

Systematic Analysis of Critical Systems Certification

A Thesis

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science (Computer Science)

by

Panayiotis Steele

August 2013

Abstract

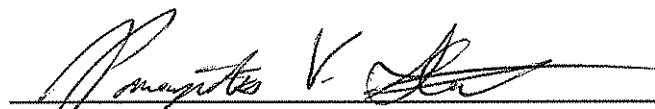
Any given regulatory agency, such as the US Food and Drug Administration, strives to protect the public interest through certification of systems in the agency's purview. Modern safety-critical systems have significant software components. Due to the deterministic nature of software failures, certifiers cannot apply traditional statistical risk assessment methods. Thus, certifiers struggle to assess whether safety-critical systems are adequately safe. Current practice for certification revolves around two different types of standards: (a) prescriptive and (b) goal-based. Both types of standards exhibit significant faults; these faults can lead to the regulatory approval of systems that are *not* adequately safe.

To facilitate analysis and repair of certification faults, this work presents the **filter model of certification**. The filter model views any given certification mechanism as a safety-critical system in itself. This insight allows certifiers to apply systematic safety engineering to their certification mechanisms.

The filter model is evaluated for feasibility through a case study. First, common hazard analysis techniques are adapted and applied to a specimen certification mechanism, the Graydon-Knight-Green mechanism (GKG). The results of hazard analysis are used to adjudicate certification faults. Second, GKG is used in hypothetical certification of a safety-critical system, the Diabetes Advanced Information System (DAIS). The results of the hypothetical certification are used to inform the adaptation and application of common fault mitigation techniques to GKG.

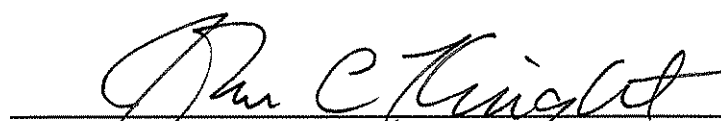
Approval Sheet

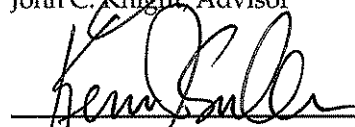
This thesis is submitted in partial fulfillment of the requirements for the degree of
Master of Science (Computer Science)

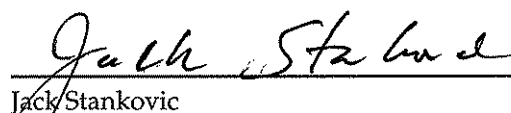


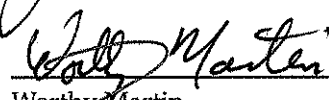
Panayiotis Steele

This thesis has been read and approved by the Examining Committee:



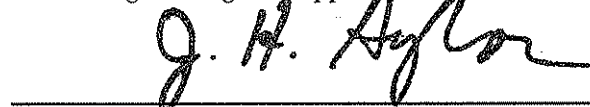
John C. Knight, Advisor

Kevin Sullivan, Chair

Jack Stankovic

Worthy Martin

Accepted for the School of Engineering and Applied Science:



James Aylor, Dean, School of Engineering and Applied Science

December 2013

*“I will bless the LORD at all times: His praise shall be continually in my mouth.”
(Psalms 33:10)*

Acknowledgments

The first thanks are always due to the LORD, as is the lion's share of gratitude. I have been blessed beyond measure and am still learning, day by day, how best to thank Him for all that He has done for me.

Thanks also go to my family, who have tirelessly supported and encouraged me. Special thanks go to the Orthodox Christian Fellowship at U.Va.; in particular, Eric and Larry have been the brothers I never had.

Last but certainly not least, I am eternally grateful to my advisor, John Knight, for guiding me through the evolution of this work to the very end, and teaching me the value of patience, perseverance, and being thorough.

Contents

Contents	v
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 The Certification Problem	3
1.2 The Filter Model	5
1.3 Thesis Statement	8
1.4 Thesis Outline	9
2 Certification: Goals and Requirements	11
2.1 Goals	11
2.1.1 Sub-Goals	12
2.1.2 Certification and Residual Risk	14
2.2 Current Practices	14
2.2.1 Prescriptive Standards	15
2.2.2 Goal-Based Standards	16
2.3 Summary	19
3 The Filter Model of Certification	20
3.1 The Filter Model	20
3.1.1 Inputs and Outputs	20
3.1.2 General View of Certification	22
3.2 System Safety Analysis	26
3.2.1 Discovering Faults	27
3.2.2 Fault Mitigation	30
3.3 Summary	32
4 Evaluation Overview	33
4.1 Introduction	33
4.2 Evaluation	34
4.2.1 Ideal Evaluation	34
4.2.2 Practical Evaluation	35
4.2.3 Aims	35
4.2.4 Phase 1 - GKG Hazard Analysis	36
4.2.5 Phase 2 - GKG _{SEPE} Fault Treatment	38
4.2.6 Assumptions	39
4.3 Summary	40
4.3.1 GKG Hazard Analysis	40
4.3.2 GKG Fault Mitigation	40

5	Graydon-Knight-Green Certification	41
5.1	Description	42
5.1.1	Phased Inspection	42
5.1.2	Dialectic	45
5.1.3	Challenge Categories	45
5.2	Instantiation	47
5.2.1	Phased Inspection	47
5.2.2	Dialectic	48
5.3	SEPE Instantiation of GKG	48
5.3.1	Qualities	48
5.3.2	Challenges	48
5.4	Summary	56
6	Diabetes Advanced Information System	57
6.1	System Overview	57
6.1.1	DAIS Information Collection	58
6.1.2	DAIS Information Analysis	61
6.2	Role in Evaluation	61
6.2.1	GKG Candidate System Domain - SEPE	62
6.2.2	Evaluation Candidate System - DAIS	63
6.3	DAIS Baseline Safety Case	63
6.3.1	DAIS Hazard Analysis	64
6.3.2	Requirements	64
6.3.3	Hazard Analysis Techniques	66
6.3.4	Safety Case	66
7	GKG Hazard Analysis	69
7.1	Introduction	69
7.2	HazOp	71
7.2.1	Guide Word Selection	72
7.2.2	Parameter Construction	73
7.2.3	Cross Product and Question Generation	73
7.2.4	Relevancy Classification	75
7.2.5	Answers and Fault Classification	76
7.2.6	Results	76
7.3	Fault Tree Analysis	79
7.3.1	Initial Hazard and Event Hypothesizing	80
7.3.2	Event Hypothesizing and Iteration	81
7.3.3	Fault Determination	81
7.3.4	Results	82
7.4	Failure Modes, Effects, and Criticality Analysis	84
7.4.1	Module Selection	85
7.4.2	Failure Mode Determination	85
7.4.3	Effect and Criticality Determination	85
7.4.4	Results	86
7.5	Summary	88
8	GKG Fault Mitigation	90
8.1	GKG _{SEPE} Faults _{cert} Placement	91
8.2	Evaluation Safety Case Construction	93
8.2.1	Fault Mapping	93
8.2.2	Faulty Fragments	95
8.2.3	Evaluation Safety Cases	99
8.3	GKG _{SEPE} Certification	99

8.4	Fault _{cert} Mitigation Strategy Generation	102
8.5	Results	103
8.5.1	Certification Outcomes	104
8.5.2	Generated Fault _{cert} Mitigation Strategies	109
8.6	Summary	112
9	Related Work	114
9.1	Prescriptive Standards	114
9.2	Goal-Based Standards	116
9.3	Assurance Cases	117
9.4	Dependability Assessment	120
10	Conclusion	122
10.1	Results	122
10.1.1	Model Feasibility	123
10.1.2	Technique Feasibility	123
10.1.3	Technique Yield	123
10.1.4	Problem Scope	124
10.1.5	Other Contributions	124
10.2	Limitations	125
10.2.1	Scope	125
10.2.2	Specimen Certification Mechanism	125
10.2.3	Candidate System	126
10.2.4	Application of Techniques	126
10.2.5	Results Metrics	126
10.2.6	Other Limitations	126
10.3	Future Work	127
10.3.1	Refining the Filter Model	127
10.3.2	Refining Filter Model Analysis	127
10.3.3	Filter Model Analysis of Other Certification Mechanisms	128
10.3.4	Benefits and Drawbacks of Safety Cases	128
10.3.5	Standards-Based vs. Goal-Based Certification	128
	Bibliography	130
	Appendix A GKG HazOp	135
	Appendix B GKG Fault Tree Analysis	144
	Appendix C GKG Failure Modes, Effects, and Criticality Analysis	148
	Appendix D DAIS HazOp	153
	Appendix E DAIS Fault Tree Analysis	158
	Appendix F DAIS Safety Case	161

List of Tables

2.1	Disadvantages of prescriptive standards and how goal-based standards deal with these disadvantages.	18
3.1	Hypothetical FMECA results for the calculation module in a morphine infusion pump. . . .	30
3.2	Standard HazOp guide words and their meanings.	31
6.1	FDA SEPE requirements and their instantiation in DAIS.	64
7.1	Measurements for the HazOp analysis on the GKG _{SEPE} certification mechanism.	77
7.2	Results of HazOp analysis on the GKG certification mechanism.	78
7.3	Faults _{cert} discovered in the GKG certification mechanism using HazOp analysis.	79
7.4	Measurements for the fault tree analysis on the GKG certification mechanism.	83
7.5	Results of FTA on the GKG certification mechanism.	84
7.6	Measurements for the FMECA on the GKG _{SEPE} certification mechanism.	88
7.7	Categorization of the open questions in this chapter.	89
8.1	Summary of faults _{cert} found in Phase 1 of the case study.	92
8.2	How the discovered faults _{cert} are placed into GKG _{SEPE}	93
8.3	The partial GKG _{SEPE} fault mapping used in this analysis.	95
8.4	The names of each evaluation safety case and the faulty fragments contained in each evaluation safety case.	100
8.5	The characteristics of the certifications used in this analysis: the faults _{cert} each certification focuses on, the fault _{cand} example, and the evaluation safety case examined in the certification.	101
8.6	Certification results for all evaluation certifications.	104
8.7	Fault _{cert} mitigation strategies and the safety engineering techniques from which they are derived.	110
8.8	Measurements for fault mitigation strategy generation for the GKG _{SEPE} certification mechanism.	111
8.9	Categorization of the open questions in this chapter.	113
10.1	Categorization of the open questions in this thesis.	124

List of Figures

1.1	Certification as protection of the public interest.	3
1.2	The basic view of certification as a system.	6
1.3	Possible erroneous certification decisions.	7
1.4	The filter repair cycle.	9
3.1	Inputs to certification and corresponding outputs.	21
3.2	Reason’s Swiss-cheese model of system failure.	23
3.3	The filter model of certification, with a satisfactory system as input and a warranted acceptance as output.	24
3.4	The filter model of certification, with an unsatisfactory system as input and a warranted rejection as output.	25
3.5	The filter model of certification, with an unsatisfactory system as input and an unwarranted acceptance as output.	26
3.6	The filter model of certification, with a satisfactory system as input and an unwarranted rejection as output.	27
3.7	An illustrated version of the thesis statement.	28
3.8	Example fault tree for the software in a hypothetical morphine infusion pump.	29
4.1	An ideal evaluation, comprising systems across many domains.	34
4.2	The two-phase structure of the case study evaluation used in this work.	36
5.1	GKG process outline: certification based on a safety case.	43
5.2	Well-understood system domains that contributed expected practices to SEPE.	50
5.3	A fragment with two potentially dependent subarguments.	52
5.4	Example safety argument fragments exhibiting two differing strategies of top-level argumen- tation. One strategy argues over all known hazards, while the other argues conformance to a standard.	54
5.5	Example safety argument fragments exhibiting two differing strategies of arguing about verification. One strategy argues that only testing is sufficient, while the other argues that verification is <i>also</i> necessary.	54
5.6	The Hazardous Software Failure Mode Decomposition pattern.	56
6.1	The DAIS application model (from Lin [1]).	58
6.2	DAIS display, showing CGM readings and insulin boluses on the same chart.	62
6.3	A portion of the baseline safety case. This GSN structure is a diverse two-legged argument, designed to show partial mitigation for Event 1.	68
7.1	Summary of the evaluation used in this work (from Chapter 4).	70
7.2	The functional block diagram used for the safety analyses.	71
8.1	Hypothetical simple fault mapping for one fault _{cand} in a hypothetical certification mechanism.	94
8.2	The Evidence Insufficiency Fragment, <i>Frag_{EI}</i>	97
8.3	The Lack of Formal Verification Fragment, <i>Frag_{LOFV}</i>	97

8.4	The Argument From Ignorance Fragment, <i>Frag_{AFI}</i>	98
8.5	The Conformance Argument Fragment, <i>Frag_{CA}</i>	100
8.6	The top-level fragment of <i>SC_{base}</i> , against which Challenge 5.11 should have been issued. . .	107
1	The top of the DAIS safety case.	162
2	DAIS safety case portion starting with G.DIP.1.Misinformation.	163
3	DAIS safety case portion starting with G.DIP.1.Lack of Information.	164

Chapter 1

Introduction

Safety-critical application domains contain some of today's most complex software systems. Ideally, such systems would be entirely free of risk; that is, a comprehensive risk analysis would produce a way to eliminate all possible risk. However, any non-trivial system will almost invariably have some residual risk, either in the form of identified risks that could not be entirely eliminated or unidentified risks [2].

Thus, the assurance of the residual risk of a system being minimized is essential to the development of safety-critical systems. Moreover, most such systems must be approved by a regulatory agency. Such regulatory agencies are tasked by their government to protect the public interest; thus, this "stamp of approval" is an explicit statement that an independent regulatory body finds the system "adequately safe." A decision on whether a system is adequately safe or not is the product of **safety assessment**.

Definition 1.1. Safety assessment. *Safety assessment is the process through which a regulatory agency determines whether the residual risk of a particular system is acceptably low for the system's intended context and usage.*

Safety-critical systems predate the advent of the microcomputer; Lawrence Sperry received the first patent for airplane autopilot in 1929 [3]. Before computers became inexpensive and reliable enough to use in safety-critical systems, these systems consisted primarily of mechanical and electrical components. Such components have properties that are (a) well-understood and (b) subject to statistical modeling. The hardware components of computers also consist of mechanical and electrical components, and thus are subject to the same kind of statistical risk analysis. Statistical analyses yield a **probability of failure**.

Definition 1.2. Probability of failure. *The probability of failure of a system is the prior probability per unit time that, on a given input, the system does not produce the specified output.*

The probability of failure is one component of the residual risk of a system; the other is the *consequence of failure* or **loss**.

Definition 1.3. Loss. *The loss associated with a failure is the cost incurred by that failure.*

Related to the notion of loss is the notion of **accident**.

Definition 1.4. Accident. *An accident is an event that results in a loss.*

The definition of risk follows from the definitions of (a) probability of failure and (b) loss:

Definition 1.5. Risk. *Risk is the expected loss per unit time associated with using a system in its intended context and for its intended purpose.*

Because risk is an expected value, we can express this value mathematically:

$$\sum_i P[failure_i] \times loss(failure_i) \quad (1.1)$$

where i identifies a particular failure that can occur during system operation.

This equation assumes that we can measure the magnitude of loss. Loss can be measured in a few different ways, including financial cost or time lost. Losses in life-critical systems are associated with death; because of the delicate nature of putting a price on human life, engineers do not usually work directly with risk. Instead, engineers often work only with the probability of failure. Thus, the safety assessment of a system is essentially reduced to a comparison of the system's probability of failure and the **threshold probability of failure**.

Definition 1.6. Threshold probability of failure. *The threshold probability of failure in a safety assessment is the the maximum probability of failure that a certifier considers adequately safe for a particular system domain.*

Ideally, if a system's probability of failure is less than the threshold probability, the certifier will approve the system; if the system's probability of failure is *greater* than the threshold probability, the certifier will *not* approve the system.

However, software systems are fundamentally different from mechanical or electrical systems: they do not fail in *statistically* predictable ways, i.e., randomly. Rather, their failures are caused by *design* faults [4], which means that if the system executes a piece of code implementing a faulty design element, its corresponding failure will occur deterministically – there is no randomness inherent in a design fault. Since there are no generally accepted mathematical models for design faults, developers must turn to other methods to demonstrate to certifiers that their systems are adequately safe.

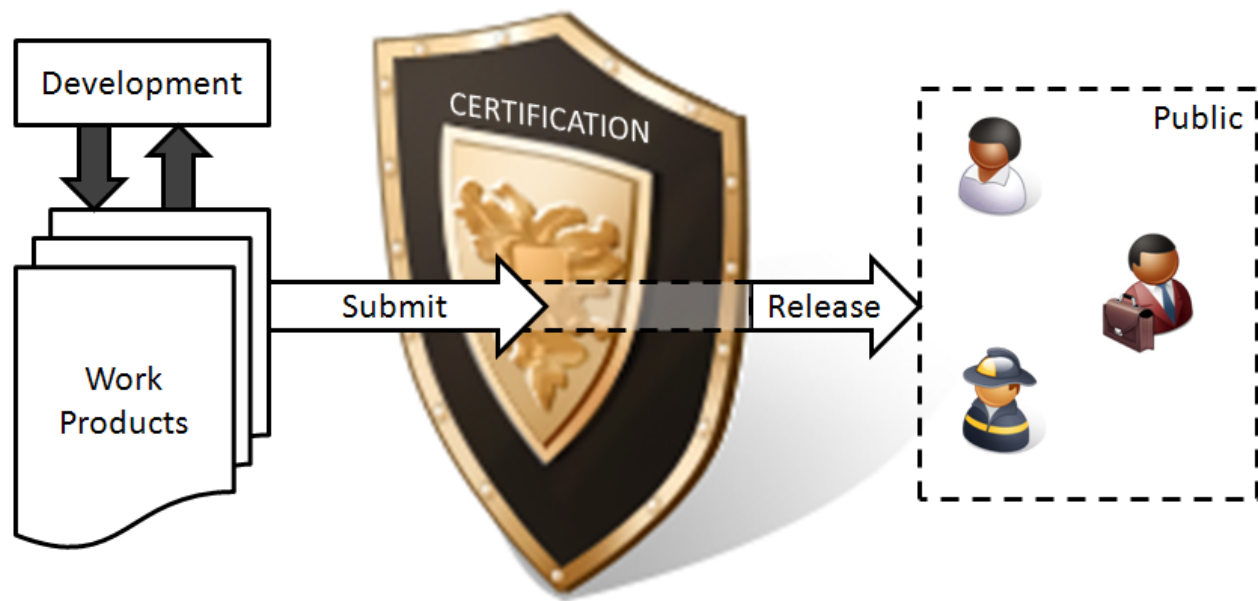


Figure 1.1: Certification as protection of the public interest.

1.1 The Certification Problem

The aim of any regulating agency is to protect the public interest through **certification** of systems that fall under its purview.

Definition 1.7. Certification. *A process employed by regulatory agencies to protect the public interest by preventing unfit systems from being released to the public.*

Figure 1.1 shows the act of certification as *shielding* the public interest from potentially dangerous systems. The mechanism of certification is dependent on two items:

1. The definition of “unfit.” For a given regulator, “unfit” is defined relative to the public interest assigned to that regulator; this thesis focuses on regulators protecting the public interest in *safety*.
2. The *safety assessment method*. Examining a system in and of itself is difficult, so a given system is represented by its *work products*. Examples of work products examined in certification include test reports, requirements traceability matrices, and specification documents.

The combination of these work products is called the certification **submission package**.

Definition 1.8. Submission package. *The submission package for a given system is a set of work products that represent the properties of the system, as accurately as possible.*

Effectively, submission packages serve as a *proxy* for the system in certifications. Regulators, such as the United States (US) Food and Drug Administration (FDA), the US Federal Aviation Administration (FAA), and the US Nuclear Regulatory Commission (NRC) must examine thousands of these submission packages to determine whether the residual risk of system usage has been reduced to an adequate level.

If quantitative modeling of software system dependability were straightforward, certification would be a simple comparison of probabilities. However, the challenges inherent in quantitative modeling of software systems are so great that regulating agencies do not require quantitative modeling alone for certification. For the most part, **prescriptive standards** provide the basis for certification efforts in the US.

Definition 1.9. Prescriptive standard. *A prescriptive standard is a document that dictates certain development practices thought to help in creating acceptably safe systems in a certain domain.*

In order to be certified by an agency that uses prescriptive standards, developers must adhere to the practices dictated in the relevant standards. Dictating development practice enables regulators to demand certain techniques that are *known* to eliminate or limit certain classes of faults in an effort to satisfy the dependability requirements of the system. However, prescriptive standards exhibit two broad categories of disadvantages:

1. Prescriptive standards are *inflexible*; by definition, they prescribe certain activities and approaches to development, often to the *exclusion* of other activities and approaches.
2. Prescriptive standards rely on an *unjustified assumption*: namely, that *adherence to the standard requirements* will fulfill the dependability requirements of a system.

This work refers to the assumption in the second problem as the “**adherence assumption**.” The adherence assumption is especially worrisome: prior work has not shown that the adherence assumption is always *wrong*, but work by German [5] has suggested that standards provisions do *not* always impart the level of dependability they are intended to provide.

German’s work, coupled with the lack of carefully controlled studies on the efficacy of oft-recommended standards provisions, has resulted in the rise of **goal-based standards** [6, 7, 8, 9, 10].

Definition 1.10. Goal-based standard. *A goal-based standard is a document that sets goals for the safety of a system.*

Developers who wish to adhere to a goal-based standard must provide a documented rationale for why they believe that their system is safe. Such a rationale is called a *safety case* [11, 12, 13, 14]. Goal-

based standards deal with the disadvantages of prescriptive standards, but present two new categories of disadvantages:

1. There is a distinct lack of established quality metrics for safety cases, i.e., ways to determine whether a safety case presents a believable rationale or not.
2. By contrast with prescriptive standards, goal-based standards are highly flexible; this flexibility allows for significant variation in the materials that comprise the safety case.

The lack of quality metrics for safety cases affects both certifiers and developers:

- **Certifiers.** *Certifiers* must conduct safety assessments without an objective notion of whether a safety case shows that a system is adequately safe. In addition, variation in the forms of safety case materials necessitates that certifiers be prepared for every eventuality, further complicating the certifier's attempts at safety assessment.
- **Developers.** Because of the difficulty certifiers have with safety assessments, *developers* have trouble predicting certification decisions. The plethora of options available to developers for safety case construction and presentation require developers to devote non-trivial amounts of resources solely for the purposes of constructing a safety case, adversely affecting development time and budget.

Thus, both prescriptive and goal-based standards exhibit faults. As the purpose of certification is to protect the public interest, *obvious faults in certification are unacceptable*. Faults in certification can lead to *unacceptably dangerous* systems being released to the public under the guise of *acceptable systems*.

Mitigating certification faults requires discovering and characterizing the faults; discovery of faults requires a general fault model. This thesis presents the **filter model** of certification to this end.

1.2 The Filter Model

The fundamental assertion of the filter model is:

*Any given certification mechanism can itself be viewed
as a safety-critical system.*

This certification *system* takes submission packages as *input* and produces **certification decisions** as *output*.

Definition 1.11. Certification decision. *Given a candidate system as input, a certification process decides whether to (a) approve or (b) reject the system for public use. This certification decision is based on the safety assessment of the system.*

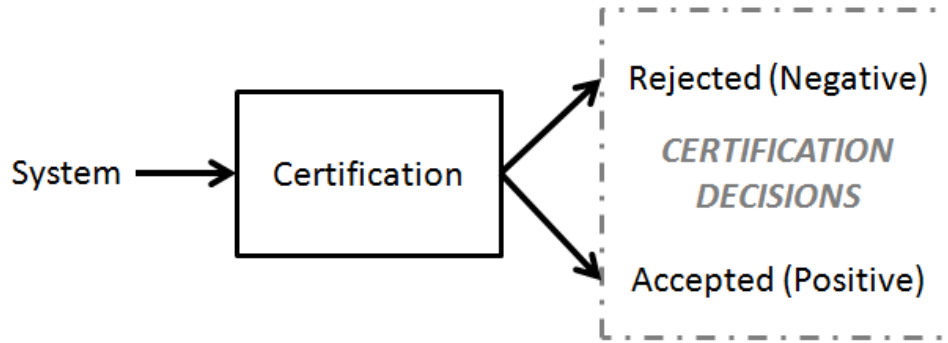


Figure 1.2: The basic view of certification as a system.

Definition 1.12. Positive certification decision. A positive certification decision is a certification decision to approve a candidate system, i.e., a judgment by the certifier that the candidate system is adequately safe.

Definition 1.13. Negative certification decision. A negative certification decision is a certification decision to reject a candidate system, i.e., a judgment by the certifier that the candidate system is not adequately safe.

Protracted and frequent discussions are likely to take place between the applicant and the certifier. Although such discussions might affect the certification decision, in the end the certifier always arrives at a certification decision. Figure 1.2 shows this basic view of certification as a system. The certification mechanism examines submission packages that are provided as input. Since a given submission package serves as a proxy for a given system, this work refers generally to the input of certification mechanisms as “systems.”

Essentially, the two types of certification decision are the output of a *detection system* of whether a given system is fit for public use. As an example, consider a hypothetical certification mechanism¹ for drug infusion pumps, named CM_{DIP} . Modern drug infusion pumps, which are commonly referred to simply as “infusion pumps,” contain a significant software component that enables programmability [16, 17]. Infusion pumps are the *input* to CM_{DIP} . CM_{DIP} attempts to *detect* whether a particular drug infusion is worthy of a positive certification decision. This decision is the *output* of CM_{DIP} .

An erroneous certification decision is either:

- An unacceptably unsafe system that is deemed fit for public use.
- An acceptably safe system that is deemed unfit for public use.

¹The Food and Drug Administration (FDA) offers guidance on certification of medical devices, including infusion pumps [15]. For the purposes of this example, such detail is unnecessary.

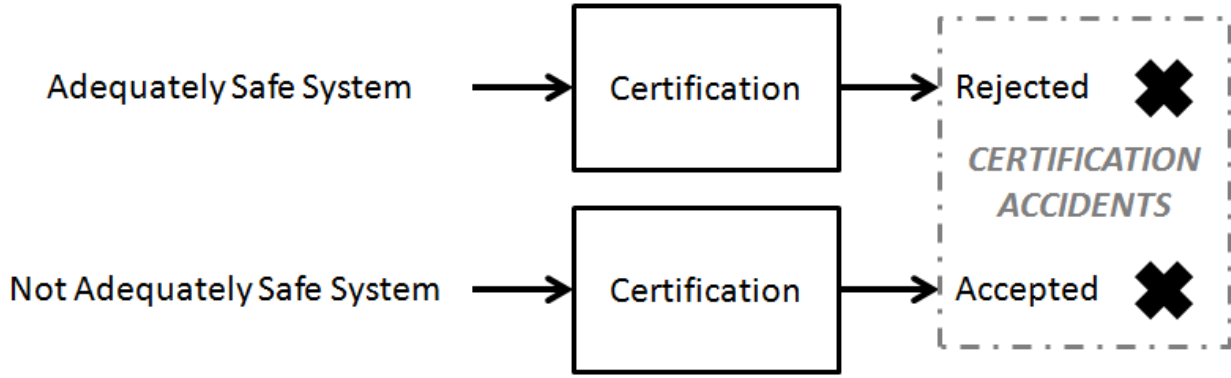


Figure 1.3: Possible erroneous certification decisions.

When viewed through the lens of certification as a safety-critical system, an erroneous certification decision is a **certification accident**. Figure 1.3 shows the concept of erroneous certification decisions as accidents (Chapter 3 expands on the concept in depth).

Continuing with the previous example of CM_{DIP} , assume that the candidate system is a morphine infusion pump. Further assume that the pump has a defective dosing system that allows lethal amounts of morphine to be infused into the patient. Certification of this candidate system is a certification accident.

Accidents, hazards, and faults are related but subtly distinct and, as a result, often confused with one another. This work uses the definitions given below, which are in line with safety engineering literature and practice [18]. Recall the previous definition of **accident**: an accident is a a loss event.

Definition 1.14. Hazard. *A hazard is an erroneous system state that could lead to an accident.*

Definition 1.15. Fault. *A fault is the adjudged cause of an erroneous system state.*

While sufficient for other discussions of accidents, hazards, and faults, *the definitions so far are not precise enough for the purposes of this work*. Since the fundamental assertion of the filter model is that **certification itself is a system**, we necessarily use the same terminology of “accident,” “hazard,” and “fault” to refer to hazards and faults of both the *certification mechanism* and the *candidate system*. This ambiguity introduces significant difficulty in determining to which *type* of hazard/fault we refer. As the two terms are already easily confused, this thesis introduces two new designations for hazard and fault:

- The subscript **cert** designates that the scope of the accident/hazard/fault as at the *certification mechanism* level.
- The subscript **cand** designates that the scope of the accident/hazard/fault as at the *candidate system* level.

The definitions below are derived from the established definitions of fault and hazard.

Certification Mechanism Terminology:

Definition 1.16. Accident_{cert}. *An accident_{cert} is an erroneous certification decision.*

Definition 1.17. Hazard_{cert}. *A hazard_{cert} is a state in a certification system that could lead to an accident_{cert}.*

Definition 1.18. Fault_{cert}. *A fault_{cert} is the adjudged cause of a hazard_{cert}.*

Candidate System Terminology:

Definition 1.19. Accident_{cand}. *An accident_{cand} is a loss event in a candidate system.*

Definition 1.20. Hazard_{cand}. *A hazard_{cand} is an erroneous state in a candidate system that could lead to an accident_{cand}.*

Definition 1.21. Fault_{cand}. *A fault_{cand} is the adjudged cause of a hazard_{cand}.*

The previous example of CM_{DIP} included an accident_{cert}: certification of a defective infusion pump. An accident_{cand} in this scenario is if the defective infusion pump *did*, in fact, provide a lethal dosage of morphine to a patient.

In both certification mechanisms and candidate systems, faults cause hazards; we would like to mitigate faults_{cert} to the extent possible. Viewing a given certification mechanism, CM , as a system allows us to leverage extant safety engineering techniques for fault_{cert} discovery and mitigation. More precisely:

- We can adapt and apply existing *hazard analysis* techniques to CM to systematically analyze the hazards_{cert} and, from there, adjudge the faults_{cert} associated with hazards_{cert} in CM .
- Once we discover various faults_{cert} in CM , we can adapt and apply existing *fault mitigation* techniques to CM to lessen the severity of or even eliminate the faults_{cert} in CM .

The result of fault_{cert} mitigation on CM is *also* a certification mechanism, just with less faults_{cert}. We can apply safety engineering *iteratively*, improving the safety of CM with each iteration. I call this iteration the *filter repair cycle*. Figure 1.4 illustrates the filter repair cycle.

1.3 Thesis Statement

The thesis of this work is that the filter model enables the systematic fault identification and mitigation in certification mechanisms. More precisely:

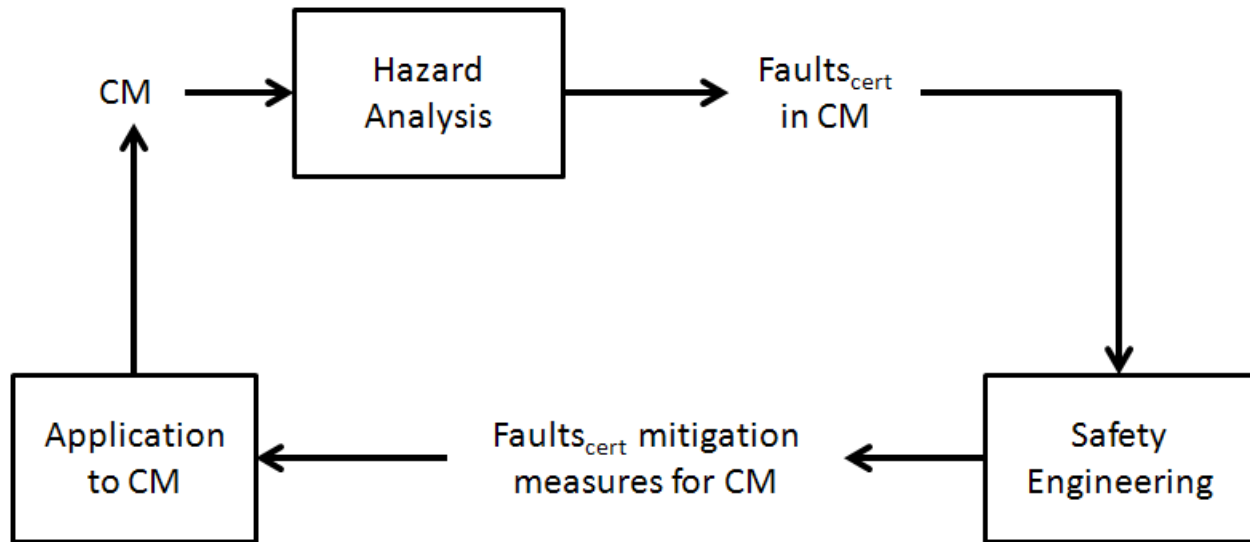


Figure 1.4: The filter repair cycle.

Existing hazard analysis techniques can identify faults in certification systems, which can then be mitigated by existing safety engineering techniques.

To test this hypothesis, the author conducted a case study on a specimen certification mechanism, the Graydon-Knight-Green method, or GKG. The target system for the GKG case study was the Diabetes Advanced Information System, or DAIS. DAIS is the first in a new class of system, the **safety-enhanced patient environment**.

Definition 1.22. Safety-enhanced patient environment. *A safety-enhanced patient environment, or SEPE, is an information system that augments the living environment of a patient in order to improve the patient's safety as much as possible without relying on the patient or any medical treatment devices.*

1.4 Thesis Outline

1. Chapter 2 describes current current product certification practices.
2. Chapter 3 presents the filter model of certification and the filter repair cycle.
3. Chapter 4 gives an overview of the evaluation, including:
 - The specimen certification mechanism, GKG.
 - The adapted techniques used to identify and mitigate faults_{cert}.
 - The target system, DAIS, and the system domain to which DAIS belongs, SEPE.
 - The methods used to seed the discovered defects into the certification process.

- Proposed fault mitigations, based on the identified faults and mitigation techniques.
- A summary of the results.

Chapters 5 through 8 expand on all of the above topics in detail.

4. Chapter 9 presents a summary of related work in the general areas of certification and assurance case quality assessment.
5. Chapter 10 concludes and presents some avenues for future work.

Chapter 2

Certification: Goals and Requirements

Regulatory agencies have the public interest as their primary focus. For safety-critical industries, the public interest is safety. Thus, the overarching purpose of regulators of safety-critical industries is to protect, to the extent possible, the public's safety. Regulators must deem a safety-critical system to be safe enough for the public to use *before* the system is deployed for public use. Safety is dependent on context and intended usage; even an innocuous "system" like a brick can be misused, despite being completely safe for its intended purpose (as a building block for a structure) and in its intended context (construction).

Chapter 1 refers informally to "certifiers" and "developers" without defining the terms. The intuitive definitions of the terms are not sufficient for a detailed description of certification; below are rigorous definitions of both terms.

Definition 2.1. Certifier. *A regulatory agency, tasked (usually within a legal framework) with protecting the public interest in safety for a particular domain.*

Definition 2.2. Developer. *A person, corporation, or conglomerate that produces systems in a particular domain.*

The fundamental goal of developers, especially corporate developers, is to *generate profit*. This goal may initially seem to be at odds with the goal of certification. However, since systems in regulated domains must be certified in order to generate profit, we can assign the goal of protecting the public interest to developers without obscuring the principles of interaction in certification.

2.1 Goals

Previously, this work referred informally to the **main goal** of certification: to protect the public interest. Below is a rigorous definition of the term.

Definition 2.3. Main goal. *The main goal of certification is to ensure that any system released to the public exhibits residual risk that is as low as reasonably practicable¹ when the system is operated in its intended context and for its intended purpose.*

This goal has several complex components. We can break this goal down into several *sub-goals* that are designed to help both certifiers and developers achieve the main goal.

2.1.1 Sub-Goals

For the purposes of this work, certification can be broken into sub-goals. The following sections elaborate on each of these sub-goals:

- A clear definition of safety.
- Clear expectations that certifiers have of submission packages.
- An accurate certification mechanism.
- A consistent certification mechanism.

Definition of Safety

The definition of “safety” regarding a particular domain should be clear to those with an interest in that domain. The exact words of the definition vary from agency to agency, but the overall meaning is the same. For example, the FDA uses the following definition of safety, from the Code of Federal Regulations [20]:

“There is reasonable assurance that a device is safe when it can be determined, based upon valid scientific evidence, that the probable benefits to health from use of the device for its intended uses and conditions of use, when accompanied by adequate directions and warnings against unsafe use, outweigh any probable risks. The valid scientific evidence used to determine the safety of a device shall adequately demonstrate the absence of unreasonable risk of illness or injury associated with the use of the device for its intended uses and conditions of use.”

The definition seems complete, but showing that a system fulfills these safety conditions is difficult. Any approach to certification must provide sufficient guidance about how to show that a certain system is safe according to the certifying agency’s chosen definition of safety.

Clear Expectations

The expectations that certifiers have of submissions should be clearly communicated and reasonable. This goal is related to the previous goal, but encompasses *all* expectations that a certifier has of submissions, e.g., presentation format, volume, content, etc. Ideally, the certification process should be systematic,

¹“As low as reasonably practicable” is an established term in the system safety literature, usually shortened to ALARP [19].

comprehensive, and focused on the technical aspects of the submission. Submission expectations define how amenable a submission is to such a review.

To a large extent, these expectations are meant to reduce, as much as is practical, the effort a reviewer must put forth in order to review a submission. Therefore, submissions should be accessible by and familiar to certifiers. However, if developers do not understand certifier expectations, or find the expectations drastically impractical, the expectations will not be met and therefore will not reduce reviewer effort. Submissions that do not meet expectations could be rejected because the certifier must spend extraordinary amounts of effort simply to understand the submission.

Accuracy

The certification process should produce accurate certification decisions on whether a given submission is **certifiably safe**.

Definition 2.4. Certifiably safe. *A certifiably safe submission demonstrates that the system meets the criteria the certifier has set with regard to safety.*

These criteria include the definition of safety and other submission expectations. Recall the definition of certification decision stated in Chapter 1 (Definition 1.11): a decision on whether a candidate system should be certified or not. Given this definition, we arrive at the definition for **accurate certification decision**.

Definition 2.5. Accurate certification decision. *An accurate certification decision is one that correctly identifies whether a candidate system is certifiably safe.*

Logically, an **inaccurate certification decision** is the opposite of an accurate certification decision.

Definition 2.6. Inaccurate certification decision. *An inaccurate certification decision is one that incorrectly identifies whether a candidate system is certifiably safe.*

Note that there are two ways that a certification decision can be accurate. The certification mechanism can:

1. Identify a certifiably safe candidate system as such.
2. Identify an uncertifiably unsafe candidate system as such.

Similarly, there are two ways that a certification decision can be inaccurate. The certification mechanism can:

1. Identify a certifiably safe candidate system as uncertifiably unsafe.
2. Identify an uncertifiably unsafe candidate system as certifiably safe.

Chapter 3 describes inaccurate certification decisions in greater detail.

Consistency

The certification process should be consistent both *across* and *within* domains in a given certifier's purview. Consistency in the certification process of a given certifier is important for objectivity and repeatability. Consider two different systems in *different* domains: A and B. Assuming both submissions demonstrate the systems' certifiable safety equally well, the certification process should yield the same certification decision for both A and B. Similarly, if A and B are different systems within the *same* domain, equally good submissions should yