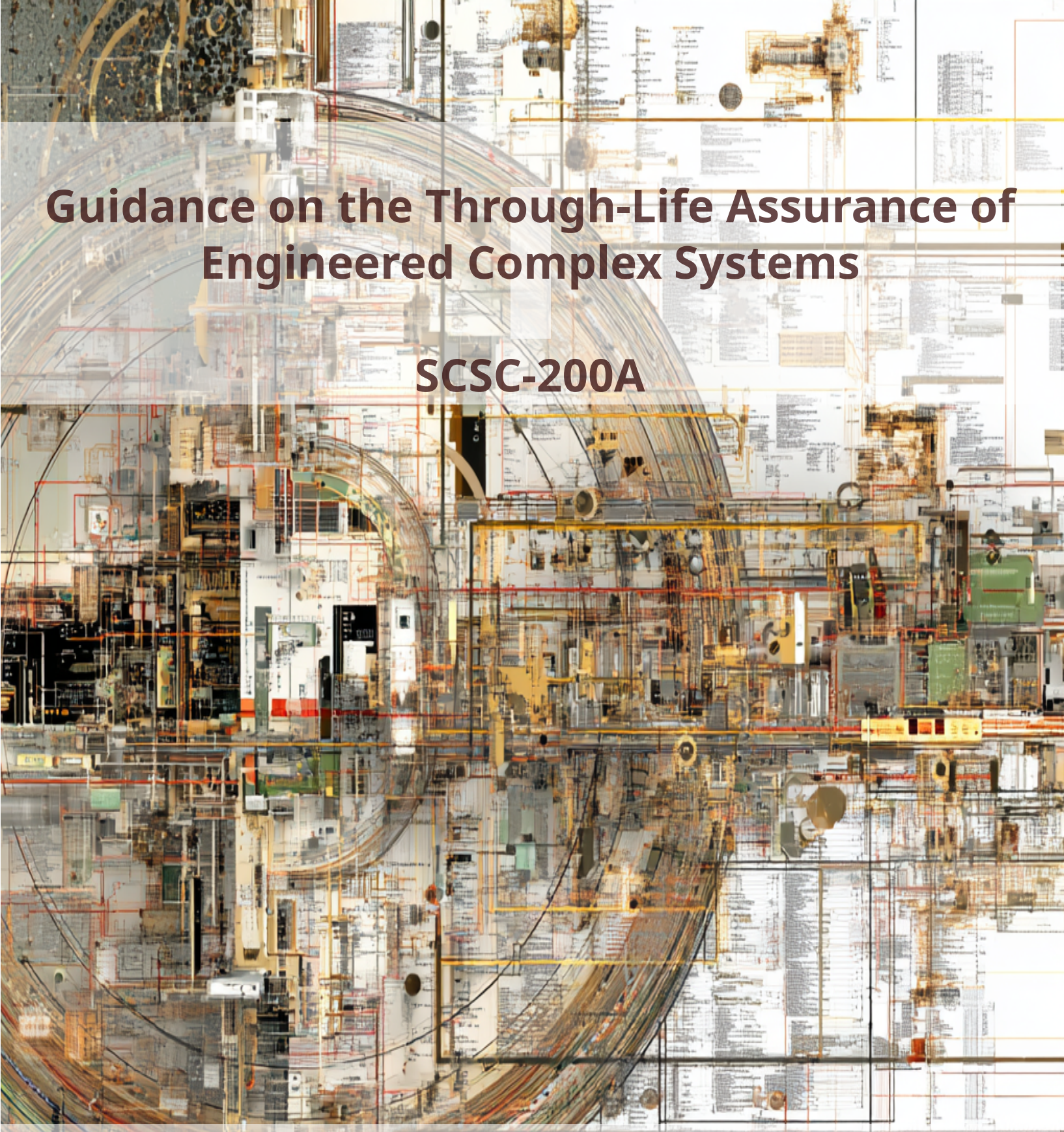
Guidance on the Through-Life Assurance of Engineered Complex Systems

SCSC-200A



SCSC
Safer Complex Systems
Working Group
Edition 2.0

This work is licensed under the Creative Commons Attribution 4.0 International Licence. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA. You are free to share the material in any form and adapt the material for any purpose providing you attribute the material to the Safety-Critical Systems Club (SCSC) Safer Complex Systems Working Group, reference the source material, include the licence details above, and indicate if any changes were made. See the licence for full details.

Disclaimer

While the authors and the publishers have used reasonable endeavours to ensure that the information and guidance given in this work is correct, all parties must rely on their own skill and judgement when making use of this work and obtain professional or specialist advice before taking, or refraining from, any action on the basis of the content of this work. Neither the authors nor the publishers make any representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability or availability with respect to such information and guidance for any purpose, and they will not be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever (including as a result of negligence) arising out of, or in connection with, the use of this work. The views and opinions expressed in this publication are those of the authors and do not necessarily reflect those of their employers, the SCSC or other organisations.

The Safety-Critical Systems Club (SCSC)

The SCSC, based in the UK, brings together a wide range of safety-critical systems specialists, including practitioners from a variety of industries, academics researching safety-critical systems, providers of the tools and services needed to develop the systems, and the regulators who oversee safety.

The SCSC focusses on current and emerging practices in safety engineering, software engineering, and product and process safety standards.

Through the annual Safety-Critical Systems Symposium (SSS) and its publications, seminars, workshops, tutorials and the collaborative features of its website, it provides opportunities to network and share knowledge on best practices, emerging techniques, etc.

The Safer Complex Systems Working Group

The Safer Complex Systems Working Group (SCSWG) was founded in 2023 and comprises engineers and specialists from a range of backgrounds across industry, consultancy and academia. The working group meets regularly to evaluate current and emerging safety techniques for complex systems.

The working group collaborates with other bodies working in the area of complex systems, including INCOSE, WICI, IET and RAE and keeps track of presentations and other events relating to safety organised by other professional bodies such as the Safety and Reliability Society (SaRS).

Contributors

A list of contributors to this guidance is provided in Appendix F on page 77.

Contents

1 Introduction **4**
- 1.1 This document . . . 4
- 1.2 Readership . . . 4
- 1.3 The need for this guidance . . . 4
- 1.4 Scope . . . 5
- 1.5 Terminology . . . 5
- 1.6 Using this guidance . . . 8
- 1.7 Changes in this edition . . . 8

2 Characteristics of Complex Systems **9**
- 2.1 Introduction . . . 9
- 2.2 Characteristics . . . 9
- 2.3 Complex System Examples . . . 14

3 Assurance Principles & Objectives **16**
- 3.1 Assurance Factors . . . 16
- 3.2 Levels of Assurance . . . 17
- 3.3 Assurance Principles . . . 17
- 3.4 Mapping Principles to LAACs . . . 25
- 3.5 Determining Fitness for Deployment . . . 29

4 Scenarios **30**
- 4.1 Complex System Scenarios . . . 30
- 4.2 Mapping of Scenarios to Principles and Objectives . . . 32

5 Through-Life Assurance of a Complex System **41**
- 5.1 Introduction . . . 41
- 5.2 Assurance Challenges . . . 41
- 5.3 Recommendations . . . 42
- 5.4 Case Study . . . 42

6 Further Work **43**
- 6.1 Questions Regarding the Scope of this Guidance . . . 43
- 6.2 Research Topics . . . 44

Appendix A Potential Analysis Techniques **46**
- A.1 Hazard & Risk Identification . . . 47
- A.2 Assurance Requirements Capture . . . 48
- A.3 System Analysis Techniques . . . 48
- A.4 System Resilience Techniques . . . 56

Appendix B Analysing Complex Systems **59**
B.1 Analysis: Scenario S-1: Build New . . . 59
B.2 Analysis: Scenario S-2: Assemble . . . 59
B.3 Analysis: Scenario S-3: Restructure . . . 60
B.4 Analysis: Scenario S-4: Evolve . . . 60
B.5 Analysis: Scenario S-5: Assure . . . 60
B.6 Analysis: Scenario S-6: Use New . . . 60
B.7 Analysis: Scenario S-7: Degrade . . . 60
B.8 Analysis: Scenario S-8: Investigate . . . 60
B.9 Analysis: Scenario S-0: Check . . . 61

Appendix C Example Analysis **63**
C.1 Motorway Active Traffic Management . . . 63

Appendix D Limitations of Traditional Approaches **68**
D.1 Development Methodologies . . . 68
D.2 Design Modelling Approaches . . . 68
D.3 Verification Approaches . . . 69

Appendix E Sample Incidents in Complex Systems **71**
E.1 Introduction . . . 71
E.2 The Herald of Free Enterprise . . . 71
E.3 CrowdStrike Software Update . . . 72
E.4 Boeing 737 Max 8 MCAS . . . 72
E.5 The R.101 Airship . . . 73
E.6 Clapham Junction Rail Crash . . . 74
E.7 Common Themes . . . 75

Appendix F Preparation of this Document **77**

Appendix G Document History **78**

Chapter 1

Introduction

Contents

1.1 This document 4
1.2 Readership 4
1.3 The need for this guidance 4
1.4 Scope 5
1.5 Terminology 5
1.6 Using this guidance 8
1.7 Changes in this edition 8

1.1 This document

This is the second edition of the guidance prepared by the Safer Complex Systems Working Group, representing the point reached at the end of the Working Group's second year of operation. Section 1.7 below outlines the primary differences between this issue and the first issue released in February 2025.

Chapter 6 lists some of the topics on which the Working Group will be concentrating in 2026.

1.2 Readership

This guidance is intended for safety professionals. It is not intended to offer an introduction to safety analysis, rather to augment existing safety analysis guidance in the context of complex systems.

It is assumed that you have been assigned the task of determining whether a complex system is, or is not, adequately safe. This may be a system under development or an existing system that needs to be reässessed, possibly because of a safety-affecting incident. This guidance provides a framework and a description of some techniques which you may find useful.

The techniques described in this guidance may also be of use if you are assessing a property other than safety of the system: perhaps whether it is adequately secure or adequately staffed.

1.3 The need for this guidance

Complex systems being built and in use today cannot be adequately analysed by using "traditional" tools and techniques, since, among other properties, they exhibit emergent behaviours (i.e. behaviours that cannot be explained by examining the system's individual components). This guidance describes tools and techniques that

can either replace or augment traditional ones to take better account of a system's complexity throughout its lifecycle. Appendix D on page 68 lists some of the inadequacies of traditional techniques.

This guidance has been written in this context: *how can we justify the safe operation of a complex system given that the traditional assumptions are no longer valid?*

1.4 Scope

This guidance discusses the safety of complex *systems*. It may also be useful for assurance of complex services, procurements and processes.

The security and other properties of complex systems are covered only where they may directly change the intended behaviour of the system. For example, a security breach might directly cause harm by exposing a person's private information, or by allowing a hacker to take control of a vehicle.

1.5 Terminology

This guidance relates to the safety of complex systems, but it is difficult to define precisely what constitutes a "complex system". For this reason, Chapter 2 gives examples of complex systems and the characteristics that make them "complex" in the meaning used within this guidance.

The acronym **SCS** is used to represent "Safer Complex System" — the target of the guidance in this document. Other terms, including "system", are used in this document as below:

- **Assurance Position.** The current status of the safety argument, changing as the analysis progresses: how close is the safety claim to being justified or rejected?
- **Assurance Requirement.** A requirement supporting the assurance (rather than its functionality) claim of a system. Such requirements cover the transparency, verifiability, auditability, accountability, access for independent scrutiny and security of the system.
- **Assurance Wrapper.** The supplementary assurance that enables the "wrapped" component (which may be a subsystem, subservice, etc.) to be used by the SCS while maintaining the overall SCS assurance.

  For example, if a system S_n is to be integrated into an existing system S_e, then it may be that the assurance provided with S_n is insufficient to maintain the Assurance Position (*quod vide*) of S_e. In this case, a "wrapper" may be provided or constructed to provide the missing assurance by "wrapping" S_n.
- **Complicated System.** Chapter 2 provides characteristics and examples of **complex** systems as the term is used in this document. There are, however, systems which appear superficially to be complex, but are better termed "complicated". For example, a (petrol) car engine is complicated, whereas traffic is complex.

  Reference [1] differentiates between "complex" and "complicated" with the following example. *"The [behaviour of the] atmosphere is difficult to predict because it is both complicated and complex. It is complicated in that it encompasses many interconnected physical aspects such as air movement, pressure, humidity, gravitational forces, rotation, evaporation and condensation, radiation, diffusion, convection and friction. And it is also a complex system, one in which larger-scale behaviours and properties like hurricanes, the jet stream and El Niño emerge from the combination of these local physical processes acting together."*

  Reference [2] differentiates between the terms as follows:

  - A **simple** system has elements, the relationship between the states of which, once observed, are readily comprehended.
  - A **complicated** system has elements, the relationship between the states of which can be unfolded and comprehended, leading to sufficient certainty between cause and effect.
  - A **complex** system has elements, the relationship between the states of which are weaved together so that they are not fully comprehended, leading to insufficient certainty between cause and effect.

Reference [2] then makes the point that *"traditional systems engineering approaches, which assume some form of order and deterministic behavior so the relationship between cause and effect is understood, do not handle complex systems engineering well."* See Appendix D of this guidance.

Reference [3] describes the Cynefin[1] framework which adds "chaotic" to "simple", "complicated" and "complex" and includes "disorder" for the situation where the system is not clearly in any of the other four states.

- **Contingency Mode.** A mode of operation when the service or function cannot be maintained and different elements or actions are required to maintain some level of service or function.

  For example, in a railway system, if there is a complete failure of the signalling system, some level of service could still be provided by reverting to manual signalling: a contingency mode. See also "Degraded Mode".

- **Complex and Unintentionally Fragile (CUF).** When CrowdStrike released an upgrade of its Falcon software containing a bug in July 2024, many systems on computers running Microsoft Windows were brought down. This particularly affected systems in airlines, hospitals and banks. The distribution and installation system could be considered CUF.

  Another example is the air traffic control outage in the UK in August 2023 caused by a single duplicated waypoint in a single flight plan. The system concerned (FPRSA-R) had successfully processed 15 million flight plans over the 5 years it had been in service, but, encountering the ambiguous waypoint identifier led to a 'critical exception' where both the primary system and its backup entered a fail-safe mode and stopped providing service. This caused more than 1 500 scheduled flights to be cancelled and 575 more to be delayed.

- **Degraded Mode.** A mode of operation of a system that is invoked in the event of an inability to provide full service, possibly because of the failure of one or more elements.

  In general, a degraded mode offers less service than full operating mode, but more service than a Contingency Mode *(quod vide)*.

- **Emergent Behaviour.** A behaviour of a system that its components do not have on their own, and which emerges only when they interact in the wider system.

  Some emergent behaviour is deliberate in the system design. For example, the thrust produced by a jet engine does not arise from any of its individual components, but by their interaction. This behaviour was required by the system design.

  In this guidance the term "emergent behaviour" is used for system behaviour that is unanticipated and unexpected, but that can be explained once it has occurred. In the literature, this is known as "weak emergence". An example is one driver on a motorway illuminating the brake lights for a short time and causing a traffic jam even when there is no particular obstruction. The interacting components of the system retain their independence, but the system as a whole behaves in a manner unanticipated from knowledge of the elements alone.

  This identification of emergent behaviour is not new: Aristotle said *"the whole is something over and above its parts, and not just the sum of them all"* and reference [4] was published in 1879 by George Lewis and states, *"Although each effect is the resultant of its components, we cannot always trace the steps of the process, so as to see in the product the mode of operation of each factor. In the latter case, we propose to call the effect an emergent."*

- **Harm.** The purpose of the guidance in this document is to avoid a system causing harm. Harm can include physical or emotional harm to a person; harm to the environment (e.g., oil spill at sea); or financial or reputational harm to a person or company. When applying the guidance in this document, it is important to define the precise harm that is to be avoided.

  ISO 26262:2018 restricts "harm" to *"physical injury or damage to persons"* but ISO/IEC Guide 51:2014 provides a wider definition as *"injury or damage to the health of people, or damage to property or the environment."*

[1] Welsh for "habitat".

UL 4600 is proposing moving away from the concept of "harm", replacing it with the idea of "loss". In reference [5] a loss is defined as *"an adverse outcome, including damage to the system itself, negative societal externalities, damage to property, damage to the environment, injury or death to animals, and injury or death to people."*

- **LAAC.** Levels of Assurance Addressing Complexity. See Section 3.2 on page 17 for a definition of the levels.

- **Leading Indicator.** This term is used of a Safety Performance Indicator (SPI) *(q.v.)* which is monitored to detect trends which could affect the safety of the system. An example in the area of Active Traffic Management (see section 2.3.2 on page 15) is the time taken by an operator to respond to an identified incident. If, over time, this is found to be increasing, then it might indicate a reduction in system safety.

- **Level of Abstraction.** A complex system can be considered at a hierarchy of levels of abstraction. Consider Air Traffic Control (ATC) as an example. At its most abstract, this system accepts as input the location, altitude, speed and destination of a number of aircraft and outputs instructions for changes of altitude and direction to pilots. The system must satisfy the safety condition of keeping the aircraft apart and the functional requirement of getting aircraft to their destinations with the least fuel burn. This abstract description can be made less abstract by adding details of how the system locates aircraft (different types of transponders), how it deals with emergencies, how aircraft are handed from sector to sector, etc.

  Reference [6] explores this concept in detail. In particular, it points out that, at each level of abstraction, new emergent properties can arise.

  When considering a complex system, each analyst will have a mental model of the system at some level of abstraction. If this level is too high, it can lead to the analyst believing the complex system to be simpler than it is, resulting in an inadequate analysis.

- **Safety Baseline.** A definition of the level of existing safety against which proposed modifications can be measured. The safety baseline is established from the current or recent historic performance of the activity and includes the risks to all stakeholders.

- **Safety Case.** This term is used to describe a *"structured argument, supported by a body of evidence, that provides a compelling, comprehensive, and sound argument that safety engineering efforts have ensured a system meets a comprehensive set of safety constraints"* (definition taken from reference [7]).

- **Safety Performance Indicator (SPI).** This term is used to describe a metric that can be used to assess the safety of a deployed product or service. For example, the Canadian Nuclear Safety Commission defines the number of preventive maintenance activities that have been deferred as an SPI.[2] If the number of such deferrals is increasing over time, this may indicate an ineffective maintenance programme, which could lead to safety-related incidents.

  SPI's are of two types: Leading Indicators *(q.v.)* which are proactive, preventative and predictive measures and Lagging Indicators which are retroactive, collecting data on accidents that have occurred.

- **Safety Target.** This term includes a safety baseline and safety objective relevant to the activity.

- **SOTIF.** An acronym for "Safety of the Intended Functionality". This term arises from the observation that many dangerous situations occur, not as the result of component failure or malfunction, but when every component of the system operated exactly as it was intended. See reference [8] which considers *"the absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons."*

- **Stakeholder.** Any person or organisation that has an interest in, influence on, or is affected by, the behaviour of the system. In particular, this term includes any entity to which the system may cause harm. Note that the designers and developers of a system are also stakeholders.

[2] See `https://www.cnsc-ccsn.gc.ca/eng/resources/publications/reports/powerindustry/safety-performance-indicators/`

Note that a stakeholder may not realise that he or she is a stakeholder. A person whose flight was cancelled in July 2024 as a result of the CrowdStrike download of erroneous software to the air carrier's Microsoft Windows system suffered harm and was unwittingly a stakeholder in that system.

- **System.** The term used to include the socio-technical aspects under consideration. This includes the interaction between the human and technical artefacts with respect to the world.
- **System of Systems.** A system with many parts, some of which may be other systems or services. Reference [9] defines this term as *"A collection of component systems that can exist independently and have different owners who may have conflicting objectives."* In general, this term is avoided in this guidance.

1.6 Using this guidance

If you are unsure whether the system on which you are working is "complex" as the term is used in this guidance, then:

1. Map the characteristics of a complex system given in Chapter 2 to your system: do some, most or all of them apply? It is possible that some techniques described in this guidance may be applicable even to systems that do not meet the characteristics described in that chapter; it may be convenient to treat any system as being complex.
2. Use table 4.1 in Section 4.1 to find the activity that you are to perform: are you working on the design of a totally new system or investigating a safety-related incident on an existing system? In practice, it is unlikely that any one item in that table will match your activity exactly — there will almost certainly be overlap.
3. With this information, Chapter 4 lists the assurance principles that you might apply to your particular activity. More information about each assurance principle is given in Chapter 3.
4. Consider the guidance in Chapter 5 to prepare the assurance argument for the system throughout its life.

Appendix A provides a selection of tools and techniques that you might apply to produce those artefacts.

1.7 Changes in this edition

In addition to numerous editorial modifications, including changing the order of some chapters, the following changes have been made between edition 1 and edition 2 of this guidance:

- An example of applying the guidance to a possible extension of an existing complex system has been added as Appendix C.
- Common themes have been identified in the incidents described in Appendix E and listed in Appendix E.7.
- More applicable analysis techniques have been described in Appendix A including Systemic Contributors Analysis and Diagram (SCAD), Work Models that Compute (WMC) and the Analytic Hierarchy Process (AHP).
- Chapter 5 has been introduced as a placeholder for preparing the assurance argument.
- A high-level description of the work-items for each of the Scenarios listed in table 4.1 has been included as Appendix B.

Chapter 2

Characteristics of Complex Systems

Margaret J Wheatley

"Everyone in a complex system has a slightly different interpretation. The more interpretations we gather, the easier it becomes to gain a sense of the whole."

Contents

2.1 Introduction . . . 9
2.2 Characteristics . . . 9
2.2.1 System-level characteristics . . . 11
2.2.2 Design & implementation characteristics . . . 12
2.2.3 Verification characteristics . . . 13
2.2.4 Assurance characteristics . . . 13
2.3 Complex System Examples . . . 14
2.3.1 A Hospital . . . 14
2.3.2 Motorway Active Traffic Management . . . 15
2.3.3 Air Traffic Control (ATC) . . . 15

2.1 Introduction

The term "complex system" is used by different groups to cover very different things: for example, the human body, the earth's atmosphere, a country's economy and business cycles.

This guidance focuses on engineered complex systems: digital, socio-technical systems that are constructed through engineering practices.

It is difficult to define such a system and, in practice, it may be effective to apply the approaches listed in this guidance to any system, even if it doesn't meet a definition of being "complex".

Complex systems do have common characteristics and some of those characteristics are listed in Section 2.2 below, based in part on the list in reference [10].

2.2 Characteristics

Section 2.2.1 describes characteristics which can be considered to recognise that a system is complex. Sections 2.2.2 and 2.2.3 list some of the resulting characteristics that can be recognised during design and verification.

