# Developing a methodology for the use of COTS operating systems with safety-related software

**Simon Connelly** **Holger Becht**
Ansaldo STS, PO Box 1168, Brisbane 4009, Queensland
{connelly.simon,becht.holger}@ansaldo-sts.com.au

## Abstract

Conventional wisdom within the System Safety community has been that Commercial-Off-The-Shelf (COTS) Operating Systems (OS) with unknown pedigree are unsuitable for deployment in safety-related systems at anything other than the lowest integrity levels. Without assurance evidence for the OS it is difficult to gain confidence in safe behaviour of the functions provided. The typical solution therefore has been to either develop wholly embedded systems or use operating systems which have been certified to a particular standard.

Regulatory and societal expectations on software assurance is continually increasing, however ever-competitive market conditions are causing budgets to remain stable, if not decreased. As modern systems become more complex artefacts, the use of certified operating systems, or development of a bespoke embedded system, present challenges to system designers which are difficult to solve within these budgetary and schedule constraints. Consequently, the use of generic COTS OS is becoming more of a necessity.

Standards poorly define how to manage OS as far as COTS is concerned, allowing for either excessively restrictive or permissive definitions of what is required. This paper proposes a methodology to isolate the safety-related service or program from failures of the COTS OS through design and detection techniques.

The model argument presented, within the framework of the SIL based standards, justifies the use of Microsoft Windows OS (or equivalent) to enable safety-related functionality up to SIL 2.

*Keywords*: COTS, software safety, windows operating system.

## 1 Scope

The scope of this paper discusses safety-related applications (i.e. up to SIL 2, SW Level C) only and is not applicable to safety-critical (i.e. vital, SIL 3/4, SW Level A/B); the reason for this is discussed further towards the end of this paper. This paper expands previous work to discuss a it's applicability to a more complex example system, and the observed difficulties this presents.



## 2 Introduction

The use of COTS software components within safety-related applications is a reality and has become increasingly more a necessity for service providers to remain competitive in a market that is driven by cost savings due to recent economic downturns. COTS software artefacts are continuing to increase in complexity, making the development of an assurance argument about systems utilising them increasingly difficult in the context of existing safety standards. Understanding the impact of COTS software failures with respect to system safety is a crucial and difficult step but key to the safety assurance of the overall system.

The term COTS, in this paper refers to software components which are readily available from commercial sources, for general application and not easily modified. Access to source code and development process is denied, or heavily restricted. Typical characteristics of COTS components are that a number of different configurations may be available, more functions than required are available, and upgrades may occur either during system development or while it is in service.

This paper is organized into the following sections.

1. A literature survey of software safety standards and how they address COTS.
2. Previous use of and assessment of COTS Operating Systems within safety-related applications.
3. Taking into consideration the literature survey and previous assessment, our approach to providing safety assurance for the use of COTS operating systems to enable a safety-related application up to SIL 2.
4. An example of this approach implemented as part of a train movement authority management system. This section expands on previous work [Connelly10] to discuss a more complex system.

## 3 Literature Survey

Most current safety standards require that a Safety Case (or similar) be developed to provide assurance evidence that the system is safe to operate and maintain. Assurance evidence for software components which perform one or more safety functions is required to demonstrate that sufficient rigour has been applied to the development process to meet the safety obligation. Generally this is executed through demonstration of compliance / achievement of a Safety Integrity Level (or something similar). The assurance evidence for software safety that is then required relies on a rigorous development process where the level of rigour and independence between teams is proportional to the SIL associated with the

functions provided by that software component. Because the provision of assurance evidence for software safety is through demonstration of compliance to a specific development process, this approach cannot be applied for COTS components as this information is generally not available. As such most safety standards provide guidance on how to manage COTS, with varying degrees of expected effort.

### 3.1 IEC61508

Requires a proven-in-use argument, and a previously developed subsystem shall only be regarded as proven in use when it has a clearly restricted functionality and when there is adequate documentary evidence, based on the previous use of a specific configuration of the subsystem (during which time all failures have been formally recorded), and which takes into account any additional analysis or testing, as required. A component or software module can be sufficiently trusted if it is already verified to the required safety integrity level, or if it fulfils the following criteria: unchanged specification; systems in different applications; at least one year of service history; - operating time according to the safety integrity level, e.g. 100,000 hours for SIL 2.

