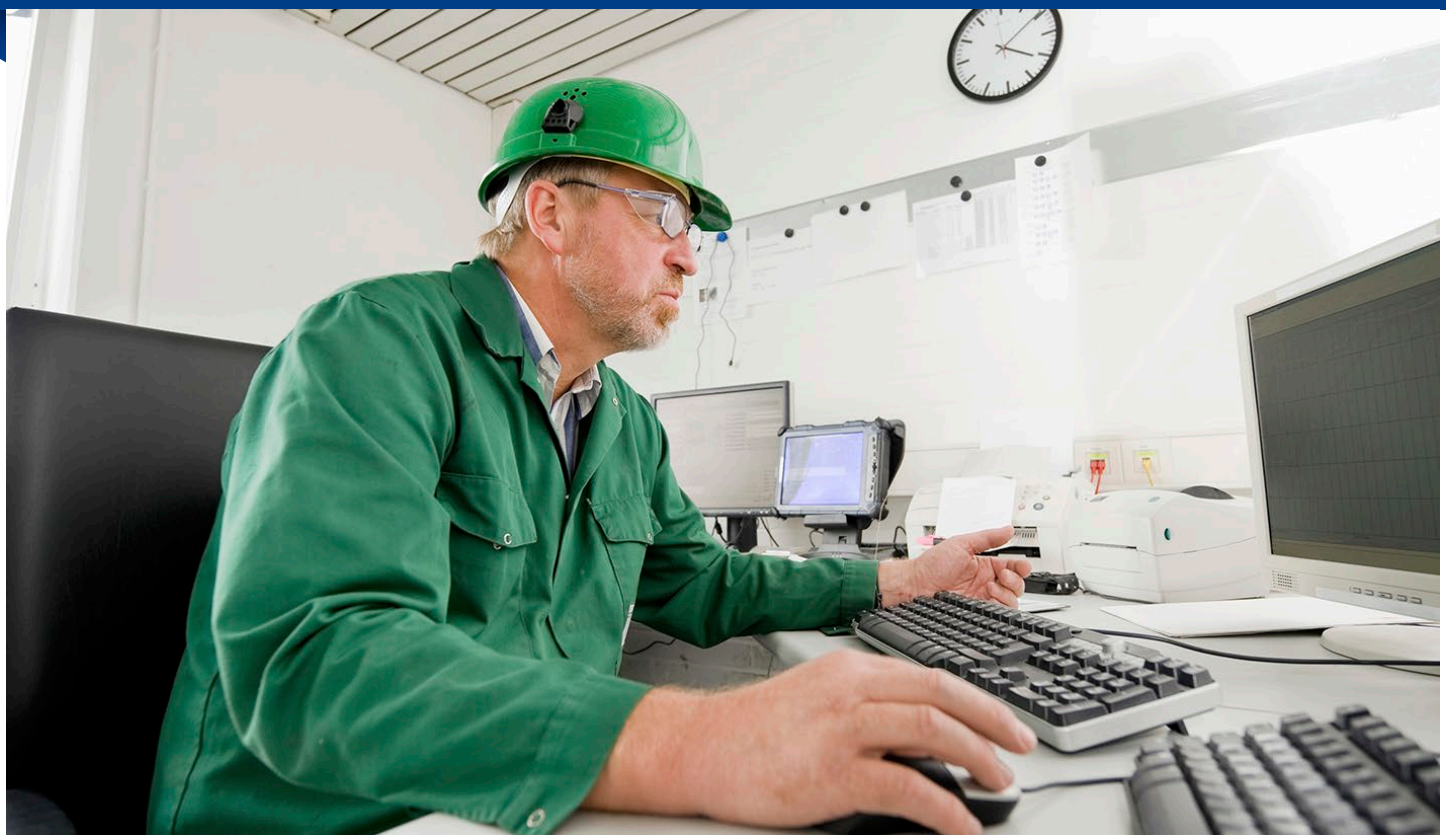
Expertise in Turbomachinery Controls

WHITEPAPER

Effectively execute legacy safety systems migration – Part 1



Publish Date: 11/1/2016
Author: Dan Mulholland , CCC

In this two-part article, we will investigate the challenges of migrating legacy Safety Systems and review effective means of repeatedly and effectively executing compliant migrations. In Part 1 we will focus on the requirements within the applicable standards.