A survey of definitions of "complex system" from sources ranging from the Wikipedia to the Waterloo Institute for Complexity & Innovation (WICI) and the University of York indicates that the terms most commonly associated with complex systems are:

- **"emergent behaviour"**: system behaviour that could not have been predicted by examining its components individually. Reference [9], for example, states: *"A system is complex if some of its behaviours are emergent properties of the interactions between the parts of the system, where you would not be able to predict those behaviours from knowledge of the parts and their interactions alone."*

  IEC 63187-1 uses just the emergence property to define a complex system: *"[a] system where the collective behaviour of the system elements entails emergence of properties that can hardly, if at all, be inferred from properties of those system elements."*

- **"whole/part interactions"**: reference [11] states that: *"A system is said to be complex if there is a bidirectional non-separability between the identities of the parts and the identity of the whole. Then, not only the identity of the whole is determined by the constituent parts, but also the identity of the parts are determined by the whole."*

- **"non-linearity of effect"**: a small change to part of the system can cause a disproportionately large change in the system's behaviour. Reference [9], for example, states: *"Small perturbations may have a proportionate response, no response, or a disproportionate response (which might include a change in intended behaviour)."*

- **"coupled feedback"**: one element of the system feeds back positively to other parts which then affect the original element. Reference [9], for example, states: *"The system continuously cycles through similar states and there is both internal feedback and feedback via the environment where the output of the system forms, or influences, part of the input and the feedback paths can influence each other."* This is important for a complex system because, as reference [12] points out, *"In a feedback system there is no clear discrimination between "causes" and "effects", since the output influences the input."*

Reference [2] from the International Council on Systems Engineering (INCOSE) lists those characteristics and adds some more including:

- **"multiscale behaviour"**: the system's structure exists on many scales and characteristics are not reducible to only one level of description.

- **"adaptability"**: the system proactively and/or reactively changes function, relationships, and behaviour to balance changes in environment and application to achieve system goals. In socio-technical systems, these changes may result from humans adapting the work-flow for their own convenience or to improve local safety. As noted below, optimisation to local demands may result in global reduction in optimisation.

- **"indeterminate boundaries"**: complex system boundaries are intricately woven with their environment and other interacting systems.

In reference [13], Teiva Harsanyi adds further characteristics:

- **"delayed consequences":** actions don't always have immediate effects and consequences may become apparent only later. In complex systems, relying solely on immediate feedback can create a misleading sense of stability.

- **"local vs. global optimization":** optimising one part of a complex system doesn't necessarily improve the whole system, and in some cases, it can even make matters worse.

- **"hysteresis":** a system's past state continues to influence its behavior, even after the original cause is removed. Simply fixing the root cause is not always enough. Therefore, it's crucial to assess whether a system is prone to hysteresis and, if so, anticipate its effects.

The Royal Academy of Engineering (RAE) has published reference [14] which captures the roundtable discussions held in December 2024 between 27 engineers on the governance of complex systems.

Other characteristics that appear in the literature describing complex systems include randomness, collective dynamics, hierarchies, turbulence, autopoiesis, inertia, chaotic behaviour, domains of stability, tipping points, path-dependency and levels-and-hubs.

Note that emergent behaviour and non-linearity of effect may be positive (advantageous) or negative.

2.2.1 System-level characteristics

This section lists some characteristics of existing systems that may indicate that they are "complex". These may be of use in Scenario Types S-0 and S-3 to S-8 as defined in table 4.1 to determine whether the existing system is "complex". Note that any system may be treated as "complex" and explored using the techniques in this guidance.

Fundamental system-level characteristics

A complex system may:

SF1: exhibit emergent behaviours in its functionality, safety, reliability and other key properties.

SF2: comprise a combination of physical, digital and human independent agents interacting with each other and with the system as a whole (these are "digital, socio-technical" systems).

SF3: involve many interactions, both between system components (including the humans who are system components) and with entities outside the system boundary. These interactions may operate on a combination of products and services, defined by service-level agreements and contracts with corresponding legal and national aspects. A complex system has relationships with many stakeholders.

These interactions occur between all of the components identified in point SF2: and may cause coupled feedback.

SF4: perform many, potentially overlapping and interacting, functions.

SF5: have a blurred and difficult-to-define boundary. As reference [6] states: *"the location of the boundary is normally uncertain and moves as capabilities and interdependencies change, and as other parts of the network adapt."*

Other system-level characteristics

A complex system may:

S1: comprise large numbers of subsystems, parts and components.

S2: be subject to poorly-controlled scope creep.

S3: have been deployed for a long time and therefore contain elements of legacy technology whose design, construction, verification and justification may be lost.

S4: have been re-used or re-purposed without regard to the original intended function.

S5: contain elements developed by different parties (organisations, vendors, open source communities, etc.) which have inconsistent processes, methods, cultures and working languages. The assumptions relevant to each party need to be monitored throughout the life of the SCS.

S6: contain components that have undergone change individually, in an unsynchronised manner.

S7: be used in open or poorly controlled environments, in which it is hard to ascertain whether all the assumptions of its specification continue to be fulfilled.

S8: be geographically (possibly internationally) distributed.

S9: have requirement specifications that are obsolete, ambiguous, incomplete or inconsistent.

2.2.2 Design & implementation characteristics

The characteristics listed in this section are of concern to an engineer building a new complex system (Scenario Type S-1 in table 4.1), an engineer substantially modifying an existing one (Scenario Types S-3 and S-6) and an assurance engineer analysing the system (Scenario Types S-0 and S-5).

Fundamental design & implementation characteristics

A complex system may:

DF1: have functions supporting system resilience, that enable the system to respond to unexpected changes and continue to exhibit key parts of its functionality.

However, over time these may be eroded because (i) they may be rarely used and therefore not maintained, (ii) have been worked around, (iii) commercial or design constraints may determine they are superfluous, and also (iv) the understanding of the need for such functions may be lost.

DF2: need to handle failures or malfunctions of elements safely. This may involve "graceful degradation" of system functionality or "crash-only" behaviour.

DF3: make use of many levels of hidden implementation and abstraction, e.g., cloud-based software and hosting services.

An observer may see and understand only one level of abstraction.

DF4: use machine learning or other technologies which introduce opaque, non-deterministic behaviours.

DF5: have initialisation, configuration and parameterisation data that are difficult to produce and coördinate between elements, making the management of these data difficult.

DF6: involve significant risk when any change is made, because of unknown dependencies and interactions: it may be an accidental system.

Local changes, particularly those made by new maintenance personnel, may introduce adverse consequences through internal paths and interrelationships that were neither recognised nor documented.

Other design & implementation characteristics

A complex system may:

D1: have been subject to incremental evolution and enhancement without explicit design work or documentation.

D2: have parameterisation which includes synchronisation, tolerances, timing cycles and alignment. These interactions may operate on different timing cycles; timings may be asynchronous.

D3: be difficult to analyse to obtain an overall system view.

D4: involve time-consuming and difficult integration activities.

2.2.3 Verification characteristics

The characteristics listed in this section are of concern to an engineer analysing the behaviour of an existing system (including the history of its human interactions) and (possibly retrospectively) producing an assurance of adequate safety (Scenario Types S-0, S-5 and S-8 in table 4.1).

Because of their complex nature, these systems may:

V1: have weak verification evidence because traditional verification tools designed for small or non-complex systems do not work effectively on complex systems (see Appendix D.3 on page 69).

For example, some non-complex systems may be verified by dynamic testing or static analysis of the software source code. Neither of these techniques is appropriate for complex systems where dynamic testing cannot cover more than a tiny fraction of the system's state space and where static analysis cannot easily be coördinated across subsystems.

V2: rely on evidence based on high-level analysis, sampling or the use of nominal scenarios. This occurs because effective dynamic testing is not possible (as described above).

V3: have poor fail-over and fail-safe validation evidence because of the cost and difficulty of creating realistic testbeds and test harnesses.

V4: not be susceptible to analysis using standard analysis techniques.

Component- and causal-based analysis (FTA, ETA, FMEA, etc.) is inappropriate for complex systems: see Appendix D.3. Appendix A contains suggestions of analysis techniques which may be appropriate for complex systems.

V5: be difficult to test because there are many interfaces and connected systems that must be in place, or be simulated, for realistic testing.

Generating appropriate test data and test cases is difficult because of the multiplicity of interfaces and the nature of accidental systems.

2.2.4 Assurance characteristics

The characteristics listed in this section will affect an engineer preparing the assurance that the complex system is sufficiently safe: any of the Scenarios in table 4.1.

Fundamental assurance characteristics

The safety assurance for a complex system must include evidence that:

CF1: the Safety Targets are met for the overall system, noting that:

(a) there may be a different target for each stakeholder group, and

(b) these may be specified in terms of emergent behaviours (which will change during the life of the system).

CF2: the Safety Targets allocated to sub-parts of the system are consistent with the overall targets.

CF3: complexity properties such as emergence, non-linear effects, and whole/part interactions do not lead to unsafe behaviours.

CF4: there is sufficient dynamic information available in the running system to allow elements of the system to provide resilience when an unanticipated event occurs.

This relates particularly to humans who are part of a socio-technical system.

CF5: the coverage of technical, services and human elements such as skills, procedures and process is adequate.

CF6: the evidence captured from CF1 to CF5 is sufficient to demonstrate the system remains resilient during its lifetime.

The production of this evidence will have to rely on non-traditional techniques as outlined in Appendix A.

Other assurance characteristics

The assurance evidence may:

C1: be incomplete or have to rely on indirect evidence such as audits as some elements (e.g. consumed services implementation) will be hidden, some may be legacy, COTS or open-source and have limited assurance evidence and argument available.

C2: be distributed, fragmented and diverse, as different parts of the system may have evolved independently and have had assurance arguments produced at different times and possibly in different formats and notations. Notification of changes to assurance may be poor.

C3: need sophisticated tools for its management.

C4: have a modular assurance argument, relying on other parties to maintain some parts of the argument and their interfaces to the main assurance argument.

2.3 Complex System Examples

2.3.1 A Hospital

There are many subsystems within a hospital, including controlling the visitors' car park, maintaining the building, restaurant-like food preparation, reception of ambulances, calibrating equipment, training medical staff, hotel-like handling of bed linen, laboratory testing, billing, operating room procedures, handling of violent patients, reacting to sudden emergencies (Code Blue, etc.) etc.

Superficially, a hospital may be thought of as a complicated rather than complex system because there is limited interaction between the various subsystems — the management of the visitors' car park has generally little interaction with the operating room procedures.

However these subsystems are required to act together when an incident such as a fire in the building, an electrical power failure or hospital-wide computer outage occurs.

At a hospital in Ottawa, Canada, on an October afternoon, one of two transformers providing electrical power to the building unexpectedly failed and caught fire while the other was out of service being replaced. Although the emergency power system kicked in, it too was soon compromised both by the fire and by the water firefighters used to douse the fire.

Nurses and firefighters carried 17 babies out of the Neonatal Intensive Care Unit and down eight flights of stairs (the lifts were not working) to waiting ambulances. Two women in labour were also carried out by firefighters, while another stayed and gave birth at the hospital during the blackout. Many other acute emergency patients awaiting surgery were moved to other hospitals.

The power was out for 48 hours and during this time there were unforeseen interactions between all the various subsystems: the building maintenance staff were called on to install planks down flights of stairs to allow stretchers to be slid down, the car park was unlocked and became an active area, food services had to be provided to those working on the restoration of the power, etc.

Reference [15] describes the resilience required during normal and abnormal operation of an hospital. As that reference states: *"Resilience [of a hospital's systems] ...involves adaptation to varying demands and threats that are brought on by the continuing negotiation among competing economic and workload pressures."*

2.3.2 Motorway Active Traffic Management

This system is also known as an “Intelligent Transport System” or “Smart Motorway”.

Active Traffic Management is a system that receives input from sensors providing data about traffic flow on a motorway (in particular, cameras, radar sensors detecting stationary vehicles and radar sensors/inductive loops built into the road to detect the passage of vehicles) in order to smooth traffic flows and reduce accidents. The system can dynamically change the speed limits, change the use of the lanes on the motorway and, where fitted, control traffic lights permitting or preventing additional traffic from entering the motorway.

Reference [16] was prepared by the UK Department of Transport in 2020 and contains a description of the system and an assessment of its effectiveness in increasing road safety.

The system is a digital, socio-technical system with both automated outputs and outputs from humans working in control centres. The motorway environment being controlled is extremely dynamic with maintenance workers setting up and removing worksites, pedestrians straying (illegally) onto the motorway, vehicles stopped in a running lane, drivers with different skill levels, different vehicle types, etc.

There are also longer-term changes to the motorway environment: the increase in electrical vehicles (which have different crash characteristics), (semi-)autonomous vehicles and, in the future, truck platooning.

A possible extension to the current Active Traffic Management system is used as an example in Appendix C.1 on page 63.

2.3.3 Air Traffic Control (ATC)

The control of aircraft in flight in a particular area is a digital, socio-technical system with interfaces to pilots, adjoining ATC sectors, clearance delivery, ground traffic control, flight services, search-and-rescue and other organisations.

Interfaces to pilots includes control of aircraft flying under the instrument flight rules and assistance given to any pilot who declares an emergency. In some cases, if the pilot is too low to receive direct transmissions from ATC, the assistance has to be provided via over-flying aircraft.

Reference [17] states:

> *This paper presents the viewpoint that an air traffic management system is emergent, i.e., exhibiting behaviors at the system-wide level that emerge from the combined actions of individuals within the system. Emergence carries with it the additional implication that these phenomena typically cannot be predicted by examining the individuals' behavior alone.*

This is confirmed in [18] which states,

> *Commercial aviation is feasible thanks to the complex socio-technical air transportation system, which involves interactions between human operators, technical systems, and procedures. In view of the expected growth in commercial aviation, significant changes in this socio-technical system are in development both in the USA and Europe. Such a complex socio-technical system may generate various types of emergent behavior, which may range from simple emergence, through weak emergence, up to strong emergence.*

Reference [18] also points out that, while some of the emergent behaviour will be negative, other may be positive, leading to the system itself finding safe solutions to unanticipated situations. Using Monte-Carlo simulation (see Appendix A.3.4), that paper identifies potential emergent behaviour in the coördination of aircraft crossing an active runway.

Chapter 3

Assurance Principles & Objectives

Niklaus Wirth

"The belief that complex systems require armies of designers and programmers is wrong. A system that is not understood in its entirety, or at least to a significant degree of detail by a single individual, should probably not be built."

Contents

3.1 Assurance Factors . . . 16
3.2 Levels of Assurance . . . 17
3.3 Assurance Principles . . . 17
3.4 Mapping Principles to LAACs . . . 25
3.5 Determining Fitness for Deployment . . . 29

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this chapter are to be interpreted as described in RFC 2119 (reference [19]) and clarified in RFC 8174 (reference [20]) if the associated requirement is relevant for the SCS.

3.1 Assurance Factors

In assessing the necessary assurance for a complex system, various factors have to be taken into account:

- What motivated the assurance analysis? The scenarios considered in this guidance are listed in table 4.1 on page 31.
- How much harm could the system do? Levels of harm are considered in Section 3.2.
- How complex is the system?
- What types of assurance analysis apply to the system? Section 3.3, and in particular tables 3.1 to 3.7, cover this.
- What types of analysis are recommended for a system with a particular level of potential harm? Section 3.4 and tables 3.8 to 3.14 address this question.
- For a particular Scenario (as listed in table 4.1), what types of analysis are recommended? This is covered in Chapter 4 and, in particular, in tables 4.2 to 4.10.
- How can the system be deemed sufficiently safe for deployment? Section 3.5 contains guidance on this.

3.2 Levels of Assurance

In general, the level of assurance required of a complex system will be commensurate with the level of harm that the system could inflict on stakeholders and the environment. Note the definition of "harm" on page 6 as it relates to any stakeholder.

In this guidance, we consider five Levels of Assurance Addressing Complexity (LAAC) for a complex system:

- LAAC-0: No safety-affecting properties: unable to cause harm.
- LAAC-1: Unlikely to cause harm and, if harm occurred, it would be minor.
- LAAC-2: Able to cause minor harm to a few people or serious harm to a single person.
- LAAC-3: Able to cause significant harm to a few people or a single death.
- LAAC-4: Able to cause severe harm or multiple deaths.

Each of these must be considered over the anticipated deployment life of the system.

At the beginning of a development it is difficult to identify the required assurance level accurately. For example, the Horizon system deployed for the British Post Office was initially viewed by the software developers as an accounting system with no safety-affecting properties (LAAC-0). However, as described in reference [21], the larger system, incorporating the services provided by the software, harmed many people over a period of years: physically (including death), mentally and financially. Others were falsely imprisoned. In retrospect, the wider system incorporating the aspects mentioned above, would therefore have to be considered LAAC-4.

To estimate the LAAC appropriate for a system, it is necessary to:

1. identify the scope and boundaries of the system.

   It is a characteristic of a complex system that its boundary is difficult to define (see page 10) and it is normally found that, after an accident occurs, the boundary expands widely as trails are explored through the procedures and practices that led to the accident. A SCAD analysis as described in Appendix A.3.3 on page 51 could help to define a system's boundary *before* an accident occurs.

2. identify the stakeholders as described on page 7.
3. identify the types of harm as described on page 6 that can occur to each stakeholder.
4. identify expected and unexpected interaction.
5. characterise expected behaviours.
6. identify harm mitigations.
7. assign an appropriate LAAC.
8. review the assigned LAAC as needed.

3.3 Assurance Principles

The following assurance principles have been identified. In general, most of these could be considered appropriate for the assurance of any system, whether complex or not. The objectives given in tables 3.1 to 3.7 specialise these principles for complex systems.

The phrase "SHALL be considered" in the remainder of this chapter includes "SHALL be identified, analysed, assessed and managed".

1. P-1: Assurance Requirements SHALL be established for the SCS.

   As some systems are accidental and become complex incrementally (Scenario Types S-3 to S-8 in table 4.1), the assurance requirements would normally have to be defined retrospectively from the existing system: it is unlikely that requirements of this type were created before the SCS was designed and implemented (Scenario Types S-1 and S-2).

2. P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.

   The concept of a complex system having different levels of abstraction is described on page 7. Each abstraction may have a different associated vocabulary and different focus and it is important that the assurance requirements recognise this.

   Selecting the appropriate abstraction of the system is difficult because there is a human tendency to treat those parts of a system where the analyst is not an expert at an inappropriately high level of abstraction. This can lead to an inadequate analysis.

   Therefore, the analysis of the system should be done at a variety of levels of abstraction by a variety of players. This results in an analysis with various perspectives leveraging multiple areas of expertise. Reference [6] describes the importance of analysis considering different "units of adaptive behaviour" which refers to the need to draw system boundaries at different levels of abstraction as the perspective taken both reveals and obscures details.

   For this reason, it is important that the approval authority of the system includes analysts able to handle each part of the system at an appropriately low level of abstraction: see Section 3.5.

   See also the topic of further work in this area described in Section 6.2.5 on page 45.

3. P-3: Assurance requirements for the SCS SHALL be satisfied.

   Clearly, it is important that the system satisfy the requirements that have been defined and the method of satisfying a requirement has not introduced new risks. Where a requirement cannot be satisfied, then the residual risk must be identified and made clear to the appropriate stakeholders.

   As mentioned in the definition of "stakeholder" on page 7, the list of stakeholders is volatile and it is impossible to advise all of the residual risk. A person whose flight was cancelled and therefore who suffered harm as a result of the CrowdStrike incident was a stakeholder, but could not have been identified when the CrowdStrike system was being analysed.

   One of the common themes found in the examples of fatal accidents given in Appendix E.7 is that all assurance requirements had ostensibly been met and had been signed off by the appropriate authority. There were, however, serious flaws in the approvals process. See Section 3.5.

4. P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.

   One characteristic of a complex system is that it behaves in unanticipated ways: ways that were not considered during its design and implementation. It is important that the system be able to detect these conditions and be sufficiently resilient to maintain safe service. It is accepted that the resilience of the SCS will be incomplete, but the analysis should be performed as far as possible and be documented.

   Reference [6] describes the concept of "Graceful Extensibility" (a blend of "graceful degradation" and "software extensibility") as the opposite of brittleness: it is the ability of a system to extend its capacity to adapt when surprise events challenge its boundaries. The same reference makes the important point that "brittleness" is descriptive, not actionable: generally it is only after the system has failed that we realise that it was brittle.

5. P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.

For many complex systems, the precise boundary between the system and its external environment is impossible to define: it's porous and varies according to the abstraction at which it is being considered. Many complex systems are also long-lived, with their environment changing during their lives. This means that it is impossible to design for all eventualities of the environment; the system must be able to handle unanticipated conditions resiliently.

6. P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.

   The level of associated risk can be defined by the LAAC: see section 3.2. Note that the LAAC may need to be reässessed during the design and development of the system.

7. P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.

   Note that the life of complex systems may be long. A hospital is a complex system (see Section 2.3.1). St Bartholomew's Hospital in London has been in continued existence since 1123. It has been in its present buildings since the 18th century and the changes in medical procedures and devices since then have been significant.

Objectives are associated with each of these principles as listed in tables 3.1 to 3.7.

Id	Principle	Notes
1	**P-1: Assurance Requirements SHALL be established for the SCS.**	
	Objectives	
1.a	The environment within which the SCS is intended to be used SHALL be defined.	
1.b	The intended use (purpose) of the SCS SHALL be defined.	
1.c	The causes of complexity within the SCS SHALL be identified.	
1.d	The stakeholders of the SCS SHALL be identified.	This includes those developing the system and those affected by it after it is deployed, including operators and customers.
1.e	Safety Targets SHALL be captured as assurance requirements.	
1.f	Legal and regulatory obligations related to the SCS SHALL be treated as assurance requirements.	
1.g	Assurance requirements for required behaviours (including resilience) of the SCS SHALL be defined.	This includes integrity-, configuration- and security-related data requirements as these may affect system behaviour.
1.h	Assurance requirements to mitigate undesirable behaviours of the SCS SHALL be defined.	It is to be expected that, as emergent behaviour is unpredictable, mitigations will focus on maintaining the resilience of the system.
1.i	Assurance requirements on the SCS's states SHALL be defined (e.g. normal, abnormal and degraded), as well as requirements governing transitions between the states.	For example, within a hospital, the states might include normal operation, reaction to a local event with many casualties, emergency within the hospital itself, etc.
1.j	Approvals and sign-off authorities for assurance of the SCS SHALL be established.	This includes the identification of the people who will approve the SCS and also the conditions under which approval may be given.
1.k	Historical accidents and incidents related to the SCS SHOULD be assessed and any relevant recommendations considered.	

Table 3.1: Identifying the Assurance Requirements

Id	Principle	Notes
2	**P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.**	
	Objectives	
2.a	Levels of abstraction of the SCS SHALL be defined.	These are shown ideally in diagrammatic form and may be based on e.g. CONOPS, architecture, data flows, behaviours or services. There may be several levels of abstraction, all of which may present useful perspectives on the system.
2.b	SCS assurance requirements SHALL be identified for each level of abstraction.	
2.c	At each abstraction level, assurance requirements from higher levels of abstraction SHALL be satisfied.	The requirements may be transformed between abstraction levels.
2.d	The tracing of SCS assurance requirements through the abstraction levels SHALL be established.	
2.e	The SCS's interactions with other systems and services SHALL be defined.	The definition would normally be contained in an Interface Control Document (ICD)
2.f	Methods and techniques used to provide assurance within each abstraction level SHALL be identified.	
2.g	Methods and techniques used SHALL be justified.	
2.h	Assurance deficits SHALL be identified and managed, for example, through assurance wrappers.	Assurance wrappers are created to make good any known assurance shortfalls of subsystems or subservices: see the definition on page 5. Where wrappers are not provided or are insufficient, then restrictions and limitations MUST be devised and documented to mitigate the associated risks.
2.i	Approvals and sign-off authorities for assurance for each level of abstraction of the SCS SHALL be established.	

Table 3.2: Flowdown of the Assurance Requirements

Id	Principle	Notes
3	**P-3: Assurance requirements for the SCS SHALL be satisfied.**	
	Objectives	
3.a	Assurance artefacts SHALL be identified and produced.	This includes verification, analysis, simulation, etc. evidence.
3.b	Trusted tools SHALL be used to provide confidence in assurance activities.	See Appendix A for details of suitable tools.
3.c	Where possible, assurance requirements SHOULD be met through system design and associated design decisions.	For example, the design decision to incorporate a version of a software module or tool that has been pre-certified to an international safety standard rather than an uncertified version, increases the level of assurance of the SCS.
3.d	A justification explaining how the complex aspects of the SCS have been assured SHALL be produced.	
3.e	Assurance wrappers SHALL be implemented.	
3.f	Approvals and sign-offs SHALL be obtained.	The conditions for approval (see 1.j) are also checked to ensure that the approval is valid.
3.g	Assurance evidence SHOULD include evidence acquired from deployed prototypes and field usage where this is available.	

Table 3.3: Satisfying the Principles & Objectives

Id	Principle	Notes
4	**P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.**	
	Objectives	
4.a	Unintended behaviours arising from the SCS SHALL be considered.	
4.b	Unintended behaviours resulting from fault-free cases SHALL be considered.	That is, it shall not be assumed that a dangerous behaviour results only from component failure or malfunction. See reference [8] regarding SOTIF.
4.c	Emergent behaviours of the SCS SHALL be considered.	
4.d	Behaviours due to design decisions SHALL be considered.	Where the designers made a choice, the different possibilities considered shall be documented.
4.e	Resilience measures to deal with unknowns of the SCS SHALL be considered.	
4.f	SCS behaviours due to potential non-linearities SHALL be considered.	
4.g	SCS behaviours due to potential whole/part interactions SHALL be considered.	
4.h	SCS behaviours due to deliberate or accidental misuse SHALL be considered.	
4.i	Degraded and contingency modes of the SCS SHALL maintain a defined subset of these objectives to a specific level.	
4.j	Resilience to SCS behaviours SHALL be considered as part of sign-off/approval.	

Table 3.4: Managing Unintended & Unanticipated Behaviours

Id	Principle	Notes
5	**P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.**	
	Objectives	
5.a	Unintended behaviours of the SCS caused by the environment SHALL be considered.	
5.b	Interactions with other systems and services SHALL be considered.	
5.c	Unintended behaviours resulting from fault-free cases due to environmental causes SHALL be considered.	That is, it shall not be assumed that a dangerous behaviour results only from component failure or malfunction: the system shall be resilient to an unforeseen condition in its environment. See reference [8] regarding SOTIF.
5.d	Emergent behaviours due to the interaction of the SCS with its environment SHALL be considered.	
5.e	Behaviours of the SCS due to misuse within its environment SHALL be considered.	
5.f	Resilience to external factors SHALL be considered as part of sign-off/approval.	

Table 3.5: Handling Unintended & Unanticipated Situations

Id	Principle	Notes
6	**P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.**	
	Objectives	
6.a	An overall Level of Assurance Addressing Complexity (LAAC) SHALL be established, taking into consideration the reasonably foreseeable level of risk that the SCS presents to stakeholder groups.	See Section 3.2 for a description of LAACs. The LAAC assignment must be continually monitored as it might change when an incident occurs.
6.b	LAACs SHALL be developed at each level of abstraction of the SCS, consistent with the overall LAAC, tracing through the abstraction levels.	This considers both the desired behaviours and mitigations for the undesired behaviours.
6.c	Assurance artefacts SHALL be produced according to the LAAC and requirements.	
6.d	Activities, methods, analyses, and tools used to provide assurance SHALL be appropriate for the LAAC and stakeholder group.	
6.e	Specific confidence demands from stakeholder groups SHALL be addressed.	
6.f	Residual risks SHALL be documented.	
6.g	The rigour of approvals and sign-offs SHALL commensurate with the LAAC.	

Table 3.6: Obtaining Risk-Commensurate Confidence

Id	Principle	Notes
7	**P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.**	
	Objectives	
7.a	All changes to the SCS that affect these principles and objectives SHALL be assessed and managed.	
7.b	Assurance artefacts SHALL be maintained.	
7.c	Measures SHALL be in place to monitor the behaviour of the SCS at appropriate intervals throughout its life.	This includes the monitoring of defined SPIs (see definition on page 7) and other field reports.
7.d	Use of the SCS SHALL be monitored for change and an assurance impact analysis undertaken where necessary.	
7.e	Evolution and incremental change of the SCS SHALL be considered.	
7.f	Longevity and obsolescence of elements of the SCS SHALL be considered.	
7.g	All changes to the SCS SHALL be appropriately approved and signed off.	
7.h	Lessons learned from successful operation, failures and incidents SHOULD be reviewed.	

Table 3.7: Managing the System's Lifetime

3.4 Mapping Principles to LAACs

Tables 3.8 to 3.14 list the typical evidence provided for each of the objectives. This mapping will vary between types of complex system, but guidance is given in each table as to whether the objective is generally highly recommended (●) or simply recommended (○).

Id	Principle	Typical Containers	Needed for LAAC			
1	**P-1: Assurance Requirements SHALL be established for the SCS.**					
	Objectives		1	2	3	4
1.a	The environment within which the SCS is intended to be used SHALL be defined.	Concept of Operations, Scoping and requirements, etc.	●	●	●	●
1.b	The intended use (purpose) of the SCS SHALL be defined.	Concept of Operations, Contract, etc.	●	●	●	●
1.c	The causes of complexity within the SCS SHALL be identified.	Concept of Operations, Scoping and requirements, etc.	●	●	●	●
1.d	The stakeholders of the SCS SHALL be identified.	Concept of Operations, Contract, etc.	○	●	●	●
1.e	Safety Targets SHALL be captured as assurance requirements.	Requirements	○	●	●	●
1.f	Legal and regulatory obligations related to the SCS SHALL be treated as assurance requirements.	Requirements	●	●	●	●
1.g	Assurance requirements for required behaviours (including resilience) of the SCS SHALL be defined.	Requirements (particularly those derived from the HAZOP)	○	●	●	●
1.h	Assurance requirements to mitigate undesirable behaviours of the SCS SHALL be defined.	Requirements	○	●	●	●
1.i	Assurance requirements on the SCS's states SHALL be defined (e.g. normal, abnormal and degraded), as well as requirements governing transitions between the states.	Requirements	○	●	●	●
1.j	Approvals and sign-off authorities for assurance of the SCS SHALL be established.	Concept of Operations, Contract, etc.	○	●	●	●
1.k	Historical accidents and incidents related to the SCS SHOULD be assessed and any relevant recommendations considered.	Incident Log, Impact Analysis	○	●	●	●

Table 3.8: Principle 1: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
2	**P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.**					
	Objectives		1	2	3	4
2.a	Levels of abstraction of the SCS SHALL be defined.	Design and architecture	●	●	●	●
2.b	SCS assurance requirements SHALL be identified for each level of abstraction.	Requirements document	○	●	●	●
2.c	At each abstraction level, assurance requirements from higher levels of abstraction SHALL be satisfied.	Traceability and requirements documents	○	●	●	●
2.d	The tracing of SCS assurance requirements through the abstraction levels SHALL be established.	Traceability and requirements documents	○	○	●	●
2.e	The SCS's interactions with other systems and services SHALL be defined.	Interface Control Document	●	●	●	●
2.f	Methods and techniques used to provide assurance within each abstraction level SHALL be identified.	Assurance methods matrix	○	○	●	●
2.g	Methods and techniques used SHALL be justified.	Justification documents	○	○	●	●
2.h	Assurance deficits SHALL be identified and managed, for example, through assurance wrappers.	Wrappers report	○	●	●	●
2.i	Approvals and sign-off authorities for assurance for each level of abstraction of the SCS SHALL be established.	Traceability and requirements documents	○	●	●	●

Table 3.9: Principle 2: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
3	**P-3: Assurance requirements for the SCS SHALL be satisfied.**					
	Objectives		1	2	3	4
3.a	Assurance artefacts SHALL be identified and produced.	Verification and analysis reports; simulations; reviews; audit reports, etc.	●	●	●	●
3.b	Trusted tools SHALL be used to provide confidence in assurance activities.	Tool qualification reports	○	●	●	●
3.c	Where possible, assurance requirements SHOULD be met through system design and associated design decisions.	Design	○	●	●	●
3.d	A justification explaining how the complex aspects of the SCS have been assured SHALL be produced.	Complexity justification report	○	○	●	●
3.e	Assurance wrappers SHALL be implemented.	Wrapper design and implementation report	○	●	●	●
3.f	Approvals and sign-offs SHALL be obtained.	Service history; field reports; bug and incident reports	○	○	●	●
3.g	Assurance evidence SHOULD include evidence acquired from deployed prototypes and field usage where this is available.		○	○	●	●

Table 3.10: Principle 3: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
4	**P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.**					
	Objectives		1	2	3	4
4.a	Unintended behaviours arising from the SCS SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
4.b	Unintended behaviours resulting from fault-free cases SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	○	●	●
4.c	Emergent behaviours of the SCS SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
4.d	Behaviours due to design decisions SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	●	●	●
4.e	Resilience measures to deal with unknowns of the SCS SHALL be considered.	HAZOP; analyses	●	●	●	●
4.f	SCS behaviours due to potential non-linearities SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	○	●	●
4.g	SCS behaviours due to potential whole/part interactions SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	○	●	●
4.h	SCS behaviours due to deliberate or accidental misuse SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
4.i	Degraded and contingency modes of the SCS SHALL maintain a defined subset of these objectives to a specific level.	Verification; analyses; simulations	○	●	●	●
4.j	Resilience to SCS behaviours SHALL be considered as part of sign-off/approval.	Completion certificate process; Assurance Case	○	●	●	●

Table 3.11: Principle 4: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
5	**P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.**					
	Objectives		1	2	3	4
5.a	Unintended behaviours of the SCS caused by the environment SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
5.b	Interactions with other systems and services SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	●	●	●
5.c	Unintended behaviours resulting from fault-free cases due to environmental causes SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	○	○	●	●
5.d	Emergent behaviours due to the interaction of the SCS with its environment SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
5.e	Behaviours of the SCS due to misuse within its environment SHALL be considered.	HAZOP; analyses; mode definitions; mode transitions	○	●	●	●
5.f	Resilience to external factors SHALL be considered as part of sign-off/approval.	HAZOP; analyses; mode definitions; mode transitions	○	●	●	●

Table 3.12: Principle 5: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
6	**P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.**					
	Objectives		1	2	3	4
6.a	An overall Level of Assurance Addressing Complexity (LAAC) SHALL be established, taking into consideration the reasonably foreseeable level of risk that the SCS presents to stakeholder groups.	HAZOP; risk register; LAAC report; hazard log	●	●	●	●
6.b	LAACs SHALL be developed at each level of abstraction of the SCS, consistent with the overall LAAC, tracing through the abstraction levels.	LAAC report; design documentation; traceability matrix	○	●	●	●
6.c	Assurance artefacts SHALL be produced according to the LAAC and requirements.	LAAC report; HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
6.d	Activities, methods, analyses, and tools used to provide assurance SHALL be appropriate for the LAAC and stakeholder group.	LAAC report; HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
6.e	Specific confidence demands from stakeholder groups SHALL be addressed.	Risk register; hazard log	○	●	●	●
6.f	Residual risks SHALL be documented.	Risk register; hazard log	○	●	●	●
6.g	The rigour of approvals and sign-offs SHALL commensurate with the LAAC.	Risk register; hazard log	●	●	●	●

Table 3.13: Principle 6: Assurance Evidence

Id	Principle	Typical Containers	Needed for LAAC			
7	**P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.**					
	Objectives		1	2	3	4
7.a	All changes to the SCS that affect these principles and objectives SHALL be assessed and managed.	Change impact analysis; contract changes; roll-out Plans	●	●	●	●
7.b	Assurance artefacts SHALL be maintained.	HAZOP; analyses; simulations; risk register; reviews; audits	○	●	●	●
7.c	Measures SHALL be in place to monitor the behaviour of the SCS at appropriate intervals throughout its life.	Reviews; independent audits	○	●	●	●
7.d	Use of the SCS SHALL be monitored for change and an assurance impact analysis undertaken where necessary.	Change impact analysis; contract changes	●	●	●	●
7.e	Evolution and incremental change of the SCS SHALL be considered.	Change impact analysis; contract changes	●	●	●	●
7.f	Longevity and obsolescence of elements of the SCS SHALL be considered.	Obsolescence analysis	○	●	●	●
7.g	All changes to the SCS SHALL be appropriately approved and signed off.	Incidence and accident reports	○	●	●	●
7.h	Lessons learned from successful operation, failures and incidents SHOULD be reviewed.	SPIs, incidence and accident reports	○	●	●	●

Table 3.14: Principle 7: Assurance Evidence

3.5 Determining Fitness for Deployment

Traditionally, the indication that a developed system is ready for deployment has been an approval signature from the appointed responsible person. It is noted in Appendix E.7 that, in every one of the fatal incidents listed in Appendix E, "proper" sign-off by the approved responsible person had been achieved, but that all of those sign-offs were flawed.

Some of the characteristics of complex systems (Section 2.2), particularly emergent behaviour, non-linearity and indeterminate boundaries, make it very difficult to make a reasoned decision that a complex system is sufficiently safe for deployment. In particular, it is assumed that the approval authority for a complex system will be a team of people with expertise in different parts of the system at different levels of abstraction. The mechanism of providing the approval must also be part of the system development: there must be no time pressure on the approver (as there was, for example, in the case of the Boeing 737 Max 8 (Appendix E.4) and the R.101 (Appendix E.5)).

In particular there may be changes in the nature of the deployment responsibility at different stages of the system's life-cycle: determining suitability for initial deployment might be different from determining suitability for continued operation once field data are available. It must be expected that, after deployment, the human element of a socio-technical system will adapt the system from within to simplify or reduce work or to avoid saturation.

Chapter 4

Scenarios

> Matt Blaze
>
> "As we build systems that are more and more complex, we make more and more subtle but very high-impact mistakes. As we use computers for more things and as we build more complex systems, this problem of unreliability and insecurity is actually getting worse, with no real sign of abating anytime soon."

Contents

4.1 Complex System Scenarios 30
4.2 Mapping of Scenarios to Principles and Objectives 32

4.1 Complex System Scenarios

This section identifies several circumstances (called "Scenarios") under which it would be useful to recognise a system as complex.

The levels of necessary assurance given in Chapter 3 depend on the Scenario Type for which they are being defined. These Scenario Types are also used in subsequent chapters to indicate the relevance of different tools and techniques. Table 4.1 describes each Scenario. In reality, it is unlikely that any project would exactly match one of the listed Scenarios: there would normally be overlap between the Scenarios.

Note that, for each Scenario, analysis may need to include consideration of security (as well as safety) aspects.

Table 4.1: Scenarios

Scenario	Description
S-0: Check	A complex system is to be checked for new dependencies with other systems. This could be a scheduled (regular) check to see whether there are new dependencies or an *ad hoc* check when new dependencies are known to have arisen.
S-1: Build New	A complex system is to be designed and implemented from scratch, using few existing subsystems.
S-2: Assemble	A new complex system is to be implemented, primarily incorporating pre-existing subsystems, each with a documented history. "Glue" may be needed to complete the new system. It is possible that none of the subsystems being incorporated has previously been considered safety-affecting, but that the system as a whole is so considered. The subsystems may have been built to requirements that conflict with the requirements of the other subsystems.
S-3: Restructure	An existing complex system is to undergo major restructuring to integrate new, or remove existing, subsystems and to upgrade it to modern standards. This restructuring has not resulted from a safety-related incident, but has been found necessary during the updating of the system. Safety-analysis documentation for the entire system before restructuring may exist, but probably does not, as the system may have grown organically. Design documentation may exist for some of the subsystems. The system described in Appendix E.5 is an example of how this type of restructuring, needed as a result of a design error, can go badly wrong.
S-4: Evolve	An existing system evolves as technology changes and more effective components become available. The fundamental aspects of the system, including its purpose, remain unchanged, but new components result in new system behaviour. Evolution is natural in any system: new circumstances arise that have to be accommodated and, eventually these may completely replace the circumstances for which the system was originally designed. Appendix C.1 gives an example of how an existing complex system may need to evolve.
S-5: Assure	An assurance position must be created for an existing complex system to mitigate potential safety concerns. No serious incident has occurred to date, but impending deployment, change of use, change to the way problems are handled, changing legislation or increased public concern has made such an analysis necessary. This also occurs when a system has been incrementally extended or combined with other systems until an assurance engineer (e.g. while carrying out a review or audit) recognises the system now has the characteristics of a complex system.
S-6: Use New	An existing complex system is to be deployed in a new context or jurisdiction. This is similar to Scenario Type 5, but with a different driver.
S-7: Degrade	As a result of a failure, a system is running in degraded mode and this level of operation is likely to persist for some time. A new safety analysis needs to be performed.
S-8: Investigate	A serious safety-affecting incident has occurred in a complex system. The cause of this needs to be understood and mitigations proposed to reduce the risk of such an incident happening again.

Scenario	Notes
S-0: Check: A complex system is to be checked for new dependencies with other systems.	It is assumed that assurance requirements exist for the system and that the new dependencies are being considered in that context.
Possible Claim	
The new dependencies do not affect the system meeting its safety constraints.	
Highly-Recommended Principles	
P-3: Assurance requirements for the SCS SHALL be satisfied.	This is the primary focus of the analysis: are the assurance requirements still met?
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	Do the new dependencies alter the boundary of the system?
Recommended Principles	
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	The new dependencies may alter the system's levels of abstraction.
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	Could the new dependencies give rise to new, unanticipated behaviours?

Table 4.2: Scenario "S-0: Check"

4.2 Mapping of Scenarios to Principles and Objectives

In selecting the Principles and Objectives, the claim being made must be taken into account. Typically, claims are of one of three types:

1. The process used to create or analyse the system meets the requirements of a particular international standard.
2. The system, subject to assumptions made about its use, is adequately safe.
3. All appropriate safety engineering tasks have been carried out to ensure that the system meets its safety constraints.

Scenario	**Notes**
S-1: Build New: A complex system is to be designed and implemented from scratch, using few existing subsystems.	
Possible Claim	
All appropriate safety engineering tasks have been carried out during the design and implementation of this system to ensure that the system meets its safety constraints.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	It is accepted that these requirements will change during the design and implementation, but they must exist.
P-3: Assurance requirements for the SCS SHALL be satisfied.	If an assurance requirement cannot be satisfied, then this needs to be made explicit and the associated risks assessed.
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	The guidance in ISO 21448 (reference [8]) and UL 4600 (reference [7]) may be useful for this.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	Appropriate levels of abstraction should be made explicit in the system architecture.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	The system's environment should be captured as closely as possible in the system architecture. It is accepted that this cannot be complete and unanticipated environmental conditions may occur after shipment.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.3: Scenario "S-1: Build New"

Scenario	**Notes**
S-2: Assemble: A new complex system is to be implemented, primarily incorporating pre-existing subsystems, each with a documented history.	It is assumed that, although the pre-existing subsystems are adequately documented, they have not all previously been deployed in safety-critical systems. The new, composite system is to be developed for safety-critical use.
Possible Claim	
The system has sufficient justification in its parts, and in the way those parts are combined, to produce an adequately safe system.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	It may be necessary to prepare assurance requirements for some or all of the constituent subsystems as well as for the composite system. The requirements on the subsystems should state their demands on the way in which they are used in the new system. Any conflict between the functional or assurance requirements of the individual, existing subsystems must be identified and removed.
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	
P-3: Assurance requirements for the SCS SHALL be satisfied.	It would be useful to identify those requirements satisfied by individual or combinations of subsystems and those requirements satisfied only by the composite system.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	A simulation of the interactions between the existing subsystem might identify some of the unintended behaviour, but the “glue” being added will need to add resilience against unanticipated behaviour.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.4: Scenario “S-2: Assemble”

Scenario	**Notes**
S-3: Restructure: An existing complex system is to undergo major restructuring to integrate new, or remove existing, subsystems and to upgrade it to modern standards.	It is assumed that the existing system which is to be restructured was safety-critical, but that most of the safety assurance is out-of-date or unavailable, or will be invalidated by the restructuring.
Possible Claim	
All appropriate safety engineering tasks have been carried out during the restructuring of this system to ensure that the resulting system meets its safety constraints.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	These must accommodate the changes that are driving the restructuring.
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	The restructuring will probably change the abstraction layers and these will need to be redefined.
P-3: Assurance requirements for the SCS SHALL be satisfied.	If an assurance requirement cannot be satisfied, then this needs to be made explicit and the associated risks assessed.
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	The guidance in ISO 21448 (reference [8]) and UL 4600 (reference [7]) may be useful for this.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.5: Scenario “S-3: Restructure”

Scenario	**Notes**
S-4: Evolve: An existing system evolves as technology changes and more effective components become available. The fundamental aspects of the system, including its purpose, remain unchanged, but new components result in new system behaviour.	It is assumed that the existing system was adequately analysed for its original purpose and that its assurance requirements were met.
Possible Claim	
All appropriate safety engineering tasks have been carried out on the system to ensure that it will meet its safety constraints during and after the evolution.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	The existing assurance requirements must be reëxamined and modified as needed to cover the evolving technology and, in particular, the transition period.
P-3: Assurance requirements for the SCS SHALL be satisfied.	
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	The evolution may create additional unanticipated interactions between the traditional and new components.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	This needs to take into account the transition period during which both previous and new conditions will apply.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	It is assumed that the levels of abstraction are already defined and the evolution does not change these.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.6: Scenario “S-4: Evolve”

Scenario	**Notes**
S-5: Assure: An assurance position must be created for an existing complex system to mitigate potential safety concerns.	It is assumed that the existing system was adequately analysed for its original purpose and that its assurance requirements were met. The data concerning this analysis still exists, but changes to the environment in which the system is operating mean that new analysis is required.
Possible Claim	
All appropriate safety engineering tasks have been carried out on the system in its changed environment to ensure that the system meets its safety constraints.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	The existing assurance requirements must be reëxamined and modified as needed to meet the latest standards.
P-3: Assurance requirements for the SCS SHALL be satisfied.	This may require changes to the implementation of the system: if the new analysis finds potential defects, then these must be fixed.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	This needs to take into account the new environment within which the system is now operating.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	It is assumed that the levels of abstraction are already defined.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.7: Scenario “S-5: Assure”

Scenario	**Notes**
S-6: Use New: An existing complex system is to be deployed in a new context or jurisdiction.	It is assumed that the existing system was adequately analysed for its original purpose. The data concerning this analysis still exists, but changes to the context or jurisdiction in which the system is operating mean that new analysis is required.
Possible Claim	
All appropriate safety engineering tasks have been carried out on the system in its new context to ensure that the system now meets its safety constraints.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	The existing assurance requirements must be reëxamined and modified as needed to meet the latest standards.
P-3: Assurance requirements for the SCS SHALL be satisfied.	This may require changes to the implementation of the system: if the new analysis finds potential defects, then these must be fixed.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	This needs to take into account the new environment within which the system is now operating.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.	
Recommended Principles	
P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.	It is assumed that the levels of abstraction are already defined.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	

Table 4.8: Scenario “S-6: Use New”

Scenario	**Notes**
S-7: Degrade: As a result of a failure, a system is running in degraded mode and this level of operation is likely to persist for some time.	It is assumed that the necessary analysis was carried out on the system before deployment, but that because of the new form of operation (degraded mode), a new analysis is required. It was possibly assumed that the system would operate in degraded mode for a short time (during which few environmental changes would occur), but it has now been found that it will persist.
Possible Claim	
All appropriate safety engineering tasks have been carried out on the system to ensure that the system meets its degraded mode safety constraints (which may be different to normal operation and be time-limited).	
Highly-Recommended Principles	
P-3: Assurance requirements for the SCS SHALL be satisfied.	The primary object of the analysis is to determine whether or not the system in its degraded mode still satisfies its assurance requirements.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	
Recommended Principles	
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	If it was such a behaviour that caused the system to move to its degraded mode, then that behaviour needs to be identified and its implications analysed. This does not form part of the safety analysis for the degraded mode, but it is essential information for returning the system to its normal operating mode.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	If it was such a situation that caused the system to move to its degraded mode, then that situation needs to be identified and the means of avoiding it in future recorded. This does not form part of the safety analysis for the degraded mode, but it is essential information for returning the system to its normal operating mode.