### 3.2 DO-178B

Requires a proven-in-use argument. That is if equivalent safety for the software can be demonstrated by the use of the software's product service history, some certification credit may be granted. The acceptability of this method is dependent on: Configuration management of the software; Effectiveness of problem reporting activity; Stability and maturity of the software; Relevance of product service history environment; Actual error rates and product service history; and Impact of modifications.

### 3.3 CENELEC EN50128

The use of COTS software shall be subject to the following restrictions for SIL 1 or 2; it shall be included in the software validation process.

### 3.4 Def(Aust) 5679

Allows for cross-standards acceptance up to SIL 2 only. It also requires that all the prescribed System modelling and verification activities are required for the COTS components.

### 3.5 UK DefStan 00-56

Requires a Safety Case for the COTS components, and requires “sufficient” evidence to be provided to argue for the safety of the component.

## 4 Use of COTS Operating Systems

HSE conducted a study to assess the safety and integrity of the Linux operating system [Pierce02]. The overall conclusion of the study was that Linux would be, in broad terms, suitable for use in many safety related applications with SIL 1 and SIL 2 integrity requirements, and that its certification to SIL 3 might be possible. However, it is not likely to be either suitable or certifiable for SIL 4 applications.

It was argued by Pierce that for an OS (or indeed any pre-existing software) to be suitable for use in safety related system, it must satisfy the following criteria with an argument provided in the Safety Case.

C1. The behaviour must be known with sufficient exactness, in all relevant domains of behaviour, to provide adequate confidence that hazardous behaviour of the safety related application does not arise because of a mismatch between the belief of the application designer and the true behaviour of the operating system;

C2. The behaviour must be appropriate for the characteristics of the safety related application, in all relevant domains of behaviour; and

C3. It must be sufficiently reliable to allow the safety integrity requirements of the application to be met (when taken together with other system features). In other words, the likelihood of failures must be sufficiently low.

C4. An analysis has been carried out to show that the OS is suitable for that application, and that suitable mitigation is in place for any hazards arising from OS failure.

As part of the analysis for C4, the following OS features were identified by Pierce which should be used as a minimum to assess the sufficiency or completeness of the safety requirements set on the OS:

1. Executive and scheduling – the process switching time and the employed scheduling policy of the operating system must meet all time-related application requirements;
2. Resource management (both internal to the operating system and provided to the application software) – the operating system’s own internal use of resources must be predictable and bounded;
3. Internal communication – the operating system inter-process communication mechanisms must be robust and the risk of a corrupt message affecting safety adequately low;
4. External communication – the operating system communication mechanisms used for communication with either other computers in the network or some external system must be robust and the risk of a corrupt message affecting safety adequately low;
5. Internal liveness failures –the operating system must allow the application to meet its availability requirements;
6. Partitioning – if the operating system is used to partition functions of differing SILs, functions of lower SIL should not interfere with the correct operation of higher SIL functions;
7. Real-time – timing facilities and interrupt handling features must be sufficiently accurate to meet all application response time requirements;
8. Security – only if the operating system is used in a secure application;

9. User interface – when the operating system is used to provide a user interface, the risk of the interface corrupting the user input to the application or the output data of the application must be sufficiently low;
10. Robustness – the operating system must be able to detect and respond appropriately to the failure of the application processes and external interfaces;
11. Installation – installation procedures must include measures to protect against producing a faulty installation due to user error.

The Certification Authorities Software Team (CAST) produced a paper to argue for the use of a COTS OS in safety-related application (i.e. up to SIL 2 or DO-178B level C) [CAST02]. This paper argues that the maximum integrity level that can be claimed for a COTS OS (when the source code and design information are not available) is SIL 1 (or Level D). It then goes on to argue that the COTS OS can be used with SIL 2 application if a protection and partitioning analysis is performed in conjunction with the system safety assessment. It is the opinion of the authors that the intent is equivalent to the approach suggested in the Linux paper [Pierce02].

