



Guidance on the Through-Life Assurance of Complex Systems

SCSC-200

Version 1.0

SCSC Safer Complex Systems Working Group

28th January 2025

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.

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, reaching consensus on recommended approaches.

The working group collaborates with other bodies working in the area of complex systems, including INCOSE, WICI, IET and RAE. The working group also 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 C on page 50.

Contents

1 Introduction **5**
- 1.1 This document . . . 5
- 1.2 Readership . . . 5
- 1.3 The need for this guidance . . . 5
- 1.4 Scope . . . 6
- 1.5 Terminology . . . 6
- 1.6 Using this guidance . . . 7

2 Characteristics of Complex Systems **8**
- 2.1 Introduction . . . 8
- 2.2 Complex System Scenarios . . . 8
- 2.3 Characteristics . . . 9
  - 2.3.1 System-level characteristics . . . 10
  - 2.3.2 Design & implementation characteristics . . . 11
  - 2.3.3 Verification characteristics . . . 12
  - 2.3.4 Assurance characteristics . . . 12
- 2.4 Complex system examples . . . 13
  - 2.4.1 A hospital . . . 13
  - 2.4.2 Air Traffic Control (ATC) . . . 13

3 Assurance Principles & Objectives **15**
- 3.1 Assurance factors . . . 15
- 3.2 Levels of assurance . . . 15
- 3.3 Assurance principles . . . 16
- 3.4 Mapping Principles to LAACs . . . 21

4 Mapping of Scenario Types to Principles & Objectives **25**

5 Applicable Analysis Techniques **34**
- 5.1 Hazard & Risk Identification . . . 34
  - 5.1.1 Hazard and Operability Analysis (HAZOP) . . . 34
  - 5.1.2 Ontological Hazard Analysis (OHA) . . . 35
- 5.2 System Analysis Techniques . . . 36
  - 5.2.1 Functional Resonance Analysis Method (FRAM) . . . 36
  - 5.2.2 System-Theoretic Accident Model and Processes (STAMP) . . . 37
  - 5.2.3 Discrete-Event Simulation (DES) . . . 38
  - 5.2.4 Formal Model Checking . . . 39
  - 5.2.5 Safety Through Organizational Learning (SOL) . . . 40
  - 5.2.6 Combining STAMP or FRAM with Other Methodologies . . . 40
  - 5.2.7 Assurance Case . . . 41
  - 5.2.8 Other Techniques . . . 41
- 5.3 System Resilience Techniques . . . 41

6 Further Work **42**
6.1 Questions regarding the scope of this guidance . . . 42
6.1.1 What constitutes harm? . . . 42
6.1.2 Malicious actions . . . 42
6.1.3 Responsible Person . . . 43
6.2 Research Topics . . . 43
6.2.1 Risk-Aware Systems . . . 43
6.2.2 Forming the System View before Deployment . . . 43
6.2.3 Combining FRAM/STPA with (Semi-)Formal Model Checking . . . 43
6.2.4 The Rôle of the Safety Case . . . 43
6.2.5 AcciMaps . . . 43
6.2.6 Other Analysis Techniques . . . 44
6.2.7 Resilience Techniques . . . 44

A Limitations of Traditional Approaches **45**
A.1 General Observations . . . 45
A.2 Traditional Assurance Methodologies . . . 45
A.3 Component-Based Analyses . . . 46

B Sample Incidents in Complex Systems **47**
B.1 Introduction . . . 47
B.2 The Herald of Free Enterprise . . . 47
B.3 CrowdStrike Software Update . . . 48
B.4 Boeing 737 Max 8 MCAS . . . 48
B.5 The R.101 . . . 49

C Preparation of this Document **50**

D Document History **51**

Chapter 1

Introduction

Contents

1.1 This document 5
1.2 Readership 5
1.3 The need for this guidance 5
1.4 Scope 6
1.5 Terminology 6
1.6 Using this guidance 7

1.1 This document

This document is the first draft of guidance being prepared by the Safer Complex Systems Working Group. It represents the point reached at the end of the Working Group's first year of operation and more guidance will be added during 2025. Chapter 6 lists some of the topics on which the Working Group will be concentrating in 2025.

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 A on page 45 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 are used in this document as below:

- **Assurance Position.** The current status of the safety artefacts, 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.

- **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 CUF systems on computers running Microsoft Windows were brought down. This particularly affected CUF 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 can be explained once it has occurred. In the literature, this is known as "weak emergence". Examples include 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, or the structure of a flock of starlings in flight. 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.

- **Level of Abstraction.** A complex system can be considered at a hierarchy of levels of abstraction. Consider the 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 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.

- **LAAC.** Levels of Assurance Addressing Complexity. See Section 3.2 on page 15 for a definition of the levels.

- **Safety Case.** Within this guidance, 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 [42]).

- **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 [22] 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.

- **System of Systems.** A system with many parts, some of which may be other systems or services. Reference [32] defines this term as *"A collection of component systems that can exist independently and have different owners who may have conflicting objectives."* In general, the 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 map the characteristics of a complex system given in Chapter 2 to your system: do they 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.

You could then use table 2.1 in Section 2.2 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? With this information, Chapter 4 lists the assurance principles that you might apply. More information about each assurance principle is given in Chapter 3.

Chapter 5 then provides a selection of tools and techniques that you might apply to produce those artefacts.

Chapter 2

Characteristics of Complex Systems

Contents

2.1 Introduction . . . 8
2.2 Complex System Scenarios . . . 8
2.3 Characteristics . . . 9
2.3.1 System-level characteristics . . . 10
2.3.2 Design & implementation characteristics . . . 11
2.3.3 Verification characteristics . . . 12
2.3.4 Assurance characteristics . . . 12
2.4 Complex system examples . . . 13
2.4.1 A hospital . . . 13
2.4.2 Air Traffic Control (ATC) . . . 13

2.1 Introduction

It is difficult to define a “complex” system as the term is used in this guidance, but in practice, it may be effective to use the approaches listed in this document for any system, even if it doesn’t meet a definition of being “complex”. This is termed “treat as complex” (TaC) in the remainder of this guidance.

Complex systems do have common characteristics and some of those characteristics are listed in Section 2.3 below, based in part on the list in reference [39].

2.2 Complex System Scenarios

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

The levels of necessary assurance given in Chapter 3 depend on the Scenario Type for which they are being defined and these Scenario Types are also used in subsequent chapters to indicate the relevance of different tools and techniques. Table 2.1 describes each Scenario.

Table 2.1: 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: New Build	A complex system is to be designed and implemented from scratch, using few existing subsystems.
S-2: Assembly	A new complex system is to be implemented, primarily using 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 full system is recognised as being safety-affecting.
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. Design and safety-analysis documentation for the entire system before restructuring may exist, but probably doesn't as the system may have grown organically. Design documentation may exist for some of the subsystems.
S-4: Assure	An assurance position is to 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-5: New Use	An existing complex system, which was not safety-affecting, is to be deployed in a new context or jurisdiction where it will be safety-affecting. This is similar to Scenario Type 4, but with a different driver.
S-6: Degraded	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 and position need to be created.
S-7: Investigation	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 in the future.

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

2.3 Characteristics

This section describes a system's characteristics which can be used to recognise that it is complex by considering it at the system level (Section 2.3.1). Sections 2.3.2 and 2.3.3 list some of the resulting characteristics during design and verification.

A survey of definitions of "complex system" from sources ranging from the Wikipedia to the Waterloo Institute for Complexity & Innovation 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 [32], 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."*
- **"whole/part interactions"**: reference [12] 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 disproportionate change in the system's behaviour. Reference [32], 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 [32], 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."*

Reference [40] from the International Council on Systems Engineering (INCOSE) includes those characteristics and adds:

- **"multi-scale behaviour"**: the system's structure exists on many scales and characteristics are not reducible to only one level of description.
- **"self-organisation"**: the system grows without explicit control, and is driven by multiple, locally operating, socio-technical processes, usually involving adaptation.
- **"open boundaries"**: the system cannot be analysed by consideration of elements within a defined boundary.
- **"networks of causality"**: intricate networks of evolving cause-effect relationships lead to subtle bugs and surprising system dynamics.

Other characteristics that appear in the literature describing complex systems include randomness, collective dynamics, hierarchic, 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.

2.3.1 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-3 to S-7 as defined in Table 2.1 to determine whether the existing system is "complex". Note that any system may be treated as "complex" and explored using the techniques in this document.

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.

They are "digital, socio-technical" systems.

SF3: involve many interactions, both between 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. In particular, 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.

Other system-level characteristics

A complex system may:

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

S2: be subject to scope creep which may be poorly-controlled.

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 different processes, methods, cultures and working languages.

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 be sure that all the assumptions of its specification continue to be fulfilled.

S8: be geographically (possibly internationally) distributed.

S9: Have requirements which may be obsolete, ambiguous, incomplete or inconsistent.

2.3.2 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 2.1), an engineer substantially modifying an existing one (Scenario Types S-3 and S-5) and an assurance engineer analysing the system (Scenario Type S-4).

Fundamental design & implementation characteristics

A complex system may:

DF1: have functions to support system resilience, to 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 so not maintained, (ii) commercial or design constraints may determine they are superfluous, and also (iii) the understanding of the need for the 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 and non-deterministic behaviours.

DF5: have initialisation, configuration and parameterisation data that are difficult to produce and coördinate between elements, making configuration management 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 activities or documentation.

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

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

D4: involve time-consuming and difficult integration activities.

2.3.3 Verification characteristics

The characteristics listed in this section are of concern to an engineer analysing the behaviour of an existing system and (possibly retrospectively) producing an assurance of adequate safety (Scenario Types S-4 and S-7 in table 2.1).

Because of their complex nature, these systems may:

V1: have weak verification evidence because current verification tools that are effective for small or non-complex systems do not work effectively on complex systems.

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 part 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 typical scenarios if, as described in point V1:, thorough dynamic testing is not possible.

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-based analysis (FTA, ETA, FMEA, etc.) is inappropriate for complex systems: see Appendix A.3 on page 46. Chapter 5 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 which have to 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.

2.3.4 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 2.1.

Fundamental assurance characteristics

The engineer producing the safety assurance for a complex system will:

CF1: have to show that complexity properties such as emergence, non-linear effects, and whole/part interactions do not lead to unsafe behaviours.

CF2: have to show that the resilience of the system is sufficient.

CF3: have to provide evidence of coverage including technical, services and human elements such as skills, procedures and process.

The engineer producing the assurance argument will need to rely on non-traditional techniques as outlined in Chapter 5.

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 evolve independently and have 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.

2.4 Complex system examples

2.4.1 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, 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 by both the fire and the water firefighters used to douse it.

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.

2.4.2 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 [14] 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 [3] 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 [3] 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 Section 5.2.3), that paper identifies potential emergent behaviour in the coördination of aircraft crossing an active runway.

Chapter 3

Assurance Principles & Objectives

Contents

3.1 Assurance factors 15
3.2 Levels of assurance 15
3.3 Assurance principles 16
3.4 Mapping Principles to LAACs 21

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 [4]) as clarified in RFC 8174 (reference [28]).

3.1 Assurance factors

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

- Why is the assurance analysis being carried out? The scenarios considered in this guidance are listed in table 2.1.
- How much harm could the system do? Levels of harm are considered in Section 3.2.
- What types of analysis are applicable to complex systems? Section 3.3, and in particular tables 3.1 to 3.7, cover this.
- What types of analysis are recommended for complex systems with different levels of potential harm? Section 3.4 and tables 3.8 to 3.14 address this question.
- For a particular scenario (as listed in table 2.1), what types of analysis are recommended? This is covered in Chapter 4 and, in particular, in tables 4.1 to 4.8.