Table 4.9: Scenario “S-7: Degrade”

Scenario	**Notes**
S-8: Investigate: A serious safety-affecting incident has occurred in a complex system.	It is assumed that the existing system has an appropriate level of design and implementation documentation and an Assurance Case that was considered adequate at the time of deployment. The purpose of this analysis is to identify the flaw in that Assurance Case and recommend changes that would reduce the chances of the incident occurring again to an acceptable level. This does not include implementing the recommended changes.
Possible Claim	
The error in the safety analysis performed on the system has been identified and recommendations on appropriate actions made.	
Highly-Recommended Principles	
P-1: Assurance Requirements SHALL be established for the SCS.	These will be a modification of the existing requirements to close the gap — in many cases, additional requirements regarding resilience will be needed.
P-3: Assurance requirements for the SCS SHALL be satisfied.	That is, a claim that, if the recommendations were implemented, the assurance requirements would be met.
P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.	In this situation (analysis following an incident), the level of risk posed by the system may need to be reässessed.
Recommended Principles	
P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.	It could be a lack of such resilience that can now be identified.
P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.	It could be a lack of such resilience that can now be identified.

Table 4.10: Scenario “S-8: Investigate”

Chapter 5

Through-Life Assurance of a Complex System

David Parnas

"Complexity is not a goal. I don't want to be remembered as an engineer of complex systems."

Contents

5.1 Introduction 41
5.2 Assurance Challenges 41
5.3 Recommendations 42
5.4 Case Study 42

5.1 Introduction

Conventionally, a Safety or Assurance Case is prepared to present the argument of whether or not a particular system is adequately safe for (continued) deployment. This chapter considers the preparation of an Assurance Case for a complex system and is incomplete; more exploration is needed on this topic and in this edition of the guidance, this chapter can be viewed as a placeholder.

5.2 Assurance Challenges

The preparation of an Assurance Case for any system presents challenges. In a complex system, these are more pronounced and include the following:

1. A complex system must be viewed at different levels of abstraction and each level has its own assurance criteria. In particular, no one person is likely to be able to produce or approve the assurance argument for all abstraction levels and one single, top-level claim is unlikely to be adequate.
2. In many of the Scenarios given in table 4.1 on page 31, particular Scenarios S-2 and S-3, different Assurance Cases may already exist for different parts of the system, possibly in different formats, and these will need to be combined.
3. The first step in preparing an Assurance Case is defining the system's boundary. However, the boundary of a complex system is difficult to define and changes dynamically during deployment.
4. Suitable analyses and verifications for a complex system must demonstrably support the safety argument — it's difficult know when the analyses sufficiently cover the complexity issues, particularly emergent behaviours.

5.3 Recommendations

It is probable that a modular approach to the Assurance Case is required for a complex system.[1]

More exploration is needed on this topic.

5.4 Case Study

This section considers the example of Smart Motorways in England: see Section 2.3.2 on page 15. The assurance activities that took place during the installation and operation of Smart Motorways included the following:

- A cautious assurance argument for going live with the complex system was prepared, based on available data and best endeavour assumptions of future outcomes. This was deliberately a cautious argument, in that it over-estimated factors in the argument that were not well-proven.
- Short term monitoring of selected features of the initial system (3 month, 6 month, 1 year) was instituted to check that the cautious estimates were valid for the assumptions (based on leading SPIs).
- Long term (3 year) monitoring of the initial system was instituted to validate the overall safety argument (based on lagging SPIs).
- Further systems, with moderate changes to the initial concept, were implemented (e.g. dynamic hard shoulder).
- Further systems, with significant changes to the previous concepts, were implemented (e.g. all lane running).
- Lessons learned from all the implementations of the systems were informally gathered.
- Long term (5 year) formal monitoring of all the implementations was instituted.
- Changes were made to existing implementations to reflect lessons learned (e.g. additional emergency areas).
- Sub-systems were upgraded to maintain system performance — cautious safety arguments were used and validated as far as possible before implementation (monitoring over 12, 6 and 3 months of leading SPIs as these upgrades were implemented over 6 regions).
- The safety argument was continuously reviewed based on data available (mainly lagging SPIs, but some leading)

[1] The SCSC Assurance Case Guidance (SCSC-159) contains details on the preparation of a Modular Assurance Case (section 2:4).

Chapter 6

Further Work

> Irving John Good
>
> "When discussing complex systems like brains and other societies, it is easy to oversimplify: I call this Occam's lobotomy."

Contents

6.1 Questions Regarding the Scope of this Guidance . . . 43
6.1.1 Malicious and Inappropriate Actions . . . 43
6.1.2 Responsible Authority . . . 44
6.1.3 Artificial Intelligence . . . 44
6.2 Research Topics . . . 44
6.2.1 Preparation of the Assurance Argument . . . 44
6.2.2 Risk-Aware Systems . . . 44
6.2.3 Forming the System View before Deployment . . . 44
6.2.4 The Relevance of Ashby's Law . . . 45
6.2.5 The Psychological Approach to Safety . . . 45
6.2.6 Combining FRAM/STPA with (Semi-)Formal Model Checking . . . 45
6.2.7 AcciMaps . . . 45
6.2.8 Other Analysis Techniques . . . 45
6.2.9 Resilience Techniques . . . 45

This version of the guidance represents work-in-progress. Several topics have been identified that require further investigation and these open questions are included here. Input is invited from any reader.

6.1 Questions Regarding the Scope of this Guidance

6.1.1 Malicious and Inappropriate Actions

Section 1.4 on page 5 defines the scope of the guidance in this document. Should this be extended to include malicious and inappropriate actions?

Malicious actions could include a person from within the development or operation of a complex system performing an unauthorised act or an external hacker breaking into the system.

Inappropriate, but non-malicious actions can have the same consequences as a malicious act. For example, managers putting pressure on company engineers to complete a project to a defined timescale, leading to an unsafe product. Note that such inappropriate acts may be deliberate or unintentional.

Reference [22] by Jens Rasmussen contains the observation that:

Court reports from several accidents such as Bhopal, Flixborough, Zeebrugge and Chernobyl demonstrate that they have not been caused by a coincidence of independent failures and human errors, but by a systematic migration of organisational behaviour toward accident under the influence of pressure toward cost-effectiveness in an aggressive, competitive environment. ... This leads to a natural migration of activities toward the boundary of acceptable performance.

6.1.2 Responsible Authority

Section 3.5 on page 29 discusses the practicality of an independent team taking responsibility for assessing and approving the safety assurance of a complex system and raises the questions of what should support that approval and how the approval can be kept relevant during the system's lifetime. Much more work is required to answer some of the questions in that section.

Is it practical to include in the approval authority sufficient people with the appropriate skills to ensure that all parts of the system are approved, with adequate understanding at a sufficiently low abstraction level?

6.1.3 Artificial Intelligence

Most autonomous, machine-learning systems are complex. Possible techniques for managing the safety of such systems are recommended in standards such as UL 4600 and ISO/PAS 8800. How far should this guidance document provide guidance on such systems?

6.2 Research Topics

6.2.1 Preparation of the Assurance Argument

Chapter 5 has been introduced as a place-holder in this edition of the guidance, covering how the Assurance Argument for a complex system should be developed. More exploration is needed in this area.

6.2.2 Risk-Aware Systems

Appendix A.3.5 on page 53 and references [23] and [24] refer to the concept of a "risk-aware system": a system that is aware of the risks it is creating and, by means of introspection, tries to control those risks.

The concept of a risk-aware system seems to resemble a system with a dynamic Safety Case that can be modified and recalculated in real-time when the system meets a situation for which it was not designed: see reference [25].

Are the characteristics of risk-aware systems relevant to complex systems?

6.2.3 Forming the System View before Deployment

A post-incident report typically contains far more information than analyses performed before the incident occurred. In particular, the boundary of the system often expands significantly during a post-incident analysis.

Would it be possible to carry out an analysis with the depth of the post-incident analysis **before** the system is deployed? Or are there too many potential safety-affecting outcomes to make such an analysis possible?

Note that, in the report on the Clapham Junction railway crash (reference [26]), the author writes: *"In my review I have attempted at all times to remind myself of the dangers of using the powerful beam of hindsight to illuminate the situations revealed in the evidence. The power of that beam has its disadvantages. Hindsight also possesses a lens which can distort and can therefore present a misleading picture: it has to be avoided if fairness and accuracy of judgement is to be sought."*

The question here is whether that powerful beam of hindsight can be applied *before* the incident as foresight.

In particular, as illustrated in Appendix E.7 on page 75, in each of the accidents considered in Appendix E assurance requirements were in place and satisfied. However, they were inadequate. Could imagined "hindsight" before the accident have identified those inadequacies?

6.2.4 The Relevance of Ashby's Law

Reference [27] quotes Ashby's Law of Requisite Variety that refers to a system which is subject to a disturbance or input, which applies a process of regulation, and gives an outcome. The law postulates that, to obtain a desired outcome, the system must match the number of states in the input with at least an equivalent number of states in its internal regulatory process. That is, to attain the desired output, the system must adapt its regulatory process to its environmental input.

This has been applied to many complex systems (e.g. reference [28]), particularly those of organisation within large companies. Does this law also apply to the analysis of engineered technical systems? While it cannot demonstrate that a design will meet the safety requirements, can it show that that design could not possibly meet the safety requirements?

6.2.5 The Psychological Approach to Safety

Reference [29] asks whether "safety" is a psychological instinct or something that we can define scientifically. It argues that safety is an emergent state resulting from our brain's efforts to minimise surprise and prediction error. Does this explain how we can form the correct abstraction level when analysing a complex system? An analyst will tend to build a more abstract model for parts of the system that she does not understand: how can this inherent bias be avoided?

6.2.6 Combining FRAM/STPA with (Semi-)Formal Model Checking

Appendix A.3.8 on page 55 outlines some work that has been done to combine the Functional Resonance Analysis Method (FRAM) with other techniques. It appears that additional results could be obtained by extending these combinations, particularly Systems-Theoretic Process Analysis (STPA) or FRAM, with a modelling tool.

6.2.7 AcciMaps

Most of the literature dealing with AcciMaps discusses investigating accidents that have already happened, rather than analysing systems before an accident occurs. Given AcciMap's reliance on causality and the emergent (non-causal) behaviour of complex systems, can AcciMaps be applied to pre-incident analyses of complex systems?

6.2.8 Other Analysis Techniques

Numerous techniques that have been proposed for analysing complex systems have not been included in Appendix A. These include Anticipatory Failure Determination (AFD), DynaFRAM, Resilience Assessment Grid (RAG), Dynamic Bayesian Network (DBN), and Event-B (as in the Rodin tool).

IEC 62740 (reference [30]) adds Barrier analysis, James Reason's Swiss cheese model, Events and Causal Factors (ECF) charting, Multilinear Events Sequencing (MES), Sequentially Timed Events Plotting (STEP), Causes Tree Method (CTM), Why-Because Analysis (WBA) and Management Oversight and Risk Tree (MORT).

Most of these techniques are "backward-looking", applied after a safety incident has occurred, to identify the cause. Apart from one scenario, this guidance document is "forward-looking": how can we design and build a system to avoid as many potential accidents as possible? However, it is accepted that many of the forward-looking processes arise as a result of an incident, particularly in the aviation industry.

Which of these should be included in future editions of this guidance?

6.2.9 Resilience Techniques

Because unexpected conditions are likely to occur in a complex system's environment, it is essential that the system be resilient to these. What techniques are suitable for providing resilience in a complex system?

In particular, the telecommunications industry has a history of providing resilience in its networks, but the techniques it uses (redundancy, replication, etc.) are unlikely to be appropriate for complex systems.

Appendix A

Potential Analysis Techniques

Jamais Cascio

It's a pretty widely-accepted notion that the atmosphere is a ridiculously complex system, and the best we can do with our models is a rough approximation.

Contents

A.1 Hazard & Risk Identification . . . 47
A.1.1 Hazard and Operability Analysis (HAZOP) . . . 47
A.1.2 Ontological Hazard Analysis (OHA) . . . 47
A.2 Assurance Requirements Capture . . . 48
A.2.1 Formal Requirements Elicitation Tool (FRET) . . . 48
A.3 System Analysis Techniques . . . 48
A.3.1 Functional Resonance Analysis Method (FRAM) . . . 49
A.3.2 System-Theoretic Accident Model and Processes (STAMP) . . . 50
A.3.3 Systemic Contributors Analysis and Diagram (SCAD) . . . 51
A.3.4 Discrete-Event Simulation (DES) . . . 52
A.3.5 Formal Analysis . . . 53
A.3.6 Safety Through Organizational Learning (SOL) . . . 54
A.3.7 Work Models that Compute (WMC) . . . 55
A.3.8 Combining STAMP or FRAM with Other Methodologies . . . 55
A.3.9 Analytic Hierarchy Process (AHP) . . . 56
A.3.10 Other Techniques . . . 56
A.4 System Resilience Techniques . . . 56
A.4.1 Operational Models of Infrastructure Resilience . . . 56

This chapter describes various techniques and associated tools applicable to the assurance analyses identified in Chapter 3. In particular contexts, these techniques avoid the problems associated with the traditional techniques described in Appendix D, but, for most complex systems, no single technique described in this guidance will be adequate.

It is not the purpose of this document to give direct guidance on design and verification techniques, only to introduce such techniques as will provide evidence for the assurance of the resulting system.

This guidance avoids recommending particular commercial tools, although it does mention both commercial and open source tools.

A.1 Hazard & Risk Identification

A.1.1 Hazard and Operability Analysis (HAZOP)

IEC 61882:2016 describes a *"structured and systematic technique examining a defined system, with the objective of identifying risks associated with the operation and maintenance of the system..."*. The technique involves a multi-disciplinary team examining each part of the system, using guide words to stimulate the participants' imagination. IEC 61882 contains sample guide words, but also states that additional guide words applicable to the system in question should be used as long as they are identified ahead of the HAZOP session. The guide words in table A.1 may be relevant for complex systems and could be added to those suggested in IEC 61882.

unexpected	disproportionate	unintentional	inconsistent	legacy
hidden	interaction	independent	fragile	partial
many	unknown	diverse	overlapping	inadvertent

Table A.1: Possible Additional HAZOP Guide Words

The result of the HAZOP process is a structured list of (1) identified deviations from the anticipated system environment, (2) the possible causes and consequences of those deviations, (3) the actions needed to mitigate the consequences and (4) the name of the person with responsibility for completing the actions.

Note that STAMP/STPA as described in Appendix A.3.2 can also be considered a form of HAZOP.

A.1.2 Ontological Hazard Analysis (OHA)

An "ontology" is defined in the Oxford English Dictionary as "a set of concepts and categories in a subject area or domain that shows their properties and the relations between them".

While the IEC 61882 HAZOP process described above can generate useful lists of hazards and risks, it cannot positively answer the question, *"Have we thought of everything?"* Additionally, the output is in an informal (natural) language which makes it difficult to manipulate quantitatively and, in particular, makes it difficult for computer-aided hazard identification techniques to operate on field data which become available after the system is deployed.

To improve the HAZOP process, a number of ontologies for the hazard and risk domain have been proposed: for example, references [31], [32] and [33]. These ontologies can be used to focus the HAZOP process so that the identified causes of hazards become less generic (e.g. "coding error" or "human error") and more helpfully specific. Reference [33] proposes using the Probabilistic Web Ontology Language (PR-OWL) for modelling the knowledge about a system in a particular industry (see figure A.1) and generate a generic Bayesian Network for that industry. The generic model can then be used to generate a model specific to a particular system. This will produce quantitative results regarding hazards and risks. Reference [33] describes the application of this process to a number of safety-critical incidents and lists the open-source tools required for a particular analysis.

A suitable ontology, combined with a system description, can also help to automate the production of an initial hazard and risk analysis: see, for example, reference [34].

A formal approach to OHA is described in reference [35]. This gives a structure for applying formal methods (see Appendix A.3.5 below) to an ontologically-based hazard analysis. The technique is to define a hierarchy of abstractions of the system, starting with the most abstract and becoming increasingly concrete through refinement. At each level, a HAZOP is performed and a check is made to ensure that the identified hazards and risks are controlled. Reference [35] uses the TLA$^+$ tool (reference [36]) to provide the correctness proof of the resulting argument, but the Rodin tool (reference [37]), with its emphasis on abstraction refinement may be more appropriate.

An example of the OHA process being applied is given in reference [38].

The authors of reference [33] point out that, in a quantitative model of this sort, it is challenging to capture numerical parameters describing human error.

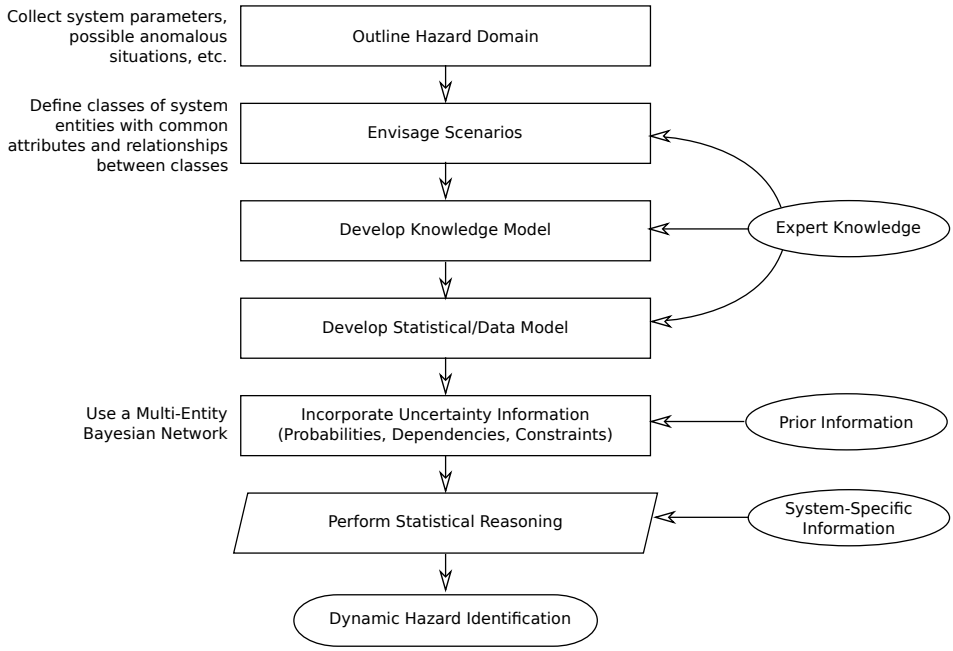


Figure A.1: Hazard Identification Methodology (Adapted from reference [33])

A.2 Assurance Requirements Capture

It was recognised as long ago as 1993 (reference [39]) that most errors in safety-critical embedded systems can be traced to inadequate requirements. Reference [40], considering software-related incidents up to 2023 in aerospace, indicates that incorrect or incomplete requirements are still a significant cause of dangerous incidents: 40% of the incidents considered in that paper relate to the absence of code and 16% to "unknown-unknowns", both indicating inadequate requirements.

Requirements written in a natural language may be incomplete, ambiguous or contradictory. Structured natural languages lie between the ambiguous natural languages and mathematical formal specifications such as Linear Temporal Logic (LTL).

A.2.1 Formal Requirements Elicitation Tool (FRET)

NASA's FRETish is a structured natural language for recording system requirements. The open-source tool `Fret` translates FRETish requirements into metric temporal logic formulæ that can be verified by various tools (e.g. SPIN, TLA$^+$).

Reference [41] describes the FRETish language and provides a mapping of a FRETish requirement into Metric Temporal Logic (MTL). Unfortunately FRETish is designed to handle functional requirements ("when X occurs then Y SHALL occur within 30 ms") rather than the non-functional requirements required for expressing assurance conditions.

A.3 System Analysis Techniques

The analysis techniques described in this section avoid the inadequacies listed in Appendix D and, in particular, address the challenges listed in Appendix D.3. Causal-flow techniques such as AcciMaps are excluded from this chapter because, as reference [42] states:

Describing accidents in a sequential fashion is arguably inadequate as it is unable to sufficiently explain the non-linear complexity of modern-day socio-technical system accidents. It can also guide analysts to search for the 'root cause' of an accident and stop an investigation when a suitable culprit is found to blame, [resulting] in too superficial an explanation to correctly inform the development of safety recommendations.

Reference [42] reports the results of reviewing 476 journal articles on accident investigation, searching for the techniques deployed. The results were that STAMP (Appendix A.3.2), FRAM (Appendix A.3.1) and AcciMaps accounted respectively for 52%, 20% and 18% of the 302 articles.

A.3.1 Functional Resonance Analysis Method (FRAM)

Detailed information about FRAM is available online at `https://functionalresonance.com/` and an introductory video is available at `https://www.youtube.com/watch?v=3ejf6vyDjcE`.

FRAM was introduced by Erik Hollnagel in reference [43] and is described in his book, reference [44]. It forms part of the "resilience engineering" approach to system safety that focuses on understanding how complex adaptive systems cope when encountering a "surprise".[1]

The basis of the method is the recognition that, in the past, we have improved safety by reacting to accidents: after an accident has occurred, we introduce new rules and make equipment changes to reduce the risk of the accident occurring again. This is not adequate for a complex system where the same set of conditions is unlikely to occur again.

FRAM can be applied both to pre-incident (Scenarios S-0 to S-7 in table 4.1) and post-incident (Scenario S-8) analysis. Reference [45] provides an example of the former, and [46] an example of the latter.

There are four steps in a FRAM analysis:

1. Identify Essential System Functions.

   For example, one function in a hospital might be "transfer patient from an ambulance to triage centre". For each identified system function, six questions must be answered:

   (a) What are the function's inputs: what initiates the function? In the example, the input would include a patient arriving in an ambulance.

   (b) What is the function's output: how does it transform the input? In this example, the output might be the patient in the triage area and the patient's condition entered into the hospital's computer system.

   (c) What preconditions have to exist before the function can begin? In this example, the computer system needs to be operating correctly, a stretcher must be available and there must be a free space in the triage area.

   (d) What control regulates how the function operates? In the example, the hospital has a written procedure describing how patients are admitted from an ambulance. This covers conditions such as no available room in the triage area.

   (e) Are there time constraints on the function? What happens if the function is delayed? In this example, delay might harm the patient further.

   (f) What resources are needed to carry out the function? In the example, the resources include trained personnel to admit the patient and enter the data into the computer system and a suitable stretcher.

2. Determine the possible variability of each function.

   How could the operation of the function vary, given existing or potential future changes? This is particularly pertinent if the function depends on human activity where, for example, at a change of shift an experienced operator might be replaced by one with little experience.

   Reference [47] gives examples of variability in technology, organisations and individuals and notes that

[1] See `https://en.wikipedia.org/wiki/Resilience_engineering`

human functions can vary due to physiological and psychological factors. ...Fatigue and stress (workload) are probably the most studied of the several physiological factors. Other factors are diurnal rhythm, well-being (or illness), various physiological needs, temporary disabilities, etc. There are also many different psychological factors that may affect the performance of a task, such as personality traits, cognitive style, bias in assessment and decision-making, etc.

3. Identify the functional resonances caused by the (wanted and unwanted) dependencies between the functions.

   As the operation of each function varies, how does this affect the other functions? For example, if function A's timing varies, how does this affect the variability of function B? In particular, if function A takes an abnormally short time, does this affect function B in such a way that a disproportionate system condition occurs?

4. Determine countermeasures.

   What protective or preventative barriers need to be introduced to dampen the effects of the variabilities of the functions? It is assumed that it is not possible to eliminate all variability, but the aim is to identify how harmful situations can arise in unanticipated ways.

FRAM has a graphical notation. An open-source tool, the FRAM Model Visualiser (FMV), is available to edit that notation.

In reference [48], David Slater contrasts FRAM with other modelling methods by pointing out that FRAM represents systems in terms of functional necessities, constraints and conditions rather than in terms of components, resources and state variables. This reference also positions FRAM as a potential candidate for a digital twin for complex systems, again stressing that FRAM acts at the level of functional behaviour rather than component state.

A.3.2 System-Theoretic Accident Model and Processes (STAMP)

The STAMP framework is described in Nancy Leveson's book, reference [49], which builds on work by Jens Rasmussen (reference [50]), including AcciMaps.

STAMP supports two methodologies:

1. Systems-Theoretic Process Analysis (STPA).

   This analyses the characteristics of a system before an incident has occurred (Scenarios S-1 to S-7 in table 4.1).

2. Causal Analysis Using System Theory (CAST).

   This is applied after an incident occurs (Scenario S-8).

In both methodologies, the STAMP approach diverges from event-based accident models, which explain accidents in terms of a chain of events sequenced over time, because it is difficult or impossible to incorporate non-linear relationships between events and feedback loops. Reference [51] also points out that, in many investigations, it is impossible to identify the "initiating event" that started the chain.

STAMP views a system as a hierarchy of controlled processes with actuators driven by a controller and sensors feeding data back to the controller. The controller may be a human, an organisation or a physical device and has a "mental model" of how the process it is controlling will react to different control inputs. The actuators and sensors may be physical devices or, more normally, abstract, as in Figure A.2: society puts pressure on the government to introduce legislation to control, for example, autonomous cars. The government has a (potentially incomplete) model of the autonomous car industry and itself controls various regulatory bodies. This hierarchy continues as the regulatory bodies put restrictions on the companies designing such cars, etc. Feedback occurs when, for example, a company complains that a regulation is inappropriate.

In analysing such a system there are particular points where hazards can arise. Using the example of a regulatory body (controlling process) interacting with a company (controlled process) producing autonomous cars:

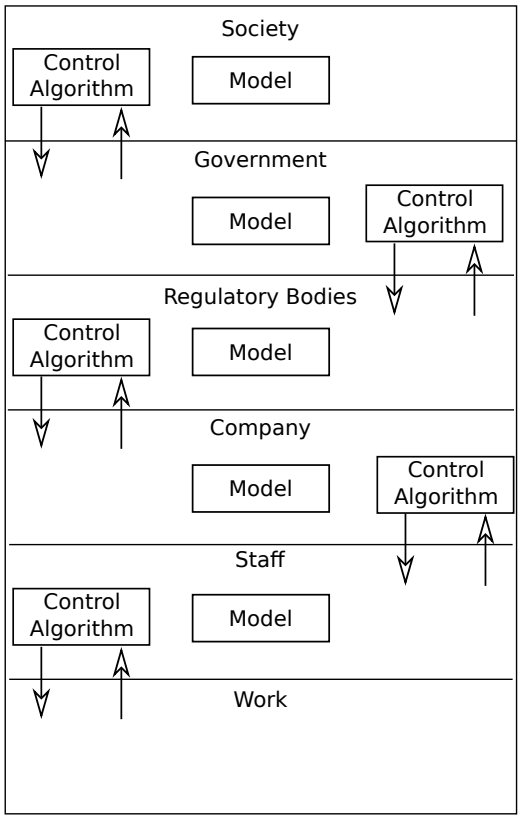


Figure A.2: An Example STAMP Framework

- the actuators may deliver incorrect or delayed data to the controlled process.

  Legislation regarding autonomous cars may be put into place more slowly than the technology is moving, making it obsolete before it is applied.

- the sensors may pass back incorrect or delayed data to the controller.

  The companies may pass back misleading statistics about accidents involving autonomous cars to the regulator.

- the controller's model of the controlled process may be incorrect.

  Because of lack of technical understanding by the regulator's employees, their mental model of the autonomous car's abilities may be wrong.

It is the last of these which is most subtle and most common: as the controlled object changes, the controller's model may become out of date or otherwise inadequate. This is often driven by emerging technology. For instance, smart watches now provide a distraction for car drivers that was not considered when the law banning the use of cell' 'phones when driving was issued. The model (in this case the law) has to be updated.

If an autonomous car causes harm, then this complete hierarchy, and the coördination between the layers, must be examined to reduce the chances of future harm.

A.3.3 Systemic Contributors Analysis and Diagram (SCAD)

Reference [52] compares a standard Root-Cause Analysis (RCA) with SCAD in the analysis of a near-fatal accident during a NASA space-walk in 2013 and states that: *"today's practice of root cause analysis is unable to come to grips with systemic issues, misses the interactions between contributors, and misses emergent system properties."*

When analysing an accident RCA, like many other techniques, starts with the accident (what SCAD terms the "sharp, proximal end") and works back to determine the cause. In contrast, SCAD starts by identifying the systemic vulnerabilities within the organisation (the "blunt, distal end") by considering the time, financial and other pressures under which decisions were made, the mental conflicts caused by the assumptions made (e.g. misunderstandings caused by mode confusion and confirmation bias) and the effort-saving or performance-enhancing adaptations made by the people involved. Taken together, these identify a drift towards failure.

Figure A.3 illustrates a SCAD diagram, the paths joining specific systemic weaknesses found in the company performing the work. These lead from distal, blunt weaknesses caused by the types of pressure mentioned above, particularly within a poor Safety Culture, through misunderstandings and other conflict, to the adaptations that operators have made and finally, to the proximal accident.

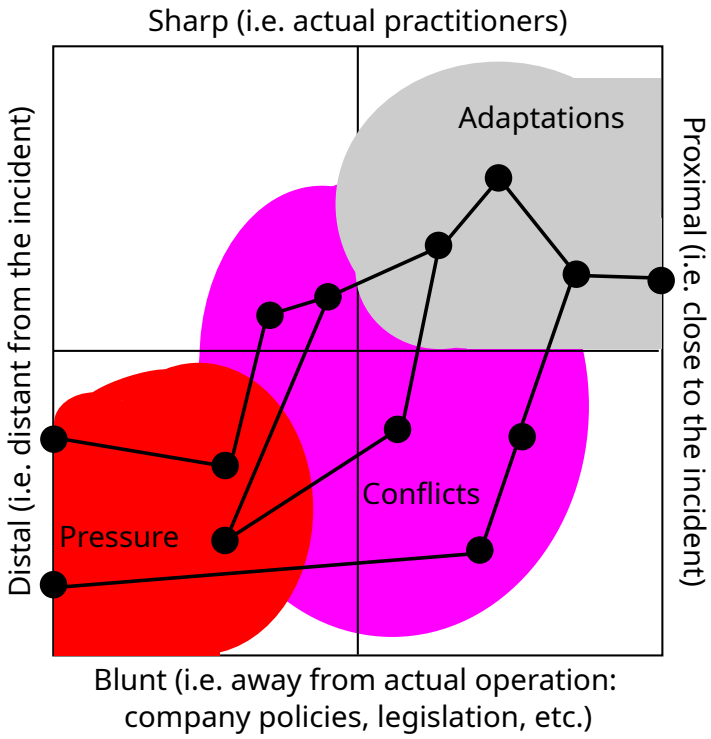


Figure A.3: A Sample SCAD Diagram

In particular, reference [52] states: *"The SCAD analysis goes beyond the label of human error to find out about the systemic factors that created the behavior in question. In the systemic view, human error is a label for normal behaviors that make sense for practitioners given the priorities and pressures they are operating under at the time."*

Much of the SCAD literature focuses on SCAD as a means of analysis after the accident has occurred. Reference [52], for example, analyses a NASA near-accident. However, its focus on starting in the Distal/Blunt quadrant implies that it could be applied when no accident has (yet) occurred. In particular, SCAD could possibly be used to monitor the organisation's safety culture to update the Safety Case dynamically as described in reference [25].

A.3.4 Discrete-Event Simulation (DES)

Discrete-event simulation (DES), also known as Monte-Carlo simulation, was one of the earliest applications of a digital computer: IBM released its General Purpose System Simulator (GPSS) in September 1961 and languages such as GASP and Simscript appeared before 1965.

Like FRAM, a DES of a system can often detect unexpected and unwanted behaviour before the system is deployed: a DES will often surprise the analyst.

DESs may be transaction- or event-focussed and, for system simulation, event-focus is more common. In this type of simulation two lists of events are kept:

1. the future-events queue: a list of events that will occur at some time in the future.
2. the current-events queue: a list of events that will occur when some condition is true (e.g. a queue is not full).

The DES algorithm is then very simple and easily programmed:

1. Take the next event from the future-events queue.
2. Set the clock to the time of that event — this is why it's called **discrete-**event simulation: the clock moves in discrete steps.
3. Process the event. This will normally mean scheduling more events for the future: for example, the event "component X fails" might lead to the event "component X repaired" at a random time between 1 and 5 hours in the future. These events are placed on the future-events queue.
4. Move any event on the current-events queue for which conditions are available to the future-events queue to be activated at the current time.
5. If there is any event on the future-events queue, return to step 1.

By defining the events that can occur in the system under analysis, a DES may uncover unacceptable behaviour. This is particularly so for systems where many activities are being carried out in parallel by different subsystems.

A DES uses (pseudo-)random numbers to explore the state space of the system being simulated—these provide the times of events to be scheduled in the future. Its results are therefore statistical rather than providing a proof of correctness, and confidence intervals can be generated using techniques as described in reference [53].

Some work has been carried out to use genetic algorithms to encourage the simulation to explore edge and corner cases.

A DES can also be an excellent source of testcases. By its nature, a DES produces a voluminous trace of the states that it has traversed and the trace can be analysed to find different paths, each of which can become a testcase.

References [54] and [18] describe examples of DES being used to perform risk assessments on runway incursions at large airports.

A.3.5 Formal Analysis

Like discrete-event simulation, formal methods have a long history: dating back at least to the Vienna Development Method (VDM) originating from IBM's Vienna laboratory in the 1970s. Also like DES, formal methods often surprise the analyst ("Oh yes, I hadn't thought of that!") and can therefore be used to predict unintended system behaviour.

Reference [55] describes an example of exploring a formal approach in the PICASSOS project: a UK government funded programme to improve the ability of automotive supply chains to develop complex software-intensive systems with high safety assurance and at an acceptable cost. A consortium of three universities and five companies collaborated on this programme. Formal methods-based analysis tools were incorporated in a model-based systems engineering workflow, and involved SysML and Simulink/Stateflow models, supported by other prototype and state-of-the-art COTS tools. Several trials were undertaken comparing two processes, one with formal methods and one without, during simulated development of electric vehicle-based systems. The results of one of the trials showed that the formal methods-based approach found errors that were missed by a standard model-verification process at software unit level.

In order to make use of formal methods, the analyst:

- describes the system using a mathematical language.

- defines invariants — system characteristics that must always be true. These can be of several types:
  - Safety properties: informally, these say that the system never does anything bad. For example, "If the north/south traffic light is green, the east/west traffic light is never green".
  - Liveness properties: informally, these say that the system always eventually does something good. For example, "Always, eventually, the north/south traffic light is green".

  Classical Aristotelian logic does not permit temporal terms such as "eventually" and "always" so a temporal logic is used for specifying the invariants.

- executes a tool to prove mathematically that, given the definition of the system, the invariants must always be true.

How the tool performs this final step varies from tool to tool. The SPIN tool (reference [56]) builds a Büchi Automaton to represent all systems satisfying the description given of the system and another Büchi Automaton representing all systems that do not conform to the invariants. It then needs to show that the intersection of these two Automata is empty. If it is not, then it displays the contents of the intersection showing how the system as described can violate one or more of the invariants.

The TLA$^+$ tool (reference [36]) builds a compressed, but exhaustive, state table of the described system using a technique that allows it to handle many millions of states. It then checks that, in each possible state, the invariants are true.

The Rodin tool (reference [37]) allows the analyst to describe the system hierarchically, starting with a completely abstract view of the system and building up until the view is completely concrete. At each step, Rodin deduces the mathematical theorems that need to be proven to ensure that the invariants are not violated. It then invokes a number of theorem provers to attempt to prove those theorems.

As can be seen from the dates on the references, these tools date from the first decade of the 21st century and are mature. The amount of mathematical awareness the analyst needs varies from tool to tool (with TLA$^+$ needing most and SPIN needing least).

Traditionally, these tools have been used to prove the correctness of designs of devices (detecting race conditions, deadlocks, livelocks, etc.), but can scale (particularly Rodin) to detecting invariance violations in complex systems.

Reference [23] describes the application of formal methods to risk-aware systems: systems that avoid risk by continuously judging risk in a situation (using introspection), estimating the current situation, and predicting future situations before taking appropriate action to minimise the risk.

Appendix A.3.8 below describes combining the use of formal methods with STAMP and FRAM.

A.3.6 Safety Through Organizational Learning (SOL)

Much of the material in this section is taken from reference [57] which itself references [58].

SOL is an event-analysis technique, which does not rely on the notion of cause/effect chain (which is often inappropriate for complex systems, particularly socio-technical ones).

The purpose of event-analyses is to use a model of the system to identify systemic weaknesses, possible improvements and recurrence preventions. A side benefit is a deeper understanding of the system.

Qualitative analysis is emphasised. Factors are classified into technology, individuals, working group, organisation, and organisational environment.

A SOL analysis starts by constructing a situational description through event building blocks that are arranged in a Time-Actor Diagram: a graph with time along the horizontal axis and the actors along the others. Factors that directly contribute to the system safety are classified into *information*, *communication*, *working conditions*, *personal performance*, *violations* and *technical components*. Factors that contribute indirectly are classified into twenty classes.

SOL is supported by a software tool, SOL-VE, which provides checklist-style forms for gathering information, displaying the event building blocks and displaying the Time-Actor Diagram. The checklists consist of a series of

questions which have been devised through the experience and research of the SOL developers, and the experience of the primary target industry: nuclear power.

Experience with SOL finds that it is very helpful at generating many more factors in the course of analysis than were conceived initially, avoiding "mono-causal thinking". The analysis generally broadens the focus on factors away from the actions of individual actors and towards more general systemic organisational and operational characteristics.

A.3.7 Work Models that Compute (WMC)

Work Models that Compute (WMC) is a simulation framework that was initially introduced and used in the Air Traffic Control domain in the early 2000s (reference [17]) specifically to simulate envisioned concepts of operation (reference [59]).

WMC is based on the concept of work interacting with the environment, driving and creating new work. The models simulated in WMC are not agent-specific but rather agent-agnostic, allowing for the testing of various configurations (reference [60]).

WMC is intended to be used as a discovery tool, rather than as a means to establish a normative path of work (reference [61]).

WMC has since been applied in various domains, including air traffic management (reference [62]), manned spaceflight operations (references [63] and [64]), and short-range, low-altitude air operations (references [61] and [65]), to investigate how different envisioned human-automation architectures impact the system.

References [59] and [61] provide insights into WMC for simulation, while reference [61] provides an overview of how to run a study using WMC. Using WMC for computational modelling and simulation involves a three-phase approach: converting paper-based representations of work into computational models, developing scenarios and test conditions, and simulating work dynamics to analyse emergent behaviors. The output from WMC is analysed by a three-lens model representing different ways of interpretation: to what degree do the envisioned operations achieve desired outcomes; how do the different versions compare with each other; and how can the robustness of these operations be enhanced? Interpreting the output using the three-lens framework along with the modelling assumptions helps fill under-specifications in the current human-AI system and gives designers data to direct future efforts (reference [61]).

A.3.8 Combining STAMP or FRAM with Other Methodologies

References [66] and [67] identify gaps between the expected and the obtained result when FRAM or STAMP was applied alone. Reference [68] lists a large number of analyses where STPA and FRAM were combined with other methodologies. The most popular combinations were:

FRAM with

- a formal model checker (e.g. SPIN: see Appendix A.3.5)
- a finite-state machine
- a discrete-event simulation (see Appendix A.3.4)

STPA with

- a formal model checker (e.g. Event-B (Rodin): see Appendix A.3.5)
- behaviour-driven development (BDD)

See Section 6.2.6 on page 45 for additional future work in this area.

A.3.9 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process, developed by Thomas Saaty (reference [69]), is a method to solve decision-making problems using a hierarchical structure of criteria and a given number of options. It can be used by automotive engineers, for example, to evaluate potential options for a complex vehicle system architecture.

It uses pairwise comparisons to compare each option one on one with each of the other options. It suggests that options are compared using a 1-9 numeric scale, where 1 means both options are equally important and 9 means one option is very much better than the other in terms of that criterion. The high-level steps can be described as:

- model the problem as a hierarchical structure by organising the components for decision making into the decision making goal, criteria involved and the options to compare
- compare the criteria pairwise
- calculate weights for the criteria
- evaluate options against the criteria to determine their scores
- combine weights and scores to rank options by multiplying each option's score on each criterion by the criterion's weight and summing the products to get the option's total score.

An example practical application is in its use by NASA (reference [70]) to narrow down candidate reactor designs for a hybrid nuclear electric propulsion (NEP) system for potential use in human exploration of Mars in the 2030s.

AHP has the potential to address the following fundamental complexity characteristics because the technique assists in ensuring that the system components are selected based on user needs and system level requirements:

- SF2, SF3 and SF4 from the system-level characteristics given in Section 2.2.1 on page 11.
- DF6 from the design and implementation characteristics given in Section 2.2.2 on page 12.
- CF1, CF2, CF3, CF4, CF5 and CF6 from the assurance characteristics given in Section 2.2.4 on page 13.

A.3.10 Other Techniques

See Section 6.2.8.

A.4 System Resilience Techniques

Because unexpected conditions are likely to occur in a complex system's environment, it is essential that the system be resilient to these. See Section 6.2.9.

During the operational life of a complex system, the human stakeholders within the system are likely to invent new behaviours either to reduce workload or to encompass new technologies. It is therefore essential that vigilance is built in to the operational procedures to detect and assess this new behaviour and that the system be resilient to these changes.

For example, in the Active Traffic Management system described in Section 2.3.2 and Appendix C.1, the recently enhanced characteristics of smart wrist watches has changed driver behaviour.

A.4.1 Operational Models of Infrastructure Resilience

Reference [71] introduces a quantitative modelling approach for analysing the resilience of an infrastructure system, such as electrical distribution. These are complex systems as they involve a mix of humans (e.g. owners, operators, managers) and autonomous "agents" (e.g. monitoring systems, feedback controllers) that make decisions to guide the behaviour of the system as a whole. These systems are subject to disruptions caused by nature (hurricanes), by accidental failures and by malicious actions.

The functional modelling approach proposed in reference [71] focuses on system operation in the face of disruptions to evaluate the consequences of the loss of an infrastructure component. Through these models, functional dependencies are considered to understand how the system operates when a component is no longer available.

Figure A.4 provides a visualization of an operational model of infrastructure resilience in a fuel supply system (this is the example used in reference [71], although the method applied is general). The circles represent an infrastructure component with the edges depicting their dependencies. A white circle represents a location with demand equal to one barrel of fuel. A black circle represents a location with supply equal to 10 barrels. Each link is bidirectional, has a fuel flow capacity of 15 barrels, and has per-barrel transit cost of $1. The penalty for unsatisfied demand per node (circle) is $10 per barrel. The network in figure A.4 has been built to be $N-1$ reliable, meaning that the loss of any single link does not disconnect any node.

When different disruptions arise in different channels, it is seen how the infrastructure operator can guide the system via alternate pathways to meet system goals. If these goals cannot be reached, the infrastructure resilience is insufficient.

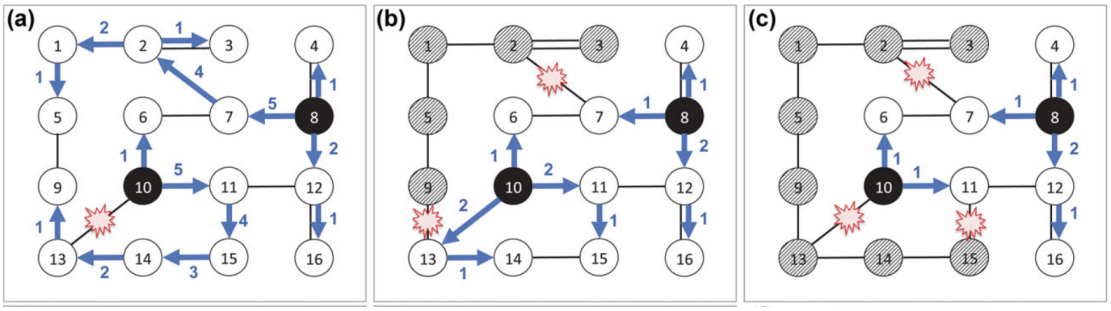


Figure A.4: Example of the Resilience Modelling Approach (From reference [71])

The results are graphed on a resilience curve. These curves depict the “best achievable worst-case performance” of the system given the magnitude of a disruption (Figure A.5). For example, a-c in Figure A.4 depict increasing magnitudes of disruption. This is plotted against the consequence of losing the components in question.

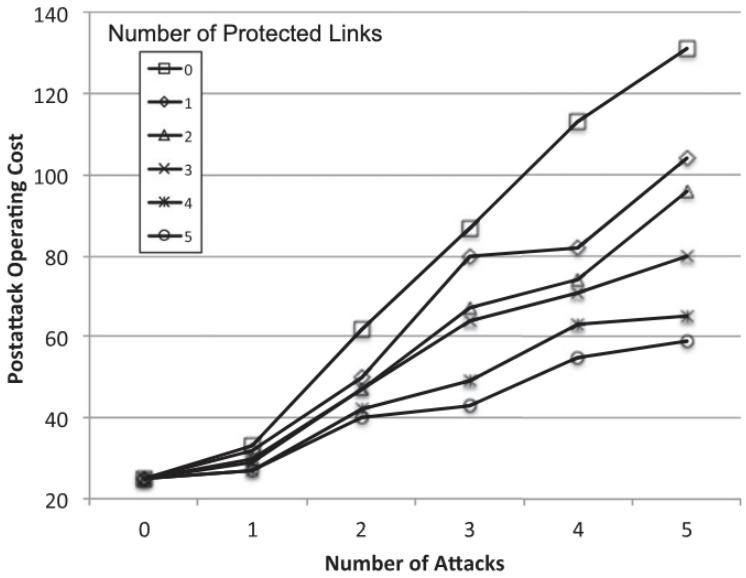


Figure A.5: Example of Resilience Curve (From reference [71])

Through this analysis, one can gain insights into operational resilience of the system, vulnerabilities to its functioning and future investments to improve operational resilience.

Figure A.5 illustrates that, without protection, the post-attack operating costs grow approximately linearly with the number of attacks. With each additional protection, this curve becomes less steep, indicating improved operational resilience for the system.

Appendix B

Analysing Complex Systems

Roger Penrose

"We have a closed circle of consistency here: the laws of physics produce complex systems, and these complex systems lead to consciousness, which then produces mathematics, which can then encode in a succinct and inspiring way the very underlying laws of physics that gave rise to it."

Contents

B.1 Analysis: Scenario S-1: Build New . . . 59
B.2 Analysis: Scenario S-2: Assemble . . . 59
B.3 Analysis: Scenario S-3: Restructure . . . 60
B.4 Analysis: Scenario S-4: Evolve . . . 60
B.5 Analysis: Scenario S-5: Assure . . . 60
B.6 Analysis: Scenario S-6: Use New . . . 60
B.7 Analysis: Scenario S-7: Degrade . . . 60
B.8 Analysis: Scenario S-8: Investigate . . . 60
B.9 Analysis: Scenario S-0: Check . . . 61

The analyses and techniques applicable to the development of a SCS depend on a number of factors, including the Scenario within which the development is occurring. This section provides guidance related to the different Scenarios described in table 4.1 on page 31. Figure B.1 provides an outline of the generic steps, with possible "entry points" for the different Scenarios.

B.1 Analysis: Scenario S-1: Build New

A complex system is to be designed and implemented from scratch, using few existing subsystems.

For a completely new design and implementation from scratch, the full process illustrated in figure B.1 would be followed, starting with the definition of the system and its environmental context.

B.2 Analysis: Scenario S-2: Assemble

A new complex system is to be implemented, primarily incorporating pre-existing subsystems, each with a documented history.

This Scenario is very similar to S-1: Build New in that, as a safety-affecting system it is being considered from scratch — in both cases components may already exist, but they were not developed in accordance with a safety standard. The new system and its environment need to be defined.

B.3 Analysis: Scenario S-3: Restructure

An existing complex system is to undergo major restructuring to integrate new, or remove existing, subsystems and to upgrade it to modern standards.

The entry point into Figure B.1 will depend on the existing safety assurance documentation for the subsystems that are to be restructured. In the case where this documentation does not exist, or where it is inappropriate for the new system, the entry point will be effectively the same as for S-2: Assemble.

B.4 Analysis: Scenario S-4: Evolve

An existing system evolves as technology changes and more effective components become available. The fundamental aspects of the system, including its purpose, remain unchanged, but new components result in new system behaviour.

The entry point in figure B.1 for this Scenario is that the objectives need to be tailored, leading to changes in the system's design, assurance argument and approval criteria. Appendix C.1 provides an example of handling this type of Scenario.

B.5 Analysis: Scenario S-5: Assure

An assurance position must be created for an existing complex system to mitigate potential safety concerns.

For this Scenario, the entry point into Figure B.1 is to assess the existing system's resilience based on the Safety Performance Indicators (SPIs) and other reports from the field (Ⓑ). This assessment may lead to the need to redesign parts of the system (Ⓔ) and make changes to the assurance argument.

B.6 Analysis: Scenario S-6: Use New

An existing complex system is to be deployed in a new context or jurisdiction.

The entry point into Figure B.1 is similar to that of Scenario "S-5: Assure", but path Ⓒ is almost certain to be taken as the original objectives will have changed.

B.7 Analysis: Scenario S-7: Degrade

As a result of a failure, a system is running in degraded mode and this level of operation is likely to persist for some time.

The primary activity if a system is running in degraded mode is to monitor the system to assess whether the SPIs are degrading. This will lead to the reconsideration of the Assurance Argument (Ⓐ in Figure B.1).

B.8 Analysis: Scenario S-8: Investigate

A serious safety-affecting incident has occurred in a complex system.

Many of the techniques listed in Appendix A, in particular FRAM, CAST and SCAD are suitable for analysing the cause of an accident. Reference [72] by Nippin Anand provides an argument, using the investigation into the

grounding of the Costa Concordia as an example, that these techniques would have to be applied differently to complex systems.

B.9 Analysis: Scenario S-0: Check

A complex system is to be checked for new dependencies with other systems.

As described in table 4.1, this Scenario describes a check to see whether new dependencies have arisen. The check is not being performed as a result of an incident, but probably as part of a scheduled process.

The entry point for this scenario is given in figure B.1 as an analysis of the monitored data collected from the field, particularly the performance of the system against the SPIs, to see whether sufficient data are being collected. The new dependencies may lead to feedbacks Ⓖ and Ⓑ to reconsider the system's behaviour and to define additional SPIs. These may then lead to a reworking of the Assurance Argument.

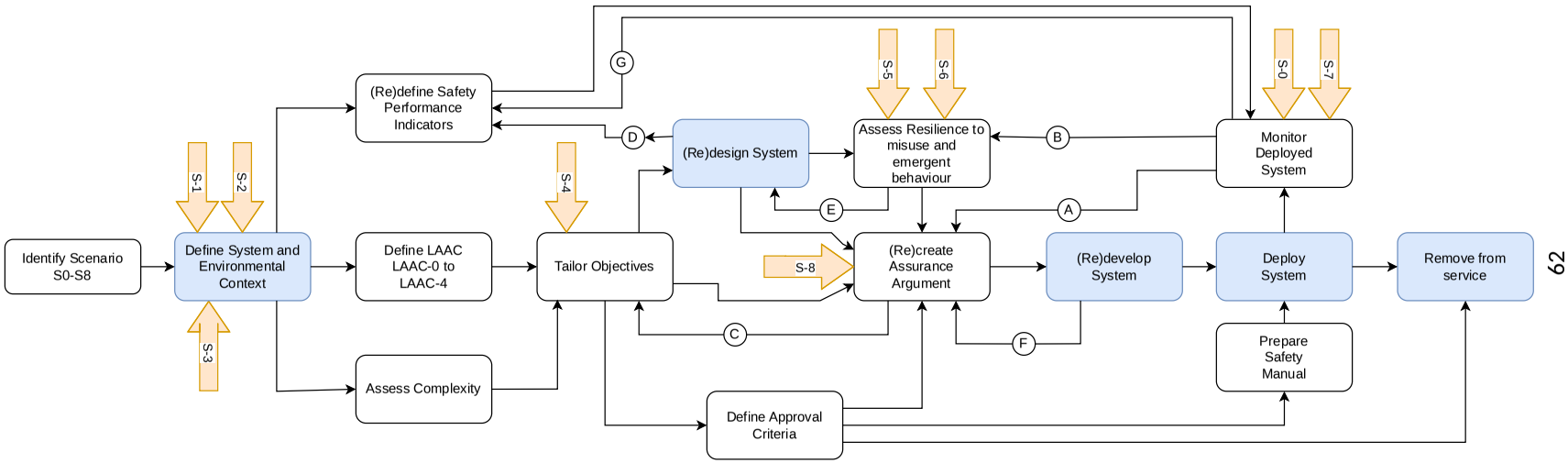


Figure B.1: Generic Analysis Activities

Appendix C

Example Analysis

> Matt Blaze
>
> "Computer science doesn't know how to build complex systems that work reliably. This has been a well-understood problem since the very beginning of programmable computers."

Contents

C.1 Motorway Active Traffic Management 63
C.1.1 Description of the System 63
C.1.2 Purpose of the Analysis 64
C.1.3 Exclusions 64
C.1.4 Analysis of the System 64

This appendix provides a worked example of applying the tools and techniques described in chapter A to realistic complex systems.

C.1 Motorway Active Traffic Management

This section describes an imaginary extension to the Active Traffic Management system in England. It is considered for the purpose of determining how the guidance in this document might be applied to a complex system. There is no current intent actually to implement any of the system described here.

This extension can be considered an example of "S-4: Evolve" in the list of Scenarios given in table 4.1 on page 31.

C.1.1 Description of the System

The "Smart Motorway" system is described in section 2.3.2 on page 15. In England[1] at the time of writing these motorways do not provide the infrastructure for Vehicle-to-Infrastructure (V2I) communication or the interception of Vehicle-to-Vehicle (V2V) communication. This example analysis considers the enhancement of the existing system to include V2I which is seen as a evolutionary step.

For a description of V2I see reference [73]. In summary, V2I provides two-way communication between vehicles and roadside wireless devices:

1. The vehicle can provide information about its intended behaviour to the infrastructure and the infrastructure can make the necessary adjustments — for example, changing a red traffic light when an emergency vehicle approaches it.

[1] The regulations relating to this example differ between the parts of the UK and what is given here is specific to England.

2. The infrastructure can provide information and advice to the driver of the vehicle — for example, a warning of stationary vehicles ahead or assistance with changing lanes. It has been found that vehicle-specific messages can be more effective than general messages on overhead boards.

It is assumed that cars equipped with V2I are also equipped with Vehicle-to-Vehicle (V2V) communication and these two technologies enable the major change to coöperative vehicle control — the vehicles and facilities work together to anticipate potentially adverse conditions. It has also to be assumed that not all vehicles will be V2I-equipped, particularly in the short-term.

While V2I can increase the volume and safety of road traffic, it is subject to complex security requirements: in principle, a vehicle could "pretend" to be an ambulance to receive preferential treatment. The cars themselves are also susceptible to security attacks and research is underway (see, for example, reference [74]) to allow the car to detect that it is under attack and inform the infrastructure of this: effectively to warn the system that its behaviour might become erratic.

It has been proposed that police pursuing a vehicle could have the ability to use V2I to cause it to flash its lights and cut off the fuel to the engine: immobilising the pursued vehicle in a relatively controlled manner.[2] This also raises security concerns: could that operation be hacked?

There are also privacy concerns to be taken into account: potentially private information about the vehicle and driver are broadcast from the vehicle.

C.1.2 Purpose of the Analysis

This possible situation is an example of Scenario "S-4: Evolve" as described in table 4.1 on page 31 and outlined in Appendix B.4.

The purpose of the analysis is to determine whether the introduction of V2I would increase or decrease safety in the light of a mixture of V2I-equipped and legacy vehicles on the smart highway and whether the claim made in table 4.6 on page 36 can be justified.

C.1.3 Exclusions

Fully autonomous vehicles and vehicle platooning[3] are excluded from the roads subject to active traffic management.

However, it is accepted that support for fully autonomous vehicles will be required (e.g., foreign cars on English roads) at some point in the future and analysis is ongoing.

C.1.4 Analysis of the System

LAAC Classification

When the smart motorway system was first introduced into England, the requirement was that it be *"as safe as the system was prior to the change":* effectively GAMAB ("Globalement Au Moins Aussi Bon"). This has since changed to a form of ALARP.

The system provides advice (rather than instructions) — under the current highway code, drivers may choose to ignore this advice. Harm could result from drivers ignoring the advice, but that is not considered in the analysis of the system.

However, the system must not provide unsafe advice: e.g. advise moving into a closed lane, or placing a lane divert right next to a lane divert left.

The V2I system interacts with many vehicles on each stretch of road and, as drivers come to depend on the received advice, unsafe advice from the system could cause significant harm to many people. For this reason, although it is only an advisory system, it is classified as LAAC-4 as described in section 3.2 on page 17.

[2] https://www.irozhlas.cz/zivotni-styl/auto/konec-honicek-a-strelby-do-kol-novy-system-umoznil-policistum-odpojit-privod_2402251531_job

[3] Platooning is the mechanism of allowing a number of vehicles with V2V communication to form a "train" and operate together with reduced separation between vehicles.

Objectives

The principles (highly) recommended for this analysis are listed in table 4.6 on page 36. These principles are described in outline here:

- Highly Recommended: P-1: Assurance Requirements SHALL be established for the SCS.

  Note: the term "Assurance Requirement" is defined on page 5.

  Requirement 1: The introduction of V2I SHALL be governed by the relevant statutory legislation — document [75].

  In particular:

  - All persons preparing safety analyses or justifications SHALL provide evidence of their competence (E/2.2.1)[4].
  - A safety control review group (SCRG) SHALL be established (E/2.11).
    The introduction of V2I is classified as a type C activity (table E/2.8.1).
  - The introduction of V2I SHALL NOT be justified if it leads to increased risk for vehicles not equipped with V2I (E/3.16).

  Requirement 2: The existing Leading Indicators (see definition of this term on page 7) SHALL be extended to include ones relevant to V2I deployment.

  Requirement 3: For roads to be equipped with V2I, Safety Performance Indicators (SPIs) SHALL be kept covering a period from at least 3 years before the introduction of V2I.

  These SPIs will form the baseline required by document [75] and must include, for each hour on the road:

  - The number of vehicles.
  - The number of collisions.
  - The number of injuries to drivers, roadside workers, passengers and pedestrians.
  - The number of fatalities to drivers, roadside workers, passengers and pedestrians.
  - The number of types of incident for which the controllers had no prior briefing and no defined action.

  Requirement 4: When a road is equipped with V2I, statistics SHALL be kept.

  These statistics must include, for each hour on the road:

  - The number of vehicles.
  - The number of vehicles equipped with V2I.
  - The number of collisions.
  - The number of injuries to drivers, workers, passengers and pedestrians.
  - The number of fatalities to drivers, workers, passengers and pedestrians.
  - The number of types of incident for which the controllers had no prior briefing and no defined action.

  Requirement 5: The V2I system SHALL NOT provide information to a driver that would encourage that driver to break rules or otherwise promote unsafe behaviours.

[4] References are to document [75].

For example, informing a driver that the speed monitoring devices are inoperative might encourage inappropriately fast driving.

In general, a high degree of confidence is needed that V2I does not promote unsafe behaviours (that is the current requirement). For example, it must never promote behaviours that contradict the signs and signals non-V2I drivers see. This is an emergent behaviour because sat-nav systems are already seen to be promoting less safe routes: a trend that is hard to manage.

Requirement 6: The goal to be achieved by the introduction of V2I SHALL be defined.

This could be a weighted average of the number of fatalities, injuries, collisions and fuel (including electrical charge) consumed per year.

- Highly Recommended: P-3: Assurance requirements for the SCS SHALL be satisfied.

  Requirement 7: The existing traffic simulation SHALL be enhanced to include defined proportions of vehicles being equipped with V2I.

  The purpose of the simulation is to estimate a level of confidence in the safety of the new system.

- Highly Recommended: P-4: The SCS SHALL be resilient to emergent, unintended and unanticipated behaviours of its elements and their interactions, including those caused by accidental or deliberate misuse.

  V2V and V2I introduce not only accidental and deliberate misuse by drivers, but also the possibility of malicious hacking by external agencies.

- Highly Recommended: P-5: The SCS SHALL be resilient to unanticipated and unintended environmental situations, including accidental or deliberate misuse of other surrounding systems.

  Requirement 8: The introduction of V2I SHALL NOT reduce the existing resilience of the Active Traffic Management System to environmental conditions.

  Requirement 9: The V2I system SHALL be resilient to unintended environmental conditions including accidental or deliberate jamming of the V2I wireless infrastructure.

  Requirement 10: The V2I system SHALL be resilient to incorrect information transmitted by a vehicle.

- Highly Recommended: P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, redeployment and repurposing.

  Requirement 11: The SCRG (see requirement 1) SHALL ensure that the principles recommended and highly recommended in this section are maintained throughout the life of the V2I system.

  The lifetime of the V2I system includes:

  - the period during which the roadside and central control office infrastructure is being installed — covering the safety of the installation teams.
  - The initial test mode where only certain vehicles (e.g., from the installation company) can access it.
  - the period when only a small fraction of the traffic on the road is V2I-equipped — covering the interaction between equipped and non-equipped vehicles.
  - the period when only a small fraction of the traffic on the road is not V2I-equipped — covering the interaction between equipped and non-equipped vehicles.
  - a potential period when non-V2I-equipped vehicles are not permitted on the road. What will prevent a non-equipped driver from entering the road?

Requirement 12: At each change of functionality the Safety Case SHALL be reworked.

- Recommended: P-2: The intent of the SCS assurance requirements SHALL be incorporated within the levels of abstraction of the SCS.

  Requirement 13: Safety argument SHALL be provided for at least the following levels of abstraction:

  - The physical highway (what does the control system know about it?).
  - Control room operator procedures and the automatic signalling system.
  - Operational modes (e.g. 3 lane, 4 lane and transitions between them).
  - The information flow from the control system to road users and vice versa (via roadside emergency phones and V2I signalling).
  - Anticipated driver behaviour in response to the information provided.

- Recommended: P-6: The confidence established in addressing these principles SHALL be commensurate with the level of risk posed by the SCS.

  Although this is an advisory system (see above), it is classified as LAAC-4. The confidence required is therefore high.

Appendix D

Limitations of Traditional Approaches

Contents

D.1 Development Methodologies 68
D.2 Design Modelling Approaches 68
D.3 Verification Approaches 69

This Appendix examines the limitations of traditional ways in which products and systems have been developed (Appendix D.1), designed (Appendix D.2) and in which their adequate safety has been verified (Appendix D.3).

D.1 Development Methodologies

Many development methodologies have been proposed, including waterfall, spiral, agile, V-models and this section considers whether these are applicable to the development of a complex system. Capers Jones' 2012 study (reference [76]) lists 30 development methodologies in use at the time, sorted by their effectiveness at preventing defects. In Jones' study, the most effective methodology was found to be "Mashup (construction from certified reusable components)", similar to Scenario "S-2: Assemble" in table 4.1 on page 31.

The waterfall model was first described in Winston Royce's 1970 paper (reference [77]) which pointed out various reasons why the basic waterfall approach was inappropriate for software development. The paper then states, *"The remainder of this discussion presents five additional features that must be added to this basic approach to eliminate most of the development risks."*

Many of the international standards specify a V-model development methodology. IEC 61508-3:2026, for example, illustrates a V-model for software development that exhibits the first two of the problems identified by Royce in 1970: it assumes that back-tracking occurs only as far as the previous step, and it assumes that requirements can be specified and finalised before any design work has been performed. It also misses several of Royce's essential "five additional features".

It is unlikely that the development of any complex system would require back-tracking only to the previous step. Figure B.1 on page 62 illustrates an approach for the development of a complex system where it will be seen that there are numerous back-tracking paths. It would be expected that this would lead to an agile methodology (number 8 on the list in reference [76]) and certainly not to a waterfall methodology (number 26 in the list).

D.2 Design Modelling Approaches

Reference [48] summarises the inadequacy of conventional design modelling techniques and describes the change of approach needed:

"This shift from complication to complexity has profound implications for modelling. Many traditional modelling approaches assume stable system boundaries, predefined structures, consistent cause–effect relationships, and repeatable execution paths. These assumptions are increasingly violated in real operational settings. Modern socio-technical systems are adaptive, partially autonomous, tightly coupled, and context-sensitive. Human operators, automated subsystems, organisational pressures, environmental conditions, and learning components interact in ways that cannot be fully anticipated at design time. As a result, understanding behaviour — not merely structure — becomes the central modelling challenge.

The modelling question therefore shifts from "How does the system work?" to "How can system behaviour vary?", "Under what conditions do particular patterns of interaction arise?", and "Which modelling commitments should be made before behaviour is observed, and which should be deferred until instantiation?"

D.3 Verification Approaches

To verify that the developed system is adequately safe, IEC 61508-7:2026 recommends:

- dynamic testing (including worst-case and fault injection testing)
- static analysis of code
- Failure Modes and Effects Analysis (FMEA)
- cause-consequence diagrams
- Event Tree Analysis (ETA)
- Failure Modes, Effects and Criticality Analysis (FMECA)
- Fault Tree Analysis (FTA)
- Markov models
- Reliability Block Diagrams (RBD)
- Monte-Carlo (discrete-event) simulation
- Generalised Stochastic Petri Net models (GSPN)

With the exception of discrete-event simulation (see Appendix A.3.4 on page 52) and deep static analysis, each of these listed techniques is inappropriate for the analysis of a complex system because of its underlying assumptions. Some of the these assumptions date from the 1960s and include the following.

1. **Assumption:** A system is the sum of its components.

   Claim: The system's components can be identified and individually analysed. Combining these analyses can provide sufficient insight into the system as a whole.

   Rebuttal: Complex systems tend to have indeterminate and porous boundaries with large variable, numbers of interacting components. These boundaries change during the life of the system. Analysis of the system requires not only analysis of each component, but also consideration of the interactions between those components. Human operators and users now also tend to be critical "components" of a system and bring their own variability in behaviour, level of training, state of mind, experience, risk tolerance, etc.

   Affected Techniques: FMEA, FMECA and RBD are all based on listing the system's components.

2. **Assumption:** A system behaves predictably, as defined by the design and implementation of its components.

   Claim: The components and the interfaces between them are defined so that all possible system behaviours can be foreseen.

   Rebuttal: Complex systems may exhibit "emergent", i.e. unpredicted, behaviour caused not by the individual components, but by the complex interactions of those components.

   Affected Techniques: ETA, Markov models, GSPN and cause-consequence diagrams all rely on the predictable behaviour of the system given a particular stimulus. Discrete-event simulation can also be affected by the causal chain, but is less susceptible to it than the other techniques — discrete-event simulation will often surprise the analyst.

3. **Assumption:** A hazardous situation occurs only when a component fails or malfunctions.

   Claim: Many of the international safety standards explicitly state that only those hazardous situations caused by failure or malfunction need be considered. For example, the automotive standard ISO 26262:2018 defines a "Hazard" as: *"potential source of harm caused by malfunctioning behaviour of the item."*

   Rebuttal: Various studies (e.g. reference [51]) have noted that, in most dangerous situations, every component in the system operated exactly as intended.

   Affected Techniques: FMEA, FMECA, FTA and RBD are all tools designed to analyse the system behaviour if a component fails or malfunctions.

4. **Assumption:** Dynamic testing (subjecting the system to test cases) is an effective way of demonstrating that a system functions as specified.

   Claim: The system-level requirements of a system define its correct behaviour for each set of inputs. By exposing the system to the conditions specified by each requirement and checking the system's response, its correct operation can be demonstrated.

   Rebuttal: The introduction of software into systems massively increased the number of states a system can assume. One estimate indicates that an operating system such as Linux has 10^{300} different states.[1] For a software-based system running on a modern processor (itself with a significant number of *errata* describing non-deterministic behaviour), the fraction of the system states that can be explored with dynamic testing is tiny. ISO 29119-1, the software testing standard, reflects this by stating *"Dynamic testing is necessary, but not sufficient to provide reasonable assurance that software will perform as intended."* Additionally, this claim assumes that a complete set of requirements exists. For many systems, particularly those created through machine learning, it is not possible to define the requirements.

   Affected Techniques: The effectiveness of all forms of dynamic testing, including worst-case testing, are affected by the failure of this assumption. Fault injection testing is perhaps the least affected form of dynamic testing because the presence of unanticipated faults can result in unanticipated behaviour.

[1] To put this into context, there are about 10^{80} atomic particles in the universe.

Appendix E

Sample Incidents in Complex Systems

Contents

E.1 Introduction 71
E.2 The Herald of Free Enterprise 71
E.3 CrowdStrike Software Update 72
E.4 Boeing 737 Max 8 MCAS 72
E.5 The R.101 Airship 73
E.6 Clapham Junction Rail Crash 74
E.7 Common Themes 75

E.1 Introduction

This appendix lists some safety-related incidents that occurred in complex systems *because of their complexity.*

E.2 The Herald of Free Enterprise

On 6th March, 1987, the car ferry, Herald of Free Enterprise, left the port of Zeebrugge with its bow doors open. As it left the harbour, ingress of water caused the ferry to capsize, leading to the death of 193 passengers and crew.

Reference [78] is the official report of this incident and reference [44] contains the outline of a FRAM[1] analysis of the incident. Reference [78] summarises the causes of the event as follows:

> *The Court, having carefully inquired into the circumstances attending the above-mentioned shipping casualty, finds, for the reasons stated in the Report, that the capsizing of the HERALD OF FREE ENTERPRISE was partly caused or contributed to by serious negligence in the discharge of their duties by Captain David Lewry (Master), Mr. Leslie Sabel (Chief Officer) and Mr. Mark Victor Stanley (Assistant bosun), and partly caused or contributed to by the fault of Townsend Car Ferries Limited (the Owners).*

The system, comprising the ship, the crew, the designers and the company, can be considered complex because of the interaction between human and technical factors. The ship was designed in such a way that the Captain on the bridge had no way of knowing whether the bow doors were closed. Although there had been requests for an indicator to be installed on the bridge giving the state of the doors (section 18 of [78]), none had been fitted and the Captain was relying on there being no report that the doors were open. Following a previous incident when another ship sailed with its bow doors open and a request was made for an indicator on the bridge, a company superintendent wrote:

[1] See Appendix A.3.1 on page 49.

Bow and stern door remote indication

...I concur in part with Mr. King's penultimate paragraph that the project is unnecessary and not the real answer to the problem. In short, if the bow or stern doors are left open, then the person responsible for closing them should be disciplined. ...So in conclusion, the Bridge indication is a 'no go'.

Reference [78] also states

That [installing a bridge warning light] was simple is illustrated by the fact that within a matter of days after the disaster indicator lights were installed in the remaining Spirit class ships and other ships of the fleet.

There was time pressure to get the ship moving (section 11 of [78]); the ship was significantly overloaded with passengers (section 17 of [78]); there was confusion over crew responsibilities and the assistant bosun, who should have ensured that the bow doors were closed, was asleep in his cabin when the ship departed (section 10 of [78]).

E.3 CrowdStrike Software Update

On 19th July 2024 at 04:09 UTC, the American cybersecurity company CrowdStrike distributed a faulty configuration update for its Falcon sensor software running on Windows PCs and servers. This resulted in about 8.5 million systems crashing and being unable to restart properly. For many affected systems, technical staff had to reboot the computers manually, deleting particular files during the reboot. A similar CrowdStrike problem had caused outages on Linux systems in the previous April.[2]

As a result, 5,078 scheduled airline flights were cancelled, many banks were unable to serve customers, in parts of the USA it was impossible to reach emergency services (911), many hospitals were forced to cancel non-emergency surgeries and many media companies were unable to broadcast. The outage also provided an opportunity for criminals to make phishing attacks: *"we are calling from CrowdStrike to help you recover your systems"*.

The CrowdStrike update illustrates several characteristics of a complex system listed in section 2.2.1 on page 11:

- There were many subsystems: it is estimated that 8.5 million subsystems were affected.
- It was geographically distributed, affecting systems in Australasia, Asia, Europe, Africa and the Americas.
- A single (incorrect) line of code caused worldwide disruption: an example of non-linear behaviour.
- The system had indeterminate boundaries and those boundaries emerged only after the incident: it would have been difficult before the incident to have determined which subsystems would be affected by the bad code.
- Its effects on many industries (hospitals, airlines, banks, etc.) emerged unexpectedly, but can be explained in retrospect.

The worldwide financial damage has been estimated to be at least US$10 billion.

E.4 Boeing 737 Max 8 MCAS

The development of the Boeing 737 Max 8 was actually a Scenario "S-1: Build New" in the categories given in table 4.1 on page 31, but, to avoid retraining pilots, was handled as a "S-4: Evolve".

The executive summary of reference [79] begins

[2] See `https://www.theregister.com/2024/07/21/crowdstrike_linux_crashes_restoration_tools/`

> *Technical design flaws, faulty assumptions about pilot responses, and management failures by both The Boeing Company and the Federal Aviation Administration (FAA) played instrumental and causative roles in the chain of errors that led to the crashes of Lion Air flight 610 in October 2018, and Ethiopian Airlines flight 302 in March 2019 that resulted in the tragic and preventable deaths of 346 people.*

In its 737 Max 8 aircraft, Boeing introduced larger engines mounted further forward on the wing. During wind-tunnel testing, this was found to change the aircraft's flight characteristics, introducing a pitch up-tendency at high angles of attack. To avoid airlines having to retrain their pilots, Boeing added a Maneuvering Characteristics Augmentation System (MCAS) which received signals from a single Angle-of-Attack sensor and could automatically trim the aircraft's nose-up or down. It was argued that no additional pilot training was necessary as, if the MCAS malfunctioned, the effect would be similar to handling a runaway stabilizer trim, something that pilots were already trained to handle on other 737 models. The MCAS was not mentioned in the flight crew manual.

As described in reference [80], the MCAS was initially limited to moving the trim only 0.6°, but after test flights, this was increased to 2.5° and the MCAS was changed to process input from a single Angle-of-Attack sensor. Unfortunately, the analysis performed by Boeing did not take into account the MCAS activating repeatedly, eventually driving the aircraft into an unrecoverable nose-down condition.

Reference [79] describes the pressure put onto the development team working on the MAX 8, including a "countdown clock" being installed in the project's conference room to emphasise the significance of the schedule.

According to the Seattle Times for 5th May 2019:

> *In 2016, a senior Boeing engineer whose job was to act on behalf of the FAA balked at Boeing management demands for less stringent testing of the fire-suppression system around the [737 Max 8's] new LEAP engines. ... He convened a meeting of all the certification engineers in his unit, who ... agreed with his assessment. Management initially rejected their position, and only after another senior engineer from outside the MAX program intervened did managers finally agree to beef up the testing to a level the engineer could accept, ...*
>
> *Less than a month after his peers had backed him, Boeing abruptly removed him from the program...*

The 737 Max 8 development was a complex, socio-technical system and the tragic incidents occurred not because of a component failure (reference [80] points out that the MCAS in both cases operated exactly as it was designed to operate), but because of its complex nature.

E.5 The R.101 Airship

The development of the R.101 airship was a Scenario "S-1: Build New" in the categories given in table 4.1 on page 31: the R.101 was the largest airship in the world and was built incorporating new technologies throughout.

At 18:30 on 4th October 1930, the R.101 took off from Cardington, UK, in poor and deteriorating weather conditions on her maiden voyage to India. On board, in addition to the crew, were the airship's designer, the Director of Airship Development, the Director of Civil Aviation and the Secretary of State for Air. Some seven and a half hours after taking off, the airship crashed just south of Beauvais in France, killing all but eight of the fifty-four people on board. Two of the eight survivors died shortly afterwards from their injuries.

Reference [81] is the report of the official government investigation into the incident. This finds that technical problems and political pressure led to the R.101 departing without having completed any of the scheduled trial flights after substantial changes to the structure of the airship. In September 1929, as construction was completed, it was found that the R.101 had only 35 tons (rather than the expected 60 tons) of useful lift and effectively needed forward motion as an aircraft to keep it in the air. To resolve this, in late July, the airship was cut in half and a new 14 metre section, with additional gasbags, was inserted.

When asked to approve the airship being cut in half, on 14th July, Lord Thomson, the Secretary of State for Air, wrote in a minute, *"So long as the R.101 is ready to go to India by the last week in September this further delay in getting her altered may pass. I must insist on the programme for the Indian Flight being adhered to as I have made my plans accordingly."* This placed time constraints on the mechanical work of cutting the airship in half, on the time-consuming calculations that were needed to ensure she was still stable (then performed manually,

rather than by digital computer) and on the time available for test flights. In fact, only a single test flight of just under 17 hours was flown, and that in perfect weather conditions. Even during that short flight, one engine failed because of a burst oil cooler.

At the request of the investigation, additional calculations were performed after the accident by the National Physical Laboratory (NPL) and these explained the manner in which the accident occurred. There had been no time to perform these calculations before the flight.

The official report (reference [81]) concludes, *"It is clear that if those responsible had been entirely free to choose the time and the weather in which the R.101 should start for the first flight ever undertaken by any airship to India, and if the only considerations governing their choice were considerations of meteorology and preparation for the voyage, the R.101 would not have started when she did. ...It is impossible to avoid the conclusion that the R.101 would not have started for India on the evening of October 4th if it had not been that reasons of public policy were considered as making it highly desirable for her to do so if she could."*

The R.100 was being built by private industry in parallel with the government building the R.101 and this created competition. In July/August 1930 the R.100 had successfully flown across the Atlantic to Montréal and back, putting additional pressure on the R.101 construction.

When the technical challenges are coupled with the social pressures, the R.101 can be seen as a complex system; its failure resulted not from any single technical cause, but from it being a complex system.

E.6 Clapham Junction Rail Crash

On 12th December 1988 a serious railway accident occurred near Clapham Junction station in the UK, with 35 fatalities and almost 500 injuries. The report on the accident is given in reference [26]. A train heading into London ran into the back of a stationary train and was thrown off its track, hitting a train travelling out of London. Almost all of the injuries occurred to people in the leading coaches of the London-bound train.

The root cause of the Clapham Junction accident was technical: the signalling system failed because, during upgrading carried out on 27th November 1988, a wire that should have been removed, or at least insulated, was simply disconnected at one end. Unrelated maintenance work on the day before the accident moved the uninsulated wire so that it made contact with the old circuit. This prevented the signal from displaying a red light.

Testing as required by regulations was not carried out after either of the maintenance activities.

The report into the incident (reference [26]) lists a number of human factors that contributed to the wire being left uninsulated: procedures related to the wiring diagrams given to the technician not being followed, the wiring diagrams themselves being incomplete, an interruption occurring just as the wire had been disconnected, the two technicians performing the upgrade not trusting each other, the 7-day week worked by the senior technician up to the date of the accident, the lack of a visual check of the work by the assigned supervisor and poor lighting in the room where the changes were made.

During the enquiry, it was found that the senior technician had had very little training and, throughout his 16 years performing this type of maintenance, had never cut back or insulated equivalent wires, this never having been noticed during inspections. The upgrade that caused the accident was one of a large number that were scheduled to occur between 1984 and 1989, replacing equipment that had been installed in the 1930s (*cf.* Scenario "S-4: Evolve" in the categories given in table 4.1 on page 31), and the timetable for the upgrades was criticised as being too short: *"It was not a safe working practice to allow so many men to work excessive levels of overtime for a sustained period. The work should have been better organised to ensure that supervisors were fulfilling their proper role and not acting as senior technicians."*

Section 7.40 of reference [26] concludes:

> *"It is inevitable in this chapter that the name of Mr Hemingway [the senior technician] has repeatedly been at the centre of the factual description of the wiring errors which caused the accident. He must and does carry a heavy burden of responsibility for the accident and its consequences. As the evidence at the Investigation developed, however, it became abundantly clear that such a responsibility was not his alone. The evidence which the Investigation heard demonstrated that such responsibility must be shared by the many others who had permitted a situation to exist in the S&T [Signal and Telecommunications] Department in which, not only could such errors be made in the first place, but they could*

be permitted to remain undetected when the work was inspected, tested and commissioned back into public service."

Section 13 of the report comments on the Safety Culture of British Railways at the time:

> *"13.2 Management systems must ensure that there is in being a regime which will preserve the first place of safety in the running of the railway. It is not enough to talk in terms of 'absolute safety' and of 'zero accidents'. There must also be proper organisation and management to ensure that actions live up to words."*
>
> *"13.3 Sadly, although the sincerity of the beliefs of those in BR at the time of the Clapham Junction accident who uttered such words cannot for a moment be doubted, there was a distressing lack of organisation and management on the part of some whose duty it was to put those words into practice. The result was that the true position in relation to safety lagged frighteningly far behind the idealism of the words."*

It is significant that the upgrade work which led to the accident was designed to improve the safety of the signalling system: Scenario "S-3: Restructure" in table 4.1 on page 31.

E.7 Common Themes

The examples given above have common factors:

- Time constraints.

  This is particularly true of the R.101 incident where significant UK government pressure was applied to get the airship to India in time for the Commonwealth meeting. But time pressures also affected the Herald of Free Enterprise (it was 5 minutes late leaving port after a number of directives from the operations manager saying that it must always leave 15 minutes early), Boeing 737 (countdown clocks in meeting rooms) and the Clapham Junction (7 day working weeks) incidents.

  It could be argued that time pressure can be the factor that makes a system complex. When working under time pressure, individual developers can make independent decisions, leading to unpredictable behaviour of the team: for example, John doing X while not knowing that Jane is doing Y can lead to a dangerous situation.

- Normalisation of Deviation.

  This is particularly true of the Clapham Junction example where the senior technician had been leaving wires uninsulated for years without this being noted by inspectors. There had also been incidents similar to that at Clapham Junction at Oxted in November 1985 and at Queenstown Road in May 1988. In both cases, incorrect wiring led to a signal showing green when it should not have done. As reference [26] remarks, *"No one in management took the good hard look which was the essential logical sequel to the Queenstown Road incident at what was going on and going wrong in the testing of new installations in the S&T Department. If they had done, the almost cavalier disregard for proper practice that was rife in the Department, with the consequent serious threat to safety, might well have been discovered."*

  The CrowdStrike outage occurred in July 2024, but there had been a similar outage in the previous April and the lessons had not been learned from that.

  There was also normalisation of deviation apparent in the Herald of Free Enterprise (acceptance of no door indicator on the bridge) and R.101 (acceptance of reduced testing).

- Satisfying Assurance Requirements.

  Section 3.3 on page 17 identifies various assurance principles, including:

  P-1: Assurance Requirements SHALL be established for the SCS.

  and

P-3: Assurance requirements for the SCS SHALL be satisfied.

In the examples, these principles had ostensibly been met: in each case assurance requirements had been established defining the development process and requiring approval of the system by an approved person. This had been met in each case. In the case of the R.101, an airworthiness certificate had been issued; in the case of the Clapham Junction crash, the (faulty) work of the technician concerned had been approved and signed off by two levels of appropriate inspectors; in the case of the Boeing 737 Max 8, the (flawed) design was signed off by the representative of the FAA; in the case of the Herald of Free Enterprise, the lack of a door-open indication on the bridge had been approved by company management and the ship was officially sea-worthy with a Class II Passenger Certificate (allowing passengers to be carried on international voyages of not more than 600 miles).

This indicates that, in each of the examples, assurance requirements had been established and satisfied, but that those requirements were inadequate. Section 6.2.3 on page 44 poses the question of whether this inadequacy could have been identified before each accident.

- Lack of Safety Culture at Management Level.

  This is evident in the quotation given in the Herald of Free Enterprise example and a whole section of the official report (reference [78]) into that accident is devoted to the responsibility of management. In particular:

  > It was the failure to give clear orders about the duties of the Officers on the Zeebrugge run which contributed so greatly to the causes of this disaster. Mr. Clarke, on behalf of the Company, said that it was not the responsibility of Mr. Develin to see that the Company orders were properly drafted. In answer to the question, "Who was responsible?" Mr. Clarke said "Well in truth, nobody, though there ought to have been". The Board of Directors must accept a heavy responsibility for their lamentable lack of directions. Individually and collectively they lacked a sense of responsibility. This left, what Mr. Owen so aptly described as, "a vacuum at the centre".

  The lack of Safety Culture at management level is also pointed out explicitly in the reports on the Boeing 737, R.101 and Clapham Junction incidents.

Appendix F

Preparation of this Document

This document was prepared by the Safer Complex Systems Working Group of the Safety Critical Systems Club. The following people contributed to the production of this guidance document (in alphabetical order):

Abhinay Paladugu

Abigail Post

Arun Chakrapani Rao

Chris Hobbs (editor of this Guidance)

Dave Banham

David Slater

James Catmur

Louise Harney

Mike Parsons

Narinder Dhillon

Paul Hampton

Sal Hargis

Stephen Barrett

Stephen Bull

... and other members of the Working Group

Appendix G

Document History

Date	Version	Notes
04 Feb 2024	0.1	Metadata version for initial comments
04 Feb 2024	0.2	Incorporating comments from MP
23 Mar 2024	0.3	Incorporating comments from SBull and MP
05 Apr 2024	0.4	Incorporating comments from SBull, PBL and MP
09 Apr 2024	0.5	Changes to Chapter 2 following MP and CH meeting
12 Apr 2024	0.6	Changes to Chapter 3 following meeting between MP, SBull and CH
19 Apr 2024	0.7	Further changes to Chapter 3 following meeting between MP and CH
30 Apr 2024	0.8	Following discussion at 9th WG meeting
16 May 2024	0.9	Including comments from WG members
11 Jun 2024	0.10	Including comments made at 10th WG Meeting
23 Jun 2024	0.11	Rework of Chapter 1 and other changes following meeting between MP and CH. Section 3.2 added.
16 Jul 2024	0.12	Modifications suggested on the common spreadsheet.
16 Sep 2024	0.13	Modifications suggested at meetings between MP and CH
13 Oct 2024	0.14	Incorporating changes suggested by LH
15 Oct 2024	0.15	Incorporating changes suggested by MP in section 2.2
12 Nov 2024	0.16	Incorporating changes suggested at WG meeting on 16th Oct 2024
10 Dec 2024	0.17	Incorporating changes from meetings between MP and CH
11 Dec 2024	0.18	Incorporating changes suggested at WG meeting on 11th Dec 2024
19 Dec 2024	0.19	Incorporating changes suggested by PBL and MP
10 Jan 2025	0.20	Minor editorial changes
28 Jan 2025	1.0	Issued for SSS'25
27 Apr 2025	1.1	Incorporating changes suggested by JC and SBull
12 Jun 2025	1.2	Example provided by JC introduced in Appendix C.1
22 Jul 2025	1.3	Miscellaneous changes, particularly in Appendix E
17 Sep 2025	1.4	Miscellaneous changes
19 Sep 2025	1.5	Following discussion at working group meeting on 17th Sep 2025
01 Oct 2025	1.6	Following editorial meeting held 1st Oct 2025
10 Dec 2025	1.7	Following WG meeting held 28th Oct 2025. Input from AP, SH and AP.
10 Jan 2026	1.8	Following WG meeting held 10th December 2025.
16 Jan 2026	1.9	Following editorial meetings held in January 2025.
30 Jan 2026	2.0	Following WG meeting held 29th January 2025: Issued for SSS'26

Figure G.1: History of this Document

Bibliography

[1] E. Thompson, *Escape from Model Land: How mathematical models can lead us astray and what we can do about it*. Basic Books, 2022.