Introduction

In this two-part article, we will investigate the challenges of migrating legacy Safety Systems and review effective means of repeatedly and effectively executing compliant migrations.

In Part 1 we will focus on the requirements within the applicable standards.

In Part 2 we will detail the means by which effective and compliant migrations can be carried out.

Historical Perspective

In recent years the requirements for compliance with Functional Safety best practices has become a key pre-requisite for the realisation of safety solutions. This arises from the predominant causes of incidents being systematic errors which cannot be corrected without modification of the system i.e., repair or replacement of hardware does not resolve the underlying problem. A study performed by the UK Health & Safety Executive in 2004 showed that the majority (44%) of systematic errors were latent from the original requirements definition with a further 15% introduced during the engineering phase. Others will have been introduced during maintenance and modification activities throughout the lifecycle. It must therefore be accepted that there is a high risk of latent systematic errors in legacy SIS application programmes.

In some cases, programmable control and safety systems may have been operating for many years and the owner/operator would like to take some "Proven in Use" credit for the application, however hardware obsolescence demands a transfer of the application software to a new platform. Control system migrations have been performed many times and it is a general misconception that safety systems can be migrated, translated, transferred or upgraded with the same lower degree of rigour.

Legacy SIS Platforms

Programmable Safety Instrumented Systems (SIS) have been around since the 1980's, however it was soon recognised that there were issues in the integrity of these systems which led to a number of guidelines and standards appearing.

Safety Standards

In 2010 the 2nd edition of IEC 61508 was released and one of the many areas addressed for improvement was application software and the validity of legacy programmes transferred to new platforms. This paper describes how SIS application migration can be performed whilst meeting the onerous requirements of the latest edition of the IEC 61508 standard.

Issues for SIS Migration

Future Proofing

The costs of deferred production drive the requirement for accurate interpretation of the legacy programme such that the migration from the legacy SIS to the new SIS is as short as possible with a high degree of confidence that it will work consistently.

In order to minimise operational costs, plant operators strive for standardisation to reduce costs for spares, training and maintenance. It is desirable to be able to replace legacy hardware with current "future proofed" hardware and to be able to easily and quickly migrate application programmes from the legacy SIS to the new hardware platform. The process should also bring the documentation of the SIS up to current required quality and compliance standards.

Prior Use?

If the legacy SIS application programme has been providing stable operation then the many years of documented service could be used to build a "Prior Use" case. If the migration process can demonstrate complete replication of the functionality then the new programme may be able to take some credit for the "Prior Use" in its Validation & Verification case. Unfortunately this is highly unlikely due to the rigorous requirements for substantiating such a claim in compliance with current standards. In many cases the legacy programme does not have a lifetime of supporting documentation and may incorporate coding practices no longer seen as good practice.

Often the only "documentation" for the legacy system is the system configuration database from the CPU's which may have been modified many times since the original "As-Built" documentation, therefore it is highly important to develop a clear and testable interpretation of the legacy code functionality. The terminology for such legacy software is SOUP (Software Of Uncertain Pedigree) and all issues of pedigree need to be cleared in the migrated version.

In summary, the challenges for SIS migrations are:

- Criteria For Acceptance
  - Little or no reliable specification to refer back to
- Compliance
  - Legacy system not to current standards (SOUP)
- Correctness
  - Avoidance of latent Systematic errors resident in legacy system and/or added in migration process
  - Source & Destination Systems have different execution methodologies
  - Separation of Process Application program from system related legacy code
- Consistency
  - Should Minimise diversity of logic typicals
  - Structured approach which provides repeatability

Migration Process Overview

Safety Requirements Specification

The key pre-requisite to any new SIS is the Safety Requirements Specification (SRS). It is commonplace for SIS migrations to somehow forego the SRS on the basis that the legacy system is the basis for the replacement system. However, the current best practices are very clear that the avoidance of systematic errors is dependent on a rigorous SRS.