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 that “harm” to a stakeholder could include physical injury, mental stress, financial stress, loss of reputation or other forms of injury. See Section 6.1.1 on page 42 for discussion of the types of harm and loss that should be considered.

In this guidance, five LAACs describing a complex system are considered:

- 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.

It is sometimes difficult at the beginning of a development to assess the required assurance level accurately so it must be reässessed at defined stages of the system lifecycle and when changes to the system or its environment are planned.

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 [6], 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 be considered LAAC-4.

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.

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-7 in table 2.1), it is assumed that the system requirements would normally have to be defined retrospectively from an 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 abstractions may have a different associated vocabulary and different focus and it is important that the assurance requirements recognise this.

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. Where a requirement cannot be satisfied, then the residual risk must be identified and this made clear to all stakeholders.

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.

5. P-5: The SCS SHALL be resilient to unanticipated and unintended situations that can arise in its environment, including accidental or deliberate misuse.

   For many complex systems, the precise boundary between the system and its external environment is difficult to define: it's porous and varies according to the abstraction by 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, and so the system must be able to handle unanticipated environmental 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.

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, those operating it and those affected by it.
1.e	Assurance requirements for desirable behaviours (including resilience) of the SCS SHALL be defined.	This includes data requirements (including integrity-, configuration- and security-related) as these may affect system behaviour.
1.f	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.g	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, etc.
1.h	Historical accidents and incidents related to the SCS SHOULD be assessed and any relevant recommendations considered.	

Table 3.1: Identifying the Assurance Requirements

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

   Note that the life of complex systems may be long. A hospital is a complex system (see Section 2.4.1) and 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. Within those tables the phrase "SHALL be considered" includes "SHALL be identified, analysed, assessed and managed".

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. 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.	
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 wrappers SHALL be identified.	Assurance wrappers are created to make good any known assurance shortfalls of subsystems or subservices: see the definition on page 6. Where wrappers are not provided or are insufficient, then restrictions and limitations MUST be devised and documented to mitigate the associated risks.

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 produce assurance.	See Chapter 5 for information about appropriate tools.
3.c	Assurance derived from the design and the design decisions SHALL be used where applicable.	For example, the design decision to incorporate a version of a software module 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	Assurance evidence SHOULD include evidence acquired from 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.	
4.c	Emergent behaviours of the SCS SHALL be considered.	See reference [22] regarding SOTIF.
4.d	Behaviours due to design decisions SHALL be considered.	
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 misuse cases SHALL be considered.	

Table 3.4: Managing Unintended & Unanticipated Behaviours

Id	Principle	Notes
5	**P-5: The SCS SHALL be resilient to unanticipated and unintended situations that can arise in its environment, including accidental or deliberate misuse.**	
	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.	
5.d	Emergent behaviours due to the interaction of the SCS with its environment SHALL be considered.	
5.e	Degraded and contingency modes of the SCS SHALL maintain a defined subset of these objectives to a specific level.	
5.f	Behaviours of the SCS due to misuse within its environment SHALL be considered.	

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	Levels of Assurance Addressing Complexity (LAACs) SHALL be established, based on the level of risk that the SCS presents to its stakeholders.	See Section 3.2 for a description of LAACs.
6.b	LAACs SHALL be assigned at each level of abstraction of the SCS with two-way tracing through the 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.	
6.e	Residual risks SHALL be documented.	

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, re-deployment and re-purposing.**	
	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.	
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	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	Assurance requirements for desirable behaviours (including resilience) of the SCS SHALL be defined.	Requirements	○	●	●	●
1.f	Assurance requirements to mitigate undesirable behaviours of the SCS SHALL be defined.	Requirements	○	●	●	●
1.g	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.h	Historical accidents and incidents related to the SCS SHOULD be assessed and any relevant recommendations considered.	Accident and incident reports	○	●	●	●

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 wrappers SHALL be identified.	Wrappers report	○	●	●	●

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 produce assurance.	Tool qualification reports	○	●	●	●
3.c	Assurance derived from the design and the design decisions SHALL be used where applicable.	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	Assurance evidence SHOULD include evidence acquired from field usage where this is available.	Service history; field reports; bug and incident reports	○	○	●	●

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 misuse cases SHALL be considered.	HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●

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 situations that can arise in its environment, including accidental or deliberate misuse.**					
	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	Degraded and contingency modes of the SCS SHALL maintain a defined subset of these objectives to a specific level.	HAZOP; analyses; mode definitions; mode transitions	○	●	●	●
5.f	Behaviours of the SCS due to misuse within its environment SHALL be considered.	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	Levels of Assurance Addressing Complexity (LAACs) SHALL be established, based on the level of risk that the SCS presents to its stakeholders.	HAZOP; risk register; LAAC report; hazard log	●	●	●	●
6.b	LAACs SHALL be assigned at each level of abstraction of the SCS with two-way tracing through the 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.	LAAC report; HAZOP; analyses; simulations; risk register; reviews; independent audits	●	●	●	●
6.e	Residual risks SHALL be documented.	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, re-deployment and re-purposing.**					
	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; independent 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	Lessons learned from successful operation, failures and incidents SHOULD be reviewed.	Incidence and accident reports	○	●	●	●

Table 3.14: Principle 7: Assurance Evidence

Chapter 4

Mapping of Scenario Types to Principles & Objectives

Section 2.2 on page 8 describes a number of scenarios for which analysis might be carried out on a complex system. Some of the Principles and Objectives given in Chapter 3 are more relevant to some Scenario Types than to others. Tables 4.1 to 4.8 below map recommended Principles and Objectives to the Scenario Types.