## 5 Solution

Guidance in standards is somewhat contradictory with widely varying requirements on assurance evidence for COTS. They offer little practical guidance on the development of safety assurance evidence for COTS software components. Research conducted into COTS OS has identified the broad requirements to achieve a satisfactory safety argument; however a specific strategy is not derived, nor is there evidence of this technique being applied. We therefore revert back to the fundamentals of safety assurance and focus on:

1. Analysing failure modes of the COTS components and mitigating these to eliminate unspecified/unexpected behaviours. When the COTS component is an OS, the analysis performed must respect the criteria C1 to C4 detailed above.
2. Verifying and validating the safety of the required behaviour in the required operational context.
3. Ensuring and maintaining safety during system upgrades and change.

Our approach is to utilize the functionality of low integrity COTS components within a high integrity design by restricting the influence of the component on the rest of the system. The way to restrict the influence of COTS components is by isolating them using encapsulation mechanisms such as wrappers [O'Halloran99]. To achieve this, a hazard analysis is conducted, at a level of design commensurate with the SIL of the application, to identify how failures in the operating system can cause or contribute to hazardous failure modes of the system.

To ensure the base platform remains invariant, the approach presented here assumes that OS upgrades will not be applied without conducting further analysis, and sufficient regression testing conducted. Additionally, the OS will be minimised as far as is practicable by disabling unnecessary system services and removal \ restriction of third party applications (such as anti-virus programs). Protection against virus infections is considered to be outside the scope of this approach. This is considered acceptable as the systems under discussion are generally not utilised on an open network, or exposed to external media (e.g. USB drives) which has not been previously determined to be uninfected.

## 6 Putting it Into Practice

Hereafter we will relate the theory above to a practical SIL 2 software application within the framework of the railway safety standard EN 50128.

To comply with EN 50128 for SIL 2 one must at least demonstrate that the COTS products are included in the software validation process. System testing must be conducted in compliance with EN 50128 and requires that it be conducted on the system configured for its final application, including the hosting environment provided by the COTS OS, and all other COTS products. This testing will action only those features of the COTS products required to provide the functionality used for the product under development. Whilst the above approach satisfies the EN 50128 requirement for SIL 2 products, it is however acknowledged that conducting sufficient testing on the COTS products to ensure correct functionality in all circumstances is infeasible due to the complexity of these artefacts. As such additional precautions are required; guided by EN 50128 (as detailed above), specifically:

1. The possible failures of the OS (e.g. data corruption), which may affect safe functioning of the product, will be identified and assessed, and mitigations will be designed within the developed SIL 2 software.
2. System testing will test these mitigations as far as is practicable; and
3. The OS will be minimised, and all un-necessary services and products either removed or disabled.

Essentially, rather than assuring the COTS, we propose using the developed safety-related code to protect against failures which may impact upon the safety functions provided (the wrapper argument). In addition to providing protection to the safety functions, it is essential to protect safety-related input and output data to and from the SIL 2 software, because this safety-related data must pass through the untrusted COTS OS. To overcome this, guidance is drawn from the CENELEC vital communications standard EN 50159-2. It must also be noted that special consideration needs to be given to the Human-Machine interface (HMI), as often it will rely heavily on interaction with many libraries and un-trusted screen elements from the COTS OS. This is demonstrated in the analysis below where specific constraints are placed on the interaction sequence.

An added benefit of the wrapper / isolation approach is that, as the developed SIL 2 software code's interface is to the OS only, with no direct interface to the hardware;

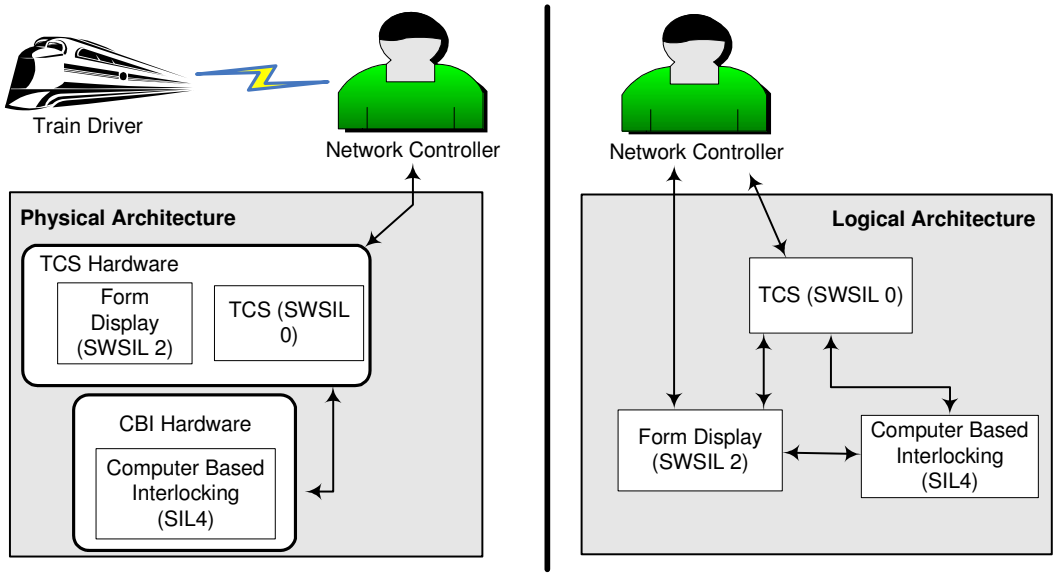


**Figure 1: Train Movement Authority Management System**

this simplifies and limits the need to assess hardware interactions.

### 6.1 Example: Train Order Management System

The example train movement authority system examined in the previous work [Connelly10] represented a centralised train control system of signalled track, with limited modification to the trackside infrastructure, the only change was the addition of track blocking and detection resets. This model has been expanded to examine management of non-signalled or "dark territory" through the use of limited trackside infrastructure with a similar concept. The analysis has been updated to take into account this operational context, and demonstration provided that the same safety requirements are appropriate in this context. A high level diagram of the example train order system is provided in Figure 1, where a SIL 0 and SIL 2 component are being executed on the same COTS OS. The logical interactions are also provided to demonstrate that the SIL rated components have separate logical communication channels.

The example system is configured as follows:

1. Safety-critical interface to trackside infrastructure is a SIL 4 interlocking, which manages validation of authorities for issuance to rail traffic and ensures points are set appropriately prior to issuing the authority to TCS for delivery; the connected infrastructure is limited to overswitch track circuits and points machines; and
2. An interface is provided to a central Train Control System (TCS), which allows the network controller to request an authority for a train, and receive a validated form for delivery to a train driver.
3. A form is delivered to a train driver via a voice communication channel. The driver is required to record each form field on a local paper copy of the form. A form is only considered "issued" and valid for execution when the network controller confirms a correct readback from the train driver.
4. The network controller confirms successful readback of the form via the TCS to the interlocking. The TCS runs on a Microsoft Windows XP PC.

As identified in the previous analysis, traditional signalling systems rely on an interlocking design such that all controls from TCS are validated and confirmed as safe prior to modifying track status. Such is not possible in train order working, as whilst the interlocking is capable of validating an authority is safe for issuance, based on the safeworking rules, it cannot determine the current location of the train being issued an authority, or of any conflicting trains. Additionally the interlocking system cannot issue an authority directly to the train driver (as opposed to clearing signals along a route). The TCS is therefore required to allow for a network controller to confirm train location, and be provided with safeworking forms for issuance to the driver.

Should the TCS corrupt location or form information the validation functions performed by the interlocking cannot be assured to prevent conflicting authorities. As a result the safety-related data is confirmed through the use of the forms display SWSIL 2 component.

As there are many pre-existing TCS products in use in active railways, it is considered favourable from a user interaction point of view to unify the interface between control of train order and signalled area. As such providing the ability to a "safety kernel" to run on the TCS managing the safety-related issuance of authorities to trains allows network operators to leverage existing systems with minimal extra training or hardware requirements. As the validation of requests is managed by the SIL 4 interlocking, analysis has determined that the safety kernel is required to achieve SIL 2 or higher. We refer to this safety kernel as "Safety Display" in Figure 1.

The safety functions provided by the kernel are limited to correct display of validated authority information, and return of network controller confirmation or rejection of the issuance to a train driver via voice. For the purposes of this paper, incorrect or unsafe requests can be considered mitigated by the interlocking design. As

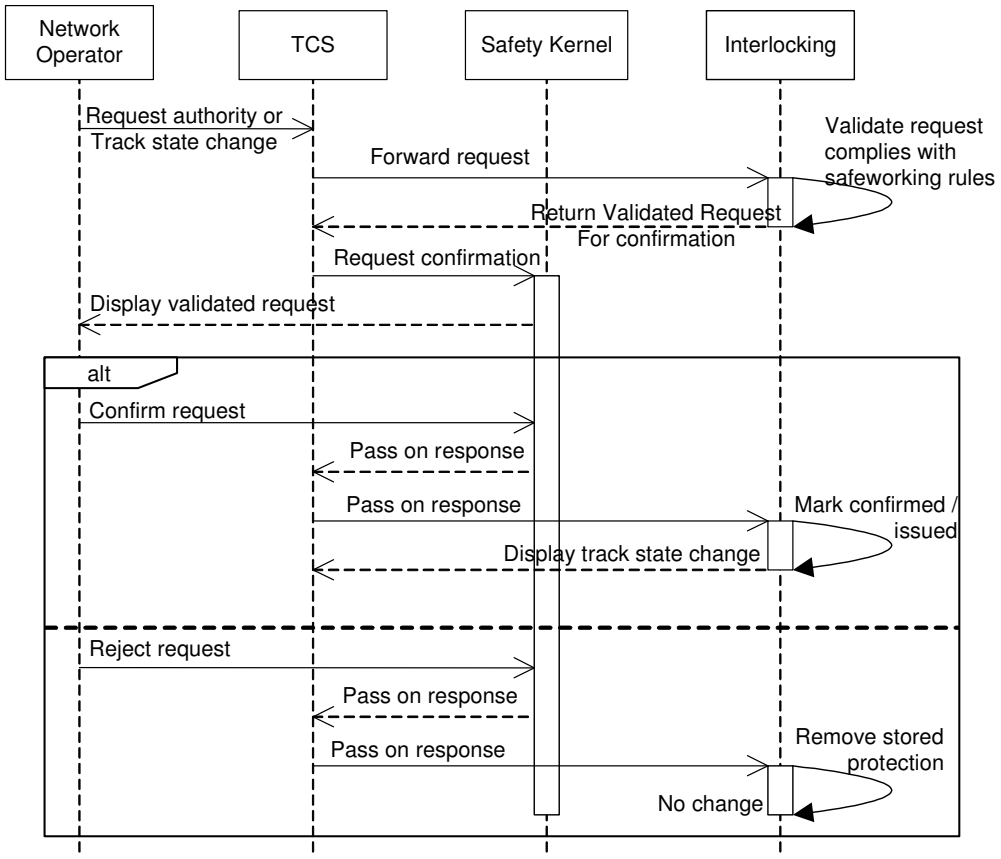


**Figure 2: Sequence of Interaction**

railway signalling systems are not run in strict real time, it is possible to design these functions such that they can be requested via the existing SIL 0 interface, and confirmed through the safety kernel. The development of a physically separate SIL 2 interface to the interlocking was considered and rejected as unnecessarily obfuscating the Network Controller's (NC) interaction workflows. Note that alarms require special consideration within the analysis, as they do have a timeliness component. A high level design of the sequence of interaction for each the safety function is shown in Figure 2.

The fundamental strategy is that no change to the protection managed within the interlocking is allowed to progress without user confirmation through the "trusted" SIL 2 kernel, justifying that the remainder of the TCS does not need to achieve any integrity level.

To demonstrate independence of the safety kernel from the SIL 0 TCS and OS the following activities were revisited in light of the modified context: perform hazard analysis on the safety functions provided by TCS, identifying the COTS causes for the hazards; and mitigate each cause in the safety kernel. Following the analysis, the hazards detailed in Table 1 were identified on the interface between SIL 0 and SIL 2 functionality (i.e. between TCS and the Forms Display).

| ID | Description | Cause(s) | Safety Requirement(s) |
|---|---|---|---|
| HAZ 1 | Confirmation / Rejection is modified by TCS in transit to the interlocking | COTS6 | HMI1, HMI8 |
| HAZ 2 | Safety Kernel SW fault corrupts message (unsafe) | COTS1, COTS2, COTS3 | HMI1, HMI2, HMI3, HMI4 |
| HAZ 3 | Unrelated NC HMI interaction leads to inadvertent confirmation | COTS4 | HMI5, HMI7 |
| HAZ 4 | TCS responds to confirmation request spuriously | COTS5, COTS7, COTS10 | HMI8, HMI9 |
| HAZ 5 | Multiple HMI failures (SIL 0 code) confirm dialog | COTS4, | HMI6 |

**Table 1: Identified hazards**

| Threat | Interpretation | Relevant failures from Table 3 |
|---|---|---|
| Repetition | Previously correct message is resent out of context | COTS5 |
| Deletion | Message to or from Safety Kernel is deleted by SIL 0 components | COTS8 and COTS9 |
| Insertion | Message to Interlocking is generated by SIL 0 components | COTS7 and COTS10 |
| Re-sequence | Messages from Safety Kernel have been changed out of sequence by SIL 0 components | COTS8 and COTS9 |
| Corruption | Messages to or from Safety Kernel are corrupted by SIL 0 components | COTS1, and COTS6 |
| Delay | Message is delayed, considered to be the same as deletion. | COTS8 and COTS9 |
| Masquerade | SIL 0 components attempt to perform functions which the Interlocking is expecting the Safety Kernel to perform. | COTS4, COTS5, COTS7, and COTS10 |

**Table 2: Treatment of EN 50159-2 data transmission integrity threats**

Causes of these hazards were identified within the COTS products and addressed as shown in Table 3, the selection of probable failure modes was based upon the "operating system failure modes" detailed in [Pierce02]. When determining the possible COTS failures, consideration was given to the identified basic message errors, or threats, defined in Clause 5 of EN 50159-2, which deals with transmission systems (shown in Table 2).

| ID | SIL 0 / COTS Software Failure | Possible Effect on SIL 2 element | Safety Requirement(s) |
|---|---|---|---|
| COTS1 | Corruption of incoming message from interlocking<br>(HAZ2) | Displayed information may not precisely match information stored in interlocking. May lead to confirmation of unsafe state change. | **HMI1:** Data correctness and integrity shall be confirmed through a sufficiently strong HASH / CRC of all data stored in the message. This shall be repeated during Safety Kernel processing, to detect intermediate memory interference. |
| COTS2 | Corruption of message during processing within Safety Kernel as a result of inappropriate memory access by SIL 0 elements. Could occur at any time, and may occur on volatile or non-volatile memory.<br>(HAZ2) | Displayed information may not precisely match information stored in interlocking. May lead to confirmation of unsafe state change. | **HMI1**<br>**HMI2:** The Safety Kernel is run as a separate process to the TCS, utilising Operating System Level memory and execution protection. |
| COTS3 | SIL 0 Elements may interfere with rendering of data in Operating System level dialog display, causing function to fail in a manner which may modify displayed data.<br>NB: This could occur as an OS level failure regardless of whether there was other SIL 0 code running or not<br>(HAZ2) | Displayed information may not precisely match information stored in interlocking. May lead to issue of incorrect information, or inability to detect operator error. | **HMI3:** Prior to delivery to the OS dialog renderer the safety-related data shall be rastererised to images (e.g. bitmaps) from a verified library of individual character images. Any image level corruption will be visually detectable, or insufficient to modify the data meaning.<br>**HMI4:** Design of rendered information shall be sufficient to mitigate undetectable modification of bitmap location i.e. data transposition / removal. |

| ID | SIL 0 / COTS Software Failure | Possible Effect on SIL 2 element | Safety Requirement(s) |
|---|---|---|---|
| COTS4 | Required confirmation response to Safety Kernel may be triggered by SIL 0 elements, or by NC during unrelated interaction with HMI.<br>NB: This could occur as an OS level failure regardless of whether there was other SIL 0 code running or not<br>(HAZ3, HAZ5) | Network Controller may not have sufficient time to interpret, or see all data. If state change is one which modifies track protection (e.g. logging train off, removal of track block) Network Controllers may make unsafe decisions. | **HMI5:** The safety display is designed such that keyboard entry is disabled, meaning that should the window take focus during unrelated data entry, the NC can't accidently cancel or confirm the state change.<br>**HMI6:** The Safety Kernel interactions shall be such that at least three Windows events (related to the NC confirmation action) are received, in the correct sequence, prior to confirmation of state change.<br>**HMI7:** Confirmation interactions for state changes shall be at least two discrete user interactions with the Safety Kernel dialog, geographically separated on the screen.<br>*NB: HMI6 and HMI7 are based on FTA not presented in this paper* |
| COTS5 | Message from Safety Kernel is cached by SIL 0 elements, and subsequently resent to the INTERLOCKING. Alternatively the SIL 0 elements may cause messages to be sent out of sequence.<br>(HAZ4) | Message may match outstanding response, and incorrectly confirm / cancel state change | **HMI8:** NONCE is included in return message. Should this NONCE not match the expected number then message will be rejected by the interlocking |
| COTS6 | Message from Safety Kernel is corrupted during transmission through TCS subsystem<br>(HAZ1) | Confirmation may be changed to rejection and vice versa | **HMI1**<br>**HMI8** |
| COTS7 | Message generated to INTERLOCKING by SIL 0 elements through some internal failure<br>(HAZ4) | Message may match outstanding response, and incorrectly confirm / cancel state change | **HMI9:** Messages shall undergo endpoint authentication between the interlocking and the Safety Kernel subsystems.<br>Message Authentication prevents messages from SIL 0 elements being treated as valid by either the interlocking or Safety Kernel. |
| COTS8 | Failure of SIL 0 elements interacts with Safety Kernel<br>(Fail safe, no hazard) | Possible failure to send or receive Safety Kernel messages. Alternatively messages may be sent out of sequence. | **HMI8**<br>**N/A:** TCS Backend Failure: no messages will be sent to or from the Safety Kernel – Fail safe state. |
| COTS9 | SIL 0 elements consume all TCS hardware resources (Safety Kernel process starvation)<br>(Fail safe, no hazard) | Safety Kernel may not receive or respond to messages. Alternatively messages may be sent out of sequence. | **N/A:** Fail Safe state for the TCS hardware, as the interlocking does not modify protection should confirmation not be received. |

| ID | SIL 0 / COTS Software Failure | Possible Effect on SIL 2 element | Safety Requirement(s) |
|---|---|---|---|
| COTS10 | SIL 0 element generates a message to Safety Kernel through some internal failure (although highly unlikely this is considered to be a credible failure mode) (HAZ4) | Safety Kernel may respond to message, which is passed onto interlocking, and interpreted as valid. If state change is one which modifies track protection (e.g. reset of track detection, removal of track block) Network Controllers may make unsafe decisions. | **HMI9** |

**Table 3: Safety Kernel data integrity protection from SIL 0 failure**

Based on the above analysis, the nine identified HMI safety requirements must be implemented in order to achieve SIL 2 for the safety kernel. With these safety requirements in place, and confirmation via a Fault Tree Analysis, an argument can be presented that the integrity of the identified safety kernel is commensurate with EN 50128 SIL 2.

## 7 Issues with Alarms

Whilst the fundamental concept is that the system must be able to fail safe, in the example train order system discussed above, it was identified that the concept presented some issues with alarm management. In all cases where a network controller has requested a change of state, should the system fail to present confirmation; the railway will remain in a safe state. Where the interlocking needs to alert the network controller of a failed railway state however, this approach is not wholly appropriate.

If a train is travelling on an existing authority, the points have been confirmed by the interlocking to be in an appropriate lie for that authority. Should the interlocking then either lose detection of those points, or detect them in the incorrect lie, the train cannot be protected through any means other than the network controller advising them of the situation. As such COTS1, COTS6, COTS8 and COTS9 need to be re-examined. To partially mitigate this risk a further Safety Requirement was identified.

> **HMI10:** The Safety Kernel shall display any safety related alarms with priority over all other messages from the interlocking

Should the TCS be unavailable, this alarm fail to be delivered, or the alarm is corrupted such that it is rejected by the system, HMI10 is insufficient to ensure the train driver can be notified within sufficient time. To ensure appropriate protection, an external mitigation was identified:

> **Application Condition:** All time-sensitive safety alarms shall require acknowledgement within a specified time, should network controller confirmation not be provided, a control centre alarm shall be raised external to TCS to ensure rail traffic can be protected.

This analysis highlights the importance of consideration of the whole of system safety argument when assessing COTS failures within a single subsystem.

## 8 Further Limitations and Issues

Further to the identified issues with presentation of time-sensitive data, there are several further limitations to the applicability of the strategy. Specifically:

1. Systems of this nature need to have a fail safe state, or have sufficient external mitigations (as for alarms above);
2. Great reliance is placed on the human-in-the-loop, both to detect system failure and to perform actions correctly;
3. At higher integrity levels (SIL3 or 4),
    a. the required integrity from the operating system is not considered justifiable due to the partial reliance on process execution integrity and separation;
    b. higher integrity is required from the hardware, e.g. 2-out-of-2 processor architecture.
4. Some integrity is assumed of the operating system. In particular that the SIL 2 binary code shall execute unperturbed by untrusted code, and that memory will remain unchanged during active execution The assumption on process protection is based on the lifetime of the Windows NT Kernel, and maturity of Windows XP; and
5. Should rich data entry be required, further analysis of the COTS failure modes would need to be conducted.
6. Any changes to the OS configuration (upgrades and patches) will need to be assessed to confirm that they do not impact on the safety kernel argument, as such may change the low level process execution behaviour of the OS.
7. Anti-Virus protection systems present difficulties with the approach detailed in this paper, as by design they have low level access to programs under execution, and can impact on the operating system's scheduling and interrupt executing processes. The current approach has been to forbid anti-virus protection systems as the product under development exists in a completely isolated and controlled network. It is not expected that this approach will be suitable for all applications, and as such analysis of the

possible interactions with anti-virus products will need to be conducted for more general roll-out.

## 9 Conclusions

This paper has presented further evidence of a practical approach to arguing for a COTS OS used to enable safety-related applications up to SIL 2. Two such systems are currently under development, and the approach determined sound by separate third party independent safety assessors. Therefore it is believed that when used within the stated limitations it is expected that the COTS OS approach described will result in a suitably safe system, whilst providing significant cost benefit to projects, and to customers in various industries.

Further work is required to apply this approach to real-time applications or ones requiring integrity greater than SIL 2.

## 10 References

R.H. Pierce, Preliminary assessment of Linux for safety related systems, 2002, UK HSE Research Report 011.

C. Jones, R. Bloomfield, P. Froome & P. Bishop, Methods for assessing the safety integrity of safety-related software of uncertain pedigree (SOUP), 2001, Contract Research Report 337.

C. O'Halloran. Assessing Safety Critical COTS Systems. Journal of the System Safety Society, 35(2), 1999.

Certification Authorities Software Team, Use of a Level D Commercial Off-the-Shelf Operating System in Systems with Other Software of Levels C and/or D, CAST-14, June 2002.

United States of America Department of Defense. MIL-STD-882: System Safety Program Requirements.

United Kingdom Ministry of Defence. DEF STAN 00-56: Safety Management of Defence Systems. 2007.

Defence Science Technology Organisation. Def (Aust) 5679: The Procurement Of Computer-based Safety Critical Systems. DSTO, 1998.

Radio Technical Commission for Aeronautics. DO178B: Software Considerations in Airborne Systems and Equipment Certification. 1992.

International Electro-technical Commission. IEC61508: Functional Safety: Safety Related Systems. IEC, 1995.

CENELEC. EN 50128: Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems. 2001.

CENELEC EN 50159-2: Railway applications – Communication, signalling and processing systems – Part 2: Safety-related communication in open transmission systems. 2002.

S. Connelly, H. Becht, Arguing for the use of COTS operating systems with safety-related software, ISSC 28 2010.