Issues To Be Considered

The first assumption to be quashed is that the legacy code is in line with the recommended programming manual of the system. It is common to find "Dead Code" in legacy systems resulting from removed safety functions being left in the logic solver.

Some legacy logic solvers have particular nuances such as the sequence of logic processing, which an engineer may have exploited to achieve a particular function more efficiently. This may not be recognisable from the code unless detailed knowledge of the legacy system operation is known.

Some legacy systems used application code to implement system architecture functions, such as redundancy and testing. Code associated with these functions is not part of the Process Safety Functionality and needs to be identified and removed in the migration process.

Legacy systems may use multiple programming formats including Function Block, Ladder Logic, Sequence Flow Charts and/or Structured Text whereas the target system will be predominantly Function Block based.

Migration Methodologies

In general there are three approaches to migrating SIS application software:-

- Manual
- Tool Assisted
- Automatic

These are described and compared in Figure – 1.

- Manual
  - Wetware dependent (Human Factors dominate)
  - Not consistent
  - Best suited to Niche/one off applications
- Automatic
  - 80/20 Rule (80% automatic, 20% manual but 80% of the time)
  - Systematic errors also mapped with limited visibility
  - Converted application structure may not be "friendly"
- Tools Assisted
  - Brings application into a functional environment
  - Combines benefits of automated processes with human perception
  - Provides a high rigour testing environment
  - Allows nuances of platforms to be addressed.

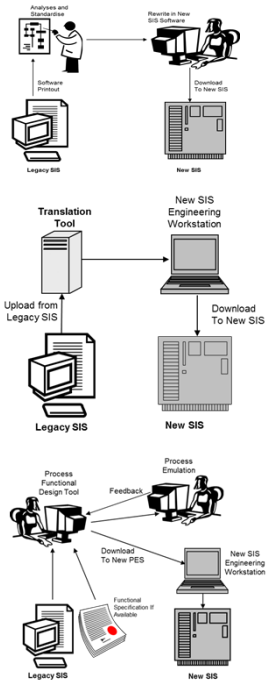


Figure 1. Migration Methodologies

MANUAL TRANSLATION

A manual translation requires human analysis of the legacy programme and human translation to a new programme in line with BSEN-61131-3 as applied in the new SIS, i.e. the translation tool is the human brain. This may be from a simple printout of the programme from the legacy SIS. This is shown schematically in figure 1.

Whilst this can produce very effective results, it is prone to the vagaries of human interpretation of the information and the particular manner in which the human writes a programme. Should the task be given to two or more humans then the likelihood is that no two resulting programmes would be identical. This interpretative issue may be acceptable, however for safety critical applications covert or systematic errors must be recognised, minimised, corrected and mitigated.

Tool Assisted Translation

An assisted translation is one which relies on human knowledge of the application but utilises tools to clarify, standardise and simplify the translation. The tool may provide a means to record or capture the perceived functionality, but in a functional environment rather than a programming environment. This makes it easier to spot and avoid systematic errors. This is shown schematically in figure 1.

The tool is derived from experience garnered from multiple manual translations which have uncovered the many challenges which have then been used to create a "rule set". The "rule set" allows the tool to automatically detect predictable features in the structure and configuration of the legacy SIS database and either apply a known corrective mapping or mitigation process or highlight a new anomaly to be addressed and added to the rule set for future use. It is therefore important that the tool and its' rule set are developed by competent persons with valid experience.

The legacy SIS database is uploaded into the tool which analyses it for known "features" and maps it into a design and test environment. In this environment the functionality can be emulated to verify dynamic consistency with the legacy SIS application. Higher test coverage can be provided in this environment by manual testing or by the application of automated test sequences and programmes developed by independent persons. The application is functionally verified before download to

the replacement system so can be performed without interfering with production.

Once the application is functionally accepted the tool may then use automatic export to a generic IEC 61131 compatible format or directly to the new SIS format incorporating any special logic functions supported. This combination of the rule set and output format are the basis for a Library of function blocks which mean that successive migrations become increasingly efficient without compromising rigour.

Automatic Translation

An automatic translation is one which in theory requires little or no human intervention. A number of automatic conversion tools have been developed primarily for control systems though it is generally recognised in these cases that the 80/20 rule applies. In other words 80% of the application is automatically mapped but 20% has to be performed manually due to complexity. Further the 20% manual effort requires 80% of the origin

An application programme database is uploaded from a legacy SIS or its' engineering station (depending on the device), it is passed through a translation tool and downloaded to the new SIS Engineering Facility. This is illustrated in figure 1. The translation tool is designed to analyse the database configuration file of the legacy system and map it to a new database for use in the target system by using a set of rules for mapping. The engineering station of the Target SIS is used for any analysis and manual adjustment/correction (80/20 rule).

Although this makes the process less dependent on system knowledge, the tool development requires very detailed knowledge of both the Legacy and Target SIS. It is extremely important that the translation tool has the capability to detect "bad" programming practices in order to prevent systematic error transfer and creation as for the Tool Assisted process. However in this case the human contribution is reduced and maximum dependence is placed on the tool. The test regime is based on the Target SIS engineering facility which reduces the test coverage. Thorough testing of the translated database means that the automatic process incurs high manual testing overheads.

Requirements of Current Standards

In 2010 the 2nd edition of IEC 61508 was published and included new requirements in relation to pre-existing software elements. In the following clauses text has been highlighted in bold where particular attention is drawn. Part 2 of the standard includes the following clause:

> 7.4.2.2 The design of the E/E/PE safety-related system (including the overall hardware and software architecture, sensors, actuators, programmable electronics, ASICs, embedded software, application software, data etc.), shall meet all of the requirements a) to e) as follows:
>
> a) the requirements for hardware safety integrity comprising;
>
> – the architectural constraints on hardware safety integrity (see 7.4.4), and
>
> – the requirements for quantifying the effect of random failures (see 7.4.5);
>
> b) the special architecture requirements for ICs with on-chip redundancy (see Annex E), where relevant, unless justification can be given that the same level of independence between different channels is achieved by applying a different set of measures;
>
> c) the requirements for systematic safety integrity (systematic capability), which can be met by achieving one of the following compliance routes:
>
> – Route 1S: compliance with the requirements for the avoidance of systematic faults (see 7.4.6 and IEC 61508-3) and the requirements for the control of systematic faults (see 7.4.7 and IEC 61508-3), or
>
> – Route 2S: compliance with the requirements for evidence that the equipment is proven in use (see 7.4.10), or
>
> **– Route 3S (pre-existing software elements only): compliance with the requirements of IEC 61508-3, 7.4.2.12;**
>
> **NOTE The "S" subscript in the above routes designates systematic safety integrity to distinguish it from Route 1H, and Route 2H for hardware safety integrity.**
>
> d) the requirements for system behaviour on detection of a fault (see 7.4.8);
>
> e) the requirements for data communication processes (see 7.4.11).

In simplest terms, if the pre-existing code is structured text and both the legacy and Target SIS will process that text in the same manner then maybe a case could be made that there is some provenance. However this is not likely to be the case as few systems were programmed exclusively in structured text and the operating systems of the two SIS will likely have no commonality anyway. This would be akin to claiming the provenance of an engine component from a vintage car makes it suitable for fitting to a modern car on the grounds that is providing the same "function". Although software does not "wear out", the context and environment in which it is applied may affect its' efficacy greatly.

It is therefore not practicable or maybe even viable to claim "Proven-in-use" for pre-existing software elements developed in different compliance regimes for totally different SIS platforms.

IEC 61508-3, clause 7.2.2.2 states the following:

> 7.2.2.2 The specification of the requirements for safety-related software shall be derived from the specified safety requirements of the E/E/PE safety-related system (see IEC 61508- 2, 7), and any requirements of safety planning (see Clause 6). This information shall be made available to the software developer.
>
> NOTE 1 This requirement does not mean that there will be no iteration between the developer of the E/E/PE system and the developer of the software (IEC 61508-2 and IEC 61508-3). As the safety-related software requirements and the software architecture become more precise, there may be an impact on the E/E/PE system hardware architecture, and for this reason close co-operation between the hardware and software developer is essential. See Figure 5.
>
> **NOTE 2 Where a software design incorporates pre-existing reusable software, that software may have been developed without taking account of the current system requirement specification. See 7.4.2.12 for the requirements on the pre-existing software to satisfy the software safety requirements specification.**

IEC 61508-3, clause 7.4.2.12 states the following:

> 7.4.2.12 Where a pre-existing software element is reused to implement all or part of a safety function, the element shall **meet both requirements** a) and b) below for systematic safety integrity:
>
> a) meet the requirements of one of the following compliance routes:
>
> – Route 1S: **compliant development**. Compliance with the requirements of this standard for the avoidance and control of systematic faults in software;
>
> – Route 2S: **proven in use. Provide evidence that the element is proven in use. See 7.4.10 of IEC 61508-2**;
>
> – Route 3S: **assessment of non-compliant development.** Compliance with 7.4.2.13.
>
> NOTE 1 Route 1S, 2S and 3S are the element compliance routes of 7.4.2.2 c) of IEC 61508-2 with particular reference to software elements. They are reproduced here for convenience only, and to minimize references back to IEC 61508-2.
>
> NOTE 2 See 3.2.8 of IEC 61508-4. The pre-existing software could be a commercially available product, or it could have been developed by some organisation for a previous product or system. Pre-existing software may or may not have been developed in accordance with the requirements of this standard.
>
> NOTE 3 Requirements on pre-existing elements apply to a run-time library or an interpreter.
>
> b) provide a safety manual (see Annex D of IEC 61508-2 and Annex D of this standard) that gives a sufficiently precise and complete description of the element to make possible an assessment of the integrity of a specific safety function that depends wholly or partly on the pre-existing software element.

What this means is that pre-existing legacy software elements can be re-used subject to a safety case being prepared to satisfy routes 1S, 2S or 3S and a Safety Manual for the legacy software must be prepared.

Route 1S must demonstrate that the original software was developed in line with the Functional Safety process of IEC 61508 current edition. This is unlikely to

be the case and even it were true would have to be supported by evidence to that effect.

3S. entails re-assessing the original application development process in accordance with the current edition of the standard.

IEC 61508-3, clause 6.2.3 states the following:

> 6.2.3 Software configuration management shall:
>
> a) apply administrative and technical controls throughout the software safety lifecycle, in order to manage software changes and thus ensure that the specified requirements for safety-related software continue to be satisfied;
>
> b) guarantee that all necessary operations have been carried out to demonstrate that the required **software systematic capability has been achieved**;
>
> c) maintain accurately and with unique identification all configuration items which are necessary to meet the safety integrity requirements of the E/E/PE safety-related system.
>
> …..
>
> NOTE 4 For further information on configuration management, see IEC 61508-7

The key points of this clause are that the Systematic Capability of the pre-existing software requires management and assessment. If the migration of the application is to involve functional modifications then these need to be done AFTER the legacy code has been validated.

Summary

The conclusion of this is that for Safety Related Software it is not practicable to take credit for pre-existing software elements developed for an operating system different to the target system. The effort involved in making the safety case is disproportionate to the effort required for making a new application. Therefore a more practical approach is to focus on the functionality of the legacy system and model it in an environment which allows the FSM procedures for a new system to be applied. This ensures maximum compliance and aids the removal of latent systematic errors.

This steers the conclusion toward tool assisted translation being the most applicable for compliance, quality, efficiency and rigour.

In Part two of this article we will investigate the challenges, procedures adopted and deliverables generated when you adopt a 'Tools Assisted' approach to Migrations.

About the Author:

Dan Mulholland is a global sales director for Trinity Integrated Systems Ltd. (TIS), and he has worked through the introduction of a new concept of physically distributed control systems in hazardous and harsh environments while working with GE Intelligent Platforms and, formerly, Measurement Technology Ltd. Mr. Mulholland has lectured both BSc and MSc curriculums in mechatronics at Leeds University and is introducing lifecycle management solutions focused on automating Phase 4 activities with TIS.