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-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 situations that can arise in its environment, including accidental or deliberate misuse.	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 4.1: Scenario "S-0: Check"

Scenario	Notes
S-1: New Build: 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 [22]) and UL 4600 (reference [42]) may be useful for this.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, re-deployment and re-purposing.	
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 situations that can arise in its environment, including accidental or deliberate misuse.	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 4.2: Scenario "S-1: New Build"

Scenario	**Notes**
S-2: Assembly: A new complex system is to be implemented, primarily using 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.
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 situations that can arise in its environment, including accidental or deliberate misuse.	
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, re-deployment and re-purposing.	
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 4.3: Scenario “S-2: Assembly”

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 [22]) and UL 4600 (reference [42]) may be useful for this.
P-7: All principles SHALL be established and maintained throughout the lifetime of the SCS, through all changes, reuse, re-deployment and re-purposing.	
Recommended Principles	
P-5: The SCS SHALL be resilient to unanticipated and unintended situations that can arise in its environment, including accidental or deliberate misuse.	
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-3: Restructure”

Scenario	**Notes**
S-4: Assure: An assurance position is to 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 situations that can arise in its environment, including accidental or deliberate misuse.	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, re-deployment and re-purposing.	
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 4.5: Scenario “S-4: Assure”

Scenario	Notes
S-5: New Use: An existing complex system, which was not safety-affecting, is to be deployed in a new context or jurisdiction where it will be safety-affecting.	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 situations that can arise in its environment, including accidental or deliberate misuse.	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, re-deployment and re-purposing.	
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 4.6: Scenario “S-5: New Use”

Scenario	**Notes**
S-6: Degraded: 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 situations that can arise in its environment, including accidental or deliberate misuse.	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 4.7: Scenario “S-6: Degraded”

Scenario	Notes
S-7: Investigation: 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 situations that can arise in its environment, including accidental or deliberate misuse.	It could be a lack of such resilience that can now be identified.

Table 4.8: Scenario “S-7: Investigation”

Chapter 5

Applicable Analysis Techniques

Contents

5.1 Hazard & Risk Identification . . . 34
5.1.1 Hazard and Operability Analysis (HAZOP) . . . 34
5.1.2 Ontological Hazard Analysis (OHA) . . . 35
5.2 System Analysis Techniques . . . 36
5.2.1 Functional Resonance Analysis Method (FRAM) . . . 36
5.2.2 System-Theoretic Accident Model and Processes (STAMP) . . . 37
5.2.3 Discrete-Event Simulation (DES) . . . 38
5.2.4 Formal Model Checking . . . 39
5.2.5 Safety Through Organizational Learning (SOL) . . . 40
5.2.6 Combining STAMP or FRAM with Other Methodologies . . . 40
5.2.7 Assurance Case . . . 41
5.2.8 Other Techniques . . . 41
5.3 System Resilience Techniques . . . 41

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 A, but, for most complex systems, it will be found that 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.

5.1 Hazard & Risk Identification

5.1.1 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 being examined should be used as long as they are identified ahead of the HAZOP session. The guide words in table 5.1 may be relevant for complex systems and could be added to those suggested in IEC 61882.

The output from the HAZOP process is a structured list of identified deviations from the anticipated system environment, the possible causes and consequences of those deviations, the actions needed to mitigate the consequences and the name of the person taking responsibility for the actions.

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

Table 5.1: Possible Additional HAZOP Guide Words

5.1.2 Ontological Hazard Analysis (OHA)

While the IEC 61882 HAZOP process described above can generate useful lists of hazards and risks, it cannot ensure that all hazards and risks have been identified: it can't 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 [44], [45] and [1]. 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 [1] proposes using the Probabilistic Web Ontology Language (PR-OWL) to define the ontology for modelling the knowledge about a system in a particular industry (see figure 5.1) 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 [1] describes the application of this process to a number of safety-affecting incidents and lists the open-source tools required for a particular analysis.

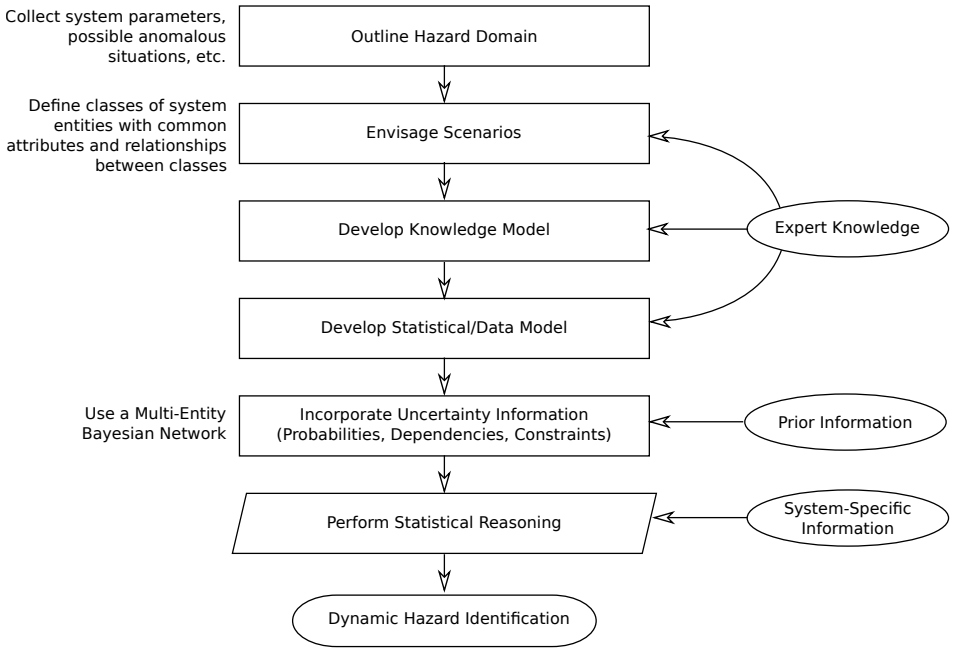


Figure 5.1: Hazard Identification Methodology (Adapted from reference [1])

A suitable ontology, combined with a system description, can also be used 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 [25]. This provides a structure for applying formal methods (see Section 5.2.4 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 [25] uses the TLA$^+$ tool (reference [27]) to provide the correctness proof of the resulting argument, but the Rodin tool (reference [5]) with its emphasis on abstraction refinement may be more appropriate.

An example of the OHA process being applied is given in reference [26].

The authors of reference [1] point out that, in a quantitative model of this sort, it is challenging to capture numerical parameters associated with human error.

5.2 System Analysis Techniques

The analysis techniques described in this section avoid the inadequacies listed in Chapter A and address the challenges listed in Section A.1. In particular, causal-flow techniques such as AcciMaps are excluded from this chapter because, as reference [43] 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 stopping 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 [43] reports the results of reviewing 476 journal articles on accident investigation, searching for the techniques deployed. The results were that STAMP (Section 5.2.2), FRAM (Section 5.2.1) and AcciMaps accounted respectively for 52%, 20% and 18% of the 302 articles.

5.2.1 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 [16] and is described in his book, reference [17]. 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.

FRAM can be applied both to pre-incident (Scenarios S-1 to S-6 in table 2.1) and post-incident (Scenario S-7) analysis. Reference [7] provides an example of the former, and [37] 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 starts the function? In the example, the input would include a patient arriving in an ambulance.

   (b) What is the function's output: what does it do with the input? In the 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 the 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 is there that 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 the example, the time constraints depend on the condition of the patient: in some cases the patient might be harmed by delay.

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

(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 [18] gives examples of variability in technology, organisations and individuals. In particular it notes that

   > *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. Define the function resonance 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 and an open-source tool, the FRAM Model Visualiser (FMV), is available to edit that notation.

5.2.2 System-Theoretic Accident Model and Processes (STAMP)

The STAMP framework is described in Nancy Leveson's book, reference [30], which builds on work by Jens Rasmussen (reference [38]), 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-6 in table 2.1).

2. Causal Analysis Using System Theory (CAST).

   This is applied after an incident occurs (Scenario S-7).

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 [29] also points out that, in many investigations, it is impossible to identify the "initiating event": what event started the chain?

STAMP views a system as a hierarchy of controlled processes with actuators driven by the controller and sensors feeding back data to the controller. The controller has a "mental model" of how the process it is controlling will react to different control inputs. The controller may be a human, an organisation or a physical device. The actuators and sensors may be physical devices or, more normally, are abstract as in Figure 5.2: society puts pressure on the government to introduce legislation to control, for example, autonmous cars. The government has a (potentially incomplete) model of the autonomous car industry and controls various appropriate regulatory bodies. This continues as the regulatory bodies put restrictions on the companies designing such cars, etc. Feedback occurs when, for example, a company reports that a regulation is inappropriate.

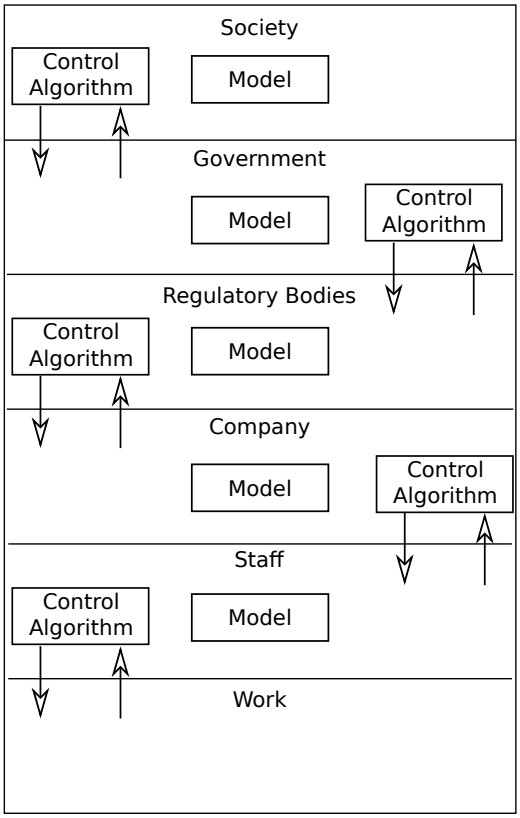


Figure 5.2: An Example STAMP Framework

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:

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

  The issuing of legislation regarding autonomous cars may be slower than the technology is moving, making it out of date 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, the mental model of the autonomous car's abilities may be wrong.

It is the last of these which is most subtle and most common: over time the controller's model may become out of date or otherwise inadequate as the controlled object changes.

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

5.2.3 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 [23].

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 [2] and [3] describe examples of DES being used to perform risk assessments on runway incursions at large airports.

5.2.4 Formal Model Checking

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.

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 [19]) 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 [27]) builds an 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 [5]) 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 [15] 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.

Section 5.2.6 below describes combining the use of formal methods with STAMP and FRAM.

5.2.5 Safety Through Organizational Learning (SOL)

Much of the material in this section is taken from reference [36] which itself references [13].

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.

5.2.6 Combining STAMP or FRAM with Other Methodologies

References [10] and [35] identify gaps between the expected and the obtained result when FRAM or STAMP was applied alone. Reference [33] 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 Section 5.2.4)
- a finite-state machine

- a discrete-event simulation (see Section 5.2.3)

STPA with

- a formal model checker (e.g. Event-B (Rodin): see Section 5.2.4)
- behaviour-driven development (BDD)

See section 6.2.3 on page 43 for additional future work in this area.

5.2.7 Assurance Case

See section 6.2.4 for a consideration of whether an Assurance Case is applicable for complex systems.

5.2.8 Other Techniques

See Section 6.2.6.

5.3 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 6.2.7.