[2] S. Sheard, S. Cook, E. Honour, D. Hybertson, J. Krupa, J. McEver, D. McKinney, P. Ondrus, A. Ryan, R. Scheurer, J. Singer, J. Sparber, and B. White, "INCOSE: A Complexity Primer for Systems Engineers: Revision 1," International Council on Systems Engineering, Tech. Rep., 2021.

[3] D.J.Snowden and M.E.Boone, "A leader's framework for decision making," *Harvard Business Review*, vol. 85, 11 2007, https://www.systemswisdom.com/sites/default/files/Snowdon-and-Boone-A-Leader's-Framework-for-Decision-Making_0.pdf.

[4] G. Lewes, *Problems of Life and Mind*, ser. Kessinger publishing's rare reprints. Kessinger Publishing, 1879. [Online]. Available: https://books.google.ca/books?id=jjDSuo6U1Y8C

[5] P. Koopman and W. Widen, "Redefining Safety for Autonomous Vehicles," in *SafeComp 2024*, 2024. [Online]. Available: https://arxiv.org/abs/2404.16768

[6] D. Woods, "The Theory of Graceful Extensibility: Basic rules that govern adaptive systems," *Environment Systems and Decisions*, vol. 38, 12 2018.

[7] UL, *UL 4600: Standard for Safety for the Evaluation of Autonomous Products*. UL, 2022.

[8] ISO, *ISO 21448: Road Vehicles — Safety of the Intended Functionality*. ISO, 2022.

[9] J. A. McDermid, S. Burton, P. Garnett, and R. Weaver, *Safer Complex Systems: An Initial Framework*. Royal Academy of Engineering, 2021.

[10] Royal Academy of Engineering, "Safer Complex Systems: An Initial Framework," Royal Academy of Engineering, Tech. Rep., 2020. [Online]. Available: https://engineeringx.raeng.org.uk/media/4wxiazh3/engineering-x-safer-complex-systems-an-initial-framework-report-v22.pdf

[11] E. Estrada, "What is a complex system, after all?" in *Foundations of Science*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:266364747

[12] P. Érdi, *Complexity Explained*. Springer Verlag, 2008.

[13] T. Harsanyi, "Working on Complex Systems," https://www.thecoder.cafe/p/complex-systems.

[14] A. Phinith and A. Walke, "Governance of Complex Systems: Roundtable report 2024," Royal Academy of Engineering, Tech. Rep., 2024. [Online]. Available: https://engineeringx.raeng.org.uk/media/xxwmwpf2/governance-of-complex-systems-final.pdf

[15] C. Nemeth and R. Cook, "Infusing Healthcare with Resilience," *INCOSE International Symposium*, vol. 20, pp. 1073–1087, 07 2010.

[16] Department for Transport, "Smart Motorway Safety: Evidence Stocktake and Action Plan," https://www.gov.uk/government/publications/smart-motorway-evidence-stocktake-and-action-plan, 2020.

[17] K. Feigh, A. Pritchett, A. Shah, S. Kalaver, A. Jadhav, D. Holl, R. Bea, and A. Gilgur, "Analyzing Air Traffic Management Systems Using Agent-based Modeling and Simulation," 01 2005.

[18] S. Bouarfa, H. Blom, R. Curran, and M. Everdij, "Agent-based modeling and simulation of emergent behavior in air transportation," *Complex Adaptive Systems Modeling*, 2013.

[19] S. O. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Mar. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2119

[20] B. Leiba, "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words," RFC 8174, May 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8174

[21] J. Christie, "The Post Office Horizon IT scandal and the presumption of the dependability of computer evidence," *Digital Evidence and Electronic Signature Law Review*, pp. 49–70, 11 2020.

[22] J. Rasmussen, "Risk Management in a Dynamic Society: A Modelling Problem," *Safety Science*, 1997.

[23] M. Gleirscher, R. Calinescu, and J. Woodcock, "RiskStructures: A design algebra for risk-aware machines," *Form. Asp. Comput.*, vol. 33, no. 4–5, p. 763–802, Aug. 2021. [Online]. Available: https://doi.org/10.1007/s00165-021-00545-4

[24] E. Candela, O. Doustaly, L. Parada, F. Feng, Y. Demiris, and P. Angeloudis, "Risk-aware controller for autonomous vehicles using model-based collision prediction and reinforcement learning," *Artificial Intelligence*, vol. 320, p. 103923, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370223000693

[25] E. Denney, G. Pai, and I. Habli, "Dynamic safety cases for through-life safety assurance," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15. IEEE Press, 2015, p. 587–590.

[26] Anthony Hidden QC, "Investigation into the Clapham Junction Railway Accident," Department of Transport, Tech. Rep., 1989. [Online]. Available: https://www.railwaysarchive.co.uk/documents/DoT_Hidden001.pdf

[27] W. R. Ashby, "Principles of the Self-Organizing System," in *Principles of Self-Organization: Transactions of the University of Illinois Symposium*, H. V. Foerster and G. W. Z. Jr, Eds. Pergamon Press, 1962, pp. 255–278.

[28] J. D. R. de Raadt, "Ashby's Law of Requisite Variety: an Empirical Study," *Cybernetics and Systems: An International Journal*, vol. 18, pp. 517–536, 01 1987.

[29] D. Slater, "What is Safety? A Basic Instinct or a Scientific Definition?" p. 11, 06 2025, https://doi.org/10.13140/RG.2.2.28021.90086.

[30] IEC, *IEC Standard 62740:2015, Root Cause Analysis (RCA)*. IEC, 2015.

[31] C.-G. Wu, X. Xu, B. Zhang, and Y.-L. Na, "Domain Ontology for Scenario-Based Hazard Evaluation," *Safety Science*, vol. 60, pp. 21–34, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092575351300132X

[32] J. Zhou, K. Hänninen, K. Lundqvist, and L. Provenzano, "An Ontological Approach to Identify the Causes of Hazards for Safety-Critical Systems," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, 2017, pp. 405–413.

[33] A. Aziz, S. Ahmed, and F. I. Khan, "An Ontology-Based Methodology for Hazard Identification and Causation Analysis," *Process Safety and Environmental Protection*, vol. 123, pp. 87–98, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095758201831365X

[34] A. Parisaca Vargas and R. Bloomfield, “Using Ontologies to Support Model-based Exploration of the Dependencies between Causes and Consequences of Hazards,” in *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, ser. IC3K 2015. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2015, p. 316–327. [Online]. Available: https://doi.org/10.5220/0005618003160327

[35] P. B. Ladkin, “Ontological Analysis,” *Safety Systems*, vol. 14, no. 3, 2005.

[36] L. Lamport, *Specifying Systems: The TLA$^+$ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2003.

[37] M. Butler and S. Hallerstede, “The Rodin formal modelling tool,” in *Proceedings of the 2007 International Conference on Formal Methods in Industry*, ser. FACS-FMI’07. Swindon, GBR: BCS Learning & Development Ltd., 2007, p. 2.

[38] P. B. Ladkin, “Ontological Hazard Analysis of a Communications Bus,” https://api.semanticscholar.org/CorpusID:8341461, 2010.

[39] R. R. Lutz, “Analyzing software requirements errors in safety-critical, embedded systems,” *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pp. 126–133, 1993. [Online]. Available: https://api.semanticscholar.org/CorpusID:1882843

[40] L. E. Prokop, “Historical Aerospace Software Errors Categorized to Influence Fault Tolerance,” *2024 IEEE Aerospace Conference*, pp. 1–12, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269775596

[41] E. Conrad, L. Titolo, D. Giannakopoulou, T. Pressburger, and A. Dutle, “A compositional proof framework for FRETish requirements,” in *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, ser. CPP 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 68–81. [Online]. Available: https://doi.org/10.1145/3497775.3503685

[42] P. Underwood and P. Waterson, “A critical review of the STAMP, FRAM and Accimap systemic accident analysis models,” in *Advances in Human Aspects of Road and Rail Transportation*, 2012, pp. 385 – 394.

[43] E. Hollnagel, *Barriers and Accident Prevention*. CRC Press, 2004.

[44] ——, *FRAM: The Functional Resonance Analysis Method — Modelling Complex Socio-technical Systems*. CRC Press, 2012.

[45] R. Clay-Williams, J. Hounsgaard, and E. Hollnagel, “Where the rubber meets the road: using FRAM to align work-as-imagined with work-as-done when implementing clinical guidelines,” *Implementation science : IS*, vol. 10, p. 125, August 2015. [Online]. Available: https://europepmc.org/articles/PMC4553017

[46] G. Praetorius, M. Lundh, and M. Lützhöft, “Learning from the past for pro-activity – A re-analysis of the accident of the MV Herald of Free Enterprise,” 06 2011.

[47] E. Hollnagel, “The Functional Resonance Analysis Method: A brief Guide on how to use the FRAM,” https://functionalresonance.com/onewebmedia/FRAM%20Handbook%202018%20v5.pdf, 2018.

[48] D. Slater, “Modelling Contemporary Complex Systems: From Structure to Possibility,” p. 7, 12 2025.

[49] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 01 2012. [Online]. Available: https://doi.org/10.7551/mitpress/8179.001.0001

[50] J. Rasmussen and I. Svedung, *Proactive Risk Management in a Dynamic Society*. Swedish Rescue Services Agency, 2000.

[51] N. Leveson, "A new accident model for engineering safer systems," *Safety Science*, vol. 42, no. 4, pp. 237–270, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092575350300047X

[52] K. Walker, D. Woods, and M. Rayo, "Multiple Systemic Contributors versus Root Cause: Learning from a NASA Near Miss," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 60, pp. 264–264, 09 2016.

[53] JCGM, *Evaluation of measurement data — Supplement 1 to the Guide to the expression of uncertainty in measurement – Propagation of distributions using a Monte Carlo method*. Joint Committee for Guides in Metrology, 2008.

[54] H. A. P. Blom, S. H. Stroeve, and H. H. de Jong, "Safety Risk Assessment by Monte Carlo Simulation of Complex Safety Critical Operations," in *Proceedings of the 14th Safety-Critical Systems Symposium*. Springer, 2006, pp. 47–67.

[55] J. Botham, G. Dhadyalla, A. Powell, P. Miller, O. Haas, D. McGeoch, A. Chakrapani Rao, C. O'Halloran, J. Kiec, A. Farooq, S. Poushpas, and N. Tudor, "PICASSOS – Practical Applications of Automated Formal Methods to Safety Related Automotive Systems," 03 2017.

[56] G. Holzmann, *The SPIN Model Checker*. Addison-Wesley, 2004.

[57] Peter Bernard Ladkin, "Root Cause Analysis: Terms and Definitions, Accimaps, MES, SOL and WBA," http://www.rvs.uni-bielefeld.de/publications/Papers/LadkinRCAoverview20130120.pdf, 2013.

[58] B. Fahlbruch and M. Schöbel, "SOL – Safety through Organizational Learning: A Method for Event Analysis," *Safety Science*, vol. 49, pp. 27–31, 01 2011.

[59] K. M. Feigh, A. R. Pritchett, S. Mamessier, and G. Gelman, "Generic agent models for simulations of concepts of operation: Part 2," *Journal of Aerospace Information Systems*, vol. 11, no. 10, pp. 623–631, Oct. 2014.

[60] A. R. Pritchett, K. M. Feigh, S. Y. Kim, and S. K. Kannan, "Work models that compute to describe multiagent concepts of operation: Part 1," *Journal of Aerospace Information Systems*, vol. 11, no. 10, pp. 610–622, Oct. 2014.

[61] A. Paladugu, A. Fernandes, and M. IJtsma, "Computational Simulation of Distributed Work as a Discovery Tool for Envisioning Future Operation," *Journal of Cognitive Engineering and Decision Making*, vol. 19, no. 4, pp. 395–419, 2025. [Online]. Available: https://doi.org/10.1177/15553434251327698

[62] A. R. Pritchett, R. P. Bhattacharyya, and M. Ijtsma, "Computational assessment of authority and responsibility in air traffic concepts of operation," *Journal of Air Transportation*, vol. 24, no. 3, pp. 93–101, 2016.

[63] M. IJtsma, L. M. Ma, A. R. Pritchett, and K. M. Feigh, "Computational Methodology for the Allocation of Work and Interaction in Human-Robot Teams," *Journal of Cognitive Engineering and Decision Making*, vol. 13, no. 4, pp. 221–241, 2019. [Online]. Available: https://doi.org/10.1177/1555343419869484

[64] L. M. Ma, M. Ijtsma, K. M. Feigh, A. Paladugu, and A. R. Pritchett, "Modelling and evaluating failures in human-robot teaming using simulation," in *2018 IEEE Aerospace Conference*, *AERO 2018*, ser. IEEE Aerospace Conference Proceedings. United States: IEEE Computer Society, Jun. 2018, pp. 1–16.

[65] E. Barrett, A. Paladugu, and M. Ijtsma, "Multi-Agent Simulation to Envision Communication Strategies in Future Air Mobility Operations," *Journal of Aerospace Computing, Information and Communication*, vol. 21, pp. 605–683, 06 2024.

[66] M. del Carmen Pardo-Ferreira, M. Martínez-Rojas, F. Salguero-Caparrós, and J. C. Rubio-Romero, "Evolution of the Functional Resonance Analysis Method (FRAM) through the combination with other methods," *Dirección y Organización*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:199007249

[67] R. Patriarca, M. Chatzimichailidou, N. Karanikas, and G. D. Gravio, "The past and present of System-Theoretic Accident Model And Processes (STAMP) and its associated techniques: A scoping review," *Safety Science*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:243965110

[68] A. Mofidi and S. Nadeau, "A Review of the Application of FRAM and STAMP Approaches Combined with Other Methods," in *Applied Human Factors and Ergonomics (AHFE2022)*, vol. 52, 06 2022, pp. 1–17.

[69] T. Saaty, "The analytic hierarchy and analytic network measurement processes: Applications to decisions under Risk," *European Journal of Pure and Applied Mathematics [electronic only]*, vol. 1, 01 2008.

[70] D. V. Raoi, K. Polzin, M. Houts, F. Curran, and M. Rodriguez, "Application of Analytical Hierarchy Process for Narrowing Down NEP Candidate Reactor Designs," NASA, Tech. Rep., 2021. [Online]. Available: https://ntrs.nasa.gov/api/citations/20210018672/downloads/JANNAF%2021%20AHP%20Final.pdf

[71] D. L. Alderson, G. G. Brown, and W. M. Carlyle, "Operational Models of Infrastructure Resilience," *Risk Analysis*, vol. 35, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:16706716

[72] N. Anand, *Are We Learning from Accidents? A quandary, a question and a way forward*. Novellus, 2024.

[73] D. Kanthavel, S. Sangeetha, and K. Keerthana, "An empirical study of vehicle to infrastructure communications - An intense learning of smart infrastructure for safety and mobility," *International Journal of Intelligent Networks*, vol. 2, pp. 77–82, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666603021000105

[74] S. Bian, M. Farsi, N. L. Azad, and C. Hobbs, "Data-Driven Intrusion Detection in Vehicles: Integrating Unscented Kalman Filter (UKF) with Machine Learning," in *International Conference on Informatics in Control, Automation and Robotics*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:274202359

[75] "GG104: Requirements for safety risk assessment," National Highways, Tech. Rep., 2025. [Online]. Available: https://www.standardsforhighways.co.uk/search/6393d1c9-31da-4675-b1b8-c43703409f1e

[76] C. Jones, "Software Defect Origins and Removal Methods," https://insights.cermacademy.com/6-software-defect-origins-and-removal-methods-c-capers-jones-technologyrisk/, Tech. Rep., 2012.

[77] W. W. Royce, "Managing the Development of Large Software Systems," in *Technical Papers of Western Electronic Show and Convention*, ser. WesCon, Aug. 1970, pp. 1–9.

[78] Honourable Mr Justice Sheen, "mv HERALD OF FREE ENTERPRISE — Formal Investigation," Department of Transport, Tech. Rep., 1987. [Online]. Available: https://assets.publishing.service.gov.uk/media/54c1704ce5274a15b6000025/FormalInvestigation_HeraldofFreeEnterprise-MSA1894.pdf

[79] Committee on Transportation and Infrastructure, USA, "Final Committee Report: The Design, Development & Certification of the Boeing 737 MAX," 2020.

[80] D. Daniels, "The Boeing 737 MAX Accidents," in *Proceedings of the 28th Safety-Critical Systems Symposium*. Safety-Critical Systems Club, 2020, pp. 1–21.

[81] J. Simon, *Report of the R.101 Inquiry*. The Stationery Office, 1931.

Index

accidental system, 12
AcciMap, 48, 49
    influence on STAMP, 50
Active Traffic Management, 15, 56
    example, 63
AFD, 45
agile methodology, 68
AHP, 8, 56
Air Traffic Control, *see* ATC
ALARP, example, 64
Analytic Hierarchy Process, *see* AHP
Anand, Nippin, 60
artificial intelligence, 44
Ashby's Law, 45
assurance case, 41–42
assurance requirement
    definition, 5
assurance wrapper, 21, 22, 26
    definition, 5
ATC, 6, 15
    abstractions, 7
autonomous system, 44

Büchi Automaton, 54
Banham, Dave, 77
Barrett, Stephen, 77
barrier analysis, 45
Beauvais, 73
behaviour-driven development, 55
Boeing, 29, 72–73, 75, 76
Bull, Stephen, 77, 78

Canadian Nuclear Safety Commission, 7
CAST, 50, 60
Catmur, James, 77, 78
cause-consequence, 69, 70
Clapham Junction, 74–76
cloud-based software, 12
CONOPS, 21
contingency mode, definition, 6
Costa Concordia, 61
crash-only, 12
CrowdStrike, 6, 8, 18, 72, 75
CTM, 45
CUF, definition, 6
Cynefin, 6

DBN, 45
degraded mode, definition, 6
Dhillon, Narinder, 77
digital twin, 50
Discrete-Event Simulation, 52–53, 69, 70
DynaFRAM, 45
dynamic testing, 13

ECF, 45
emergent behaviour, 11, 51
    definition, 6
    weak, 6
ETA, 13, 69, 70
Event-B, 45, 55

Falcon, CrowdStrike, 6
FMEA, 13, 69, 70
FMECA, 69, 70
FMV, 50
formal analysis, 53–54
formal model checking, 55
FPRSA-R, 6
FRAM, 45, 49–50, 53–55, 60
    as digital twin, 50
    Model Visualiser, 50
FRET, 48
FTA, 13, 69, 70

GAMAB, example, 64
GASP, 52
genetic algorithm, with DES, 53
graceful degradation, 12
graceful extensibility, 18
GSPN, 69, 70

Hampton, Paul, 77
Hargis, Sal, 77
harm, definition, 6
Harney, Louise, 77, 78
Harsanyi, Teiva, 10
HAZOP, 27–29, 47
Herald of Free Enterprise, 71–72, 75, 76
Hobbs, Chris, 77
Hollnagel, Erik, 49
Horizon, 17

IBM, 53
    and GPSS, 52
ICD, 21
IEC 61882, 47
IEC 62740, 45
IEC 61508, 69
IET, 1
INCOSE, 1, 10
Intelligent Transport System, 15
ISO 21448, 7, 22, 23
ISO/IEC Guide 51:2014, 6
ISO/PAS 8800, 44
ISO 26262, definition of “harm”, 6, 70

Jones, Capers, 68

LAAC, 7, 17, 19, 23, 28
    definition, 17
    example, 64
Ladkin, Peter Bernard, 78
Leading Indicator, 65
    definition, 7
Level of abstraction, 18, 21, 26, 32–38, 41, 67
    definition, 7
Leveson, Nancy, 50
Lewis, George, 6
LTL, 48

machine learning, 12, 44
Markov model, 69, 70
mashup methodology, 68
MCAS, 73
MES, 45
Monte-Carlo Simulation, 52
MORT, 45
MTL, 48

NASA, 56
    FRETish language, 48
NASA, space walk incident, 51
National Physical Laboratory, 74
normalisation of deviation, 75
nuclear power, 55

OHA, 47
Ontological Hazard Analysis, *see* OHA
operational state, 20, 25
Oxted, railway accident, 75

Paladugu, Abhinay, 77
Parsons, Mike, 77, 78
PICASSOS, 53
Post, Abigail, 77
PR-OWL, 47

Queenstown Road, railway accident, 75

R.101, 29, 73–76
RAE, 1, 11
RAG, 45
Rao, Arun Chakrapani, 77
Rasmussen, Jens, 43
RBD, 69, 70
Reason, James, 45
Resilience
    Operational Models, 56–58
RFC 2119, 16
RFC 8174, 16
risk-aware system, 44, 54
Rodin, 45, 54, 55
Royce, Winston, 68

Saaty, Thomas, 56
Safety Baseline, definition, 7
Safety Case, 41
    definition, 7
    dynamic, 44, 52
Safety Culture, 52, 75, 76
Safety Target, 13, 20, 25
    definition, 7
SaRS, 1
SCAD, 8, 17, 51–52, 60
Scenario Types, 30–31
SCRG, 65
SCSC, 1
SCSWG, 1, 77
service-level agreement, 11
Simscript, 52
Simulink, 53
Slater, David, 50, 77
Smart Motorway, 15, 63–67
    assurance case study, 42
SOL, 54–55
SOL-VE, 54
SOTIF, 22, 23

definition, 7
SPI, 7, 24, 29, 42, 60, 61, 65
definition, 7
lagging, 42
leading, 42
SPIN, 48, 54, 55
spiral methodology, 68
SSS, 1
stakeholder, 17, 18, 20, 23, 25, 28
definition, 7
STAMP, 47, 50–51, 54, 55
STEP, 45
STPA, 45, 50
Swiss Cheese Model, 45
SysML, 53

temporal logic, 54
time-actor diagram, 54
TLA+, 48, 54

UL 4600, 44
definition of loss, 7

V model, 68
V2I, 63
V2V, 63
VDM, 53
Vehicle-to-Infrastructure Communications, *see* V2I
Vehicle-to-Vehicle Communications, *see* V2V

waterfall methodology, 68
Waterloo Institute for Complexity & Innovation, *see* WICI
WBA, 45
WICI, 1, 10
Wikipedia, 10
Windows, Microsoft, 6
WMC, 8, 55

York, University of, 10



SCSC

For Everyone Working in System Safety