Chapter 6

Further Work

Contents

6.1 Questions regarding the scope of this guidance . . . 42
6.1.1 What constitutes harm? . . . 42
6.1.2 Malicious actions . . . 42
6.1.3 Responsible Person . . . 43
6.2 Research Topics . . . 43
6.2.1 Risk-Aware Systems . . . 43
6.2.2 Forming the System View before Deployment . . . 43
6.2.3 Combining FRAM/STPA with (Semi-)Formal Model Checking . . . 43
6.2.4 The Rôle of the Safety Case . . . 43
6.2.5 AcciMaps . . . 43
6.2.6 Other Analysis Techniques . . . 44
6.2.7 Resilience Techniques . . . 44

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 What constitutes harm?

Section 3.2 on page 15 defines several LAACs based on the "harm" that a complex system may cause. Standards such as ISO 26262:2018 define "harm" as being "physical injury or damage to persons". This definition explicitly excludes harm to the environment (e.g., oil spill at sea) and non-physical harm to persons (e.g., mental harm, financial harm, loss of reputation).

IEC 60601-1 provides a wider definition of "harm" as "Physical injury or damage to the health of people or animals or damage to property or the environment." This definition also excludes mental or financial harm to a person.

In its definitions, UL 4600 is proposing moving away from the concept of "harm", replacing it with the idea of "loss". In reference [24] 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."*

How broad should the definition of harm be in this guidance?

6.1.2 Malicious actions

Section 1.4 on page 6 defines the scope of the guidance in this document. Should this include malicious actions?

Malicious actions include a person from within the complex system performing an unauthorised act or an external hacker breaking into the system and performing such an act.

There are also non-malicious actions which can have the same effect as a malicious act. These include managers within a company putting pressure on engineers to complete a project to a defined timescale when that leads to an unsafe product. There are many examples of this type of behaviour: see, for example, references [41] and [8].

6.1.3 Responsible Person

Section 2.2 on page 8 assumes that someone is responsible for assessing the safety assurance of a complex system. How is this person identified? Should such a person be appointed and have the boundary of her analysis defined?

6.2 Research Topics

6.2.1 Risk-Aware Systems

Section 5.2.4 on page 39 and reference [15] refer to the concept of a "risk-aware system": a system that is aware of the risks it is creating and, using introspection, tries to control those risks.

The concept of a risk-aware system seems to be close to that of 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 [11].

6.2.2 Forming the System View before Deployment

A post-incident report typically contains much more information than analyses performed before an incident occurs. 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.

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

Section 5.2.6 on page 40 outlines some work that has been done to combine FRAM with other techniques. It appears that additional results could be obtained by extending these combinations, particularly STPA or FRAM with a modelling tool.

6.2.4 The Rôle of the Safety Case

The UL 4600 standard (reference [42]) requires that a Safety Case be produced to justify the assurance of an autonomous system and provides in-depth guidance on how that Safety Case should be created. It also provides a list of "pitfalls" to be avoided when creating the Safety Case. This raises the question of whether a Safety Case is a suitable vehicle for justifying the assurance of any complex system.

In particular, would a quantitative Bayesian Safety Case be more applicable than a qualitative GSN one?

ISO/PAS 8800 *("Road Vehicles – Safety and Artificial Intelligence")* lays stress on the production of the Safety Case for machine-learning systems in road vehicles. It also includes a large, qualitative GSN that could be used to justify a claim that the safety requirements are fulfilled.

6.2.5 AcciMaps

Most of the literature dealing with AcciMaps discusses investigating accidents that have happened, not analysing systems before an accident occurs. Given AcciMap's reliance on causality and the emergent (non-causal) behaviour from complex systems, can this technique be applied to pre-incident analysis?

6.2.6 Other Analysis Techniques

Numerous techniques have been proposed for analysing complex systems which have not been included in Chapter 5. These include the Analytic Hierarchy Process (AHP), Anticipatory Failure Determination (AFD), DynaFRAM, Resilience Assessment Grid (RAG), Dynamic Bayesian Network (DBN), and Event-B (as in the Rodin tool).

IEC 62740 (reference [21]) adds Barrier analysis, 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).

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

6.2.7 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 likely to be inappropriate for complex systems.

Appendix A

Limitations of Traditional Approaches

Contents

A.1 General Observations 45
A.2 Traditional Assurance Methodologies 45
A.3 Component-Based Analyses 46

A.1 General Observations

Historically, as described in reference [29], accident models viewed accidents as resulting from a causal chain of events, often initiated by the failure or malfunction of a component. Such models work well for losses caused by failures of physical components and for relatively simple systems, but are inadequate for complex systems.

Reference [29] claims that this model is now inadequate for a number of reasons, including:

- the increased complexity of systems and the tight coupling of subsystems within them. We are designing systems with potential interactions among the components that cannot be thoroughly planned, understood, anticipated, or guarded against.
- the changing nature of accidents: digital systems introduce new failure modes.
- the new types of hazards: hazards are no longer restricted to the result of an uncontrolled and undesired release of energy. Our increasing dependence on information systems is creating the potential for loss of information or incorrect information that can lead to unacceptable physical or financial losses.
- the decreasing tolerance for single accidents. Our new scientific and technological discoveries have not only created new or increased hazards (such as radiation exposure and chemical pollution) but have provided the means to harm increasing numbers of people including an impact on future generations through environmental pollution and genetic damage.
- the more complex relationships between humans and automation.
- the changing regulatory and public views of safety: in our increasingly complex and interrelated societal structure, responsibility for safety is shifting from the individual to companies and governments. Individuals can no longer control the risks around them and are demanding that government assume greater responsibility through laws and various forms of oversight and regulation.

A.2 Traditional Assurance Methodologies

Some of the implicit assumptions underlying the approaches used to justify whether a system is adequately safe date from the 1960s. These assumptions 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, and 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 also now tend to be critical "components" of a system and these bring their own variability in behaviour, level of training, state of mind, experience, etc.

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 and so all possible system behaviours can be foreseen.

   Rebuttal: Complex systems may exhibit "emergent behaviour": unpredicted behaviour caused not by the individual components, but by the complex interactions of those components.[1]

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

   Claim: Many of the international safety standards explicitly claim that only those hazardous situations caused by failure or malfunction need to 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 [29]) have noted that, in most dangerous situations, every component in the system operated exactly as intended.

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.[2] 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.

A.3 Component-Based Analyses

Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA) and Event Tree Analysis (ETA) are based on analysis of a system's components, assume that a dangerous situation occurs only when a component fails or malfunctions and presuppose a causal chain of events. As indicated above, these are not applicable to a complex system.

[1] This is not new: Aristotle said "the whole is something over and above its parts, and not just the sum of them all" and reference [31] 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."* More recently, the concept of emergent behaviour has moved from the physical and biological world to include our complex systems.

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

Appendix B

Sample Incidents in Complex Systems

Contents

B.1 Introduction 47
B.2 The Herald of Free Enterprise 47
B.3 CrowdStrike Software Update 48
B.4 Boeing 737 Max 8 MCAS 48
B.5 The R.101 49

B.1 Introduction

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

B.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 [20] is the official report of this incident and reference [17] contains the outline of a FRAM[1] analysis of the incident. Reference [20] 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 of human and technical factors. The ship was designed in such a way that the Captain on the bridge could not see whether the bow doors were closed and had no indicator showing their state. Although there had been requests for such an indicator (section 18 of [20]), 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:

> *Bow and stern door remote indication*

[1] See Section 5.2.1 on page 36.

> *...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 [20] 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 [20]); the ship was significantly overloaded with passengers (section 17 of [20]); there was confusion over crew responsibilities and the assistant bosun, who should have ensured that the bow doors were closed, was asleep when the ship departed (section 10 of [20]).

B.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 properly restart.

As a result, 5,078 scheduled airline flights were cancelled, many banks were unable to service 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"*.

This system illustrates several characteristics of a complex system listed in section 2.3.1 on page 10:

- There were many subsystems: it is estimated that there were 8.5 million subsystems affected.
- It was geographically distributed, affecting systems in Australasia, Asia, Europe, Africa and the Americas.
- Its effects on many industries (hospitals, airlines, banks, etc.) emerged unexpectedly, but can be explained retrospectively.

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

B.4 Boeing 737 Max 8 MCAS

The executive summary of reference [8] begins

> *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 flight characteristics of the 737 and, to avoid airlines having to retrain their pilots, Boeing added a Maneuvering Characteristics Augmentation System (MCAS). The MCAS received signals from two Angle-of-Attack sensors and could automatically move the aircraft's nose-up/down trim.

As described in reference [9], 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 read 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 [8] 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 [9] points out that the MCAS in both cases operated exactly as it was designed to operate), but because of its complex nature.

B.5 The R.101

At 18:30 on 4th October 1930, the R.101, the largest airship in the world, took off from Cardington, UK on her maiden voyage to fly 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 [41] 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. Some months before the flight to India, it was found that the R.101 was not actually lighter than air: it 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 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 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. At the request of the investigation, additional calculations were performed after the accident by the National Physical Laboratory (NPL) and these described the manner in which the accident occurred. There had been no time to perform these calculations before the flight.

The official report (reference [41]) 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."*

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

Appendix C

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):

Arun Chakrapani Rao

Chris Hobbs

Dave Banham

Louise Harney

Mike Parsons

Narinder Dhillon

Stephen Barrett

Stephen Bull

...Lots of others

Appendix D

Document History

Date	Version	Notes
04 Feb 2024	0.1	Metadata version for initial comments
04 Feb 2024	0.2	Incorporating comments from Mike Parsons
23 Mar 2024	0.3	Incorporating comments from Stephen Bull and additional comments from Mike Parsons
05 Apr 2024	0.4	Incorporating comments from Stephen Bull, Peter Bernard Ladkin and Mike Parsons
09 Apr 2024	0.5	Changes to Chapter 2 following meeting between Mike Parsons and Chris Hobbs
12 Apr 2024	0.6	Changes to Chapter 3 following meeting between Mike Parsons, Stephen Bull and Chris Hobbs
19 Apr 2024	0.7	Further changes to Chapter 3 following meeting between Mike Parsons and Chris Hobbs
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 Mike Parsons and Chris Hobbs. 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 Mike Parsons and Chris Hobbs.
13 Oct 2024	0.14	Incorporating changes suggested by Louise Harney
15 Oct 2024	0.15	Incorporating changes suggested by Mike Parsons in section 2.3
12 Nov 2024	0.16	Incorporating changes suggested at WG meeting on 16th October 2024
10 Dec 2024	0.17	Incorporating changes suggested at meetings between Mike Parsons and Chris Hobbs
11 Dec 2024	0.18	Incorporating changes suggested at WG meeting on 11th December 2024
19 Dec 2024	0.19	Incorporating changes suggested by Peter Bernard Ladkin and Mike Parsons by email
10 Jan 2025	0.20	Minor editorial changes
28 Jan 2025	1.0	Issued for SSS'25

Figure D.1: History of this Document

Bibliography

[1] Aziz, A., Ahmed, S., and Khan, F. I. An Ontology-Based Methodology for Hazard Identification and Causation Analysis. *Process Safety and Environmental Protection 123* (2019), 87–98.

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

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

[4] Bradner, S. O. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, Mar. 1997.

[5] Butler, M., and Hallerstede, S. The Rodin formal modelling tool. In *Proceedings of the 2007 International Conference on Formal Methods in Industry* (Swindon, GBR, 2007), FACS-FMI'07, BCS Learning & Development Ltd., p. 2.

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

[7] Clay-Williams, R., Hounsgaard, J., and Hollnagel, E. Where the rubber meets the road: using FRAM to align work-as-imagined with work-as-done when implementing clinical guidelines. *Implementation science : IS 10* (August 2015), 125.

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

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

[10] del Carmen Pardo-Ferreira, M., Martínez-Rojas, M., Salguero-Caparrós, F., and Rubio-Romero, J. C. Evolution of the Functional Resonance Analysis Method (FRAM) through the combination with other methods. *Dirección y Organización* (2019).

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

[12] Estrada, E. What is a complex system, after all? In *Foundations of Science* (2023).

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

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

[15] Gleirscher, M., Calinescu, R., and Woodcock, J. RiskStructures: A design algebra for risk-aware machines. *Form. Asp. Comput. 33*, 4–5 (Aug. 2021), 763–802.

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

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

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

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

[20] Honourable Mr Justice Sheen. mv HERALD OF FREE ENTERPRISE — Formal Investigation. `https://assets.publishing.service.gov.uk/media/54c1704ce5274a15b6000025/FormalInvestigation_HeraldofFreeEnterprise-MSA1894.pdf`, 1987.

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

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

[23] 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.

[24] Koopman, P., and Widen, W. Redefining Safety for Autonomous Vehicles. In *SafeComp 2024* (2024).

[25] Ladkin, P. B. Ontological Analysis. *Safety Systems 14*, 3 (2005).

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

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

[28] Leiba, B. Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. RFC 8174, May 2017.

[29] Leveson, N. A new accident model for engineering safer systems. *Safety Science 42*, 4 (2004), 237–270.

[30] Leveson, N. G. *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press, 01 2012.

[31] Lewes, G. *Problems of Life and Mind*. Kessinger publishing's rare reprints. Kessinger Publishing, 1879.

[32] McDermid, J. A., Burton, S., Garnett, P., and Weaver, R. *Safer Complex Systems: An Initial Framework*. 2021.

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

[34] Parisaca Vargas, A., and Bloomfield, R. 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* (Setubal, PRT, 2015), IC3K 2015, SCITEPRESS - Science and Technology Publications, Lda, p. 316–327.

[35] Patriarca, R., Chatzimichailidou, M., Karanikas, N., and Gravio, G. D. The past and present of System-Theoretic Accident Model And Processes (STAMP) and its associated techniques: A scoping review. *Safety Science* (2022).

[36] 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.

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

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

[39] Royal Academy of Engineering. Safer Complex Systems: An Initial Framework. Tech. rep., Royal Academy of Engineering, 2020.

[40] Sheard, S., Cook, S., Honour, E., Hybertson, D., Krupa, J., McEver, J., McKinney, D., Ondrus, P., Ryan, A., Scheurer, R., Singer, J., Sparber, J., and White, B. INCOSE: A Complexity Primer for Systems Engineers.

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

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

[43] Underwood, P., and Waterson, P. 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.

[44] Wu, C.-G., Xu, X., Zhang, B., and Na, Y.-L. Domain Ontology for Scenario-Based Hazard Evaluation. *Safety Science 60* (2013), 21–34.

[45] Zhou, J., Hänninen, K., Lundqvist, K., and Provenzano, L. 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.

Index

accidental system, 11
AcciMap, 36
    influence on STAMP, 37
AFD, 44
AHP, 44
assurance requirement
    definition, 6
assurance wrapper, 18, 22
    definition, 6
ATC, 6, 13
    abstractions, 7

Büchi Automaton, 40
Banham, Dave, 50
Barrett, Stephen, 50
barrier analysis, 44
behaviour-driven development, 41
Boeing, 48
Bull, Stephen, 50, 51

CAST, 37
cloud-based software, 11
contingency mode
    definition, 6
CrowdStrike, 6
CTM, 44
CUF, definition, 6

DBN, 44
degraded mode
    definition, 6
Dhillon, Narinder, 50
Discrete-Event Simulation, 38–39
DynaFRAM, 44
dynamic testing, 12

ECF, 44
emergent behaviour, 10
    definition, 7
    weak, 7
ETA, 12, 46
Event-B, 41, 44

Falcon, CrowdStrike, 6
FMEA, 12, 46
FMV, 37
formal model checking, 39–41
FPRSA-R, 6
FRAM, 36–38, 40, 43
    Model Visualiser, 37
FTA, 12, 46

GASP, 38
genetic algorithm, with DES, 39
graceful degradation, 11
GSN, 43

Harney, Louise, 50, 51
HAZOP, 23, 24, 34
Hobbs, Chris, 50
Hollnagel, Erik, 36
Horizon, 16

IBM, 39
    and GPSS, 38
IEC 61882, 34, 35
IEC 62740, 44
IEC 60601, 42
IET, 2
INCOSE, 2, 10
ISO 21448, 7, 19
ISO/PAS 8800, 43

LAAC, 20, 24
    definition, 7
Ladkin, Peter Bernard, 51
Level of abstraction, 16, 18, 22, 26–31
    defn, 7
Leveson, Nancy, 37
Lewis, George, 46

machine learning, 11
MCAS, 48
MES, 44
Monte-Carlo Simulation, 38
MORT, 44

National Physical Laboratory, 49
nuclear power, 40

OHA, 35–36

Ontological Hazard Analysis, *see* OHA
operational state, 17, 21

Parsons, Mike, 50, 51
PR-OWL, 35

R.101, 49
RAE, 2
RAG, 44
Rao, Arun Chakrapani, 50
Reason, James, 44
RFC 2119, 15
RFC 8174, 15
risk-aware system, 40, 43
Rodin, 41, 44

Safety Case
    Bayesian, 43
    definition, 7
    dynamic, 43
SaRS, 2
Scenario Types, 8–9
SCSC, 2
SCSWG, 2, 50
service-level agreement, 10
Simscript, 38
SOL, 40
SOL-VE, 40
SOTIF
    definition, 7
SPIN, 40
SSS, 2
stakeholder
    definition, 7
STAMP, 37–38, 40
STEP, 44
STPA, 37
Swiss Cheese Model, 44

TaC, 8
temporal logic, 39
time-actor diagram, 40

UL 4600, 43
    defn of loss, 42

VDM, 39

Waterloo Institute for Complexity & Innovation, 9
WBA, 44
WICI, 2
Windows, Microsoft, 6

York, University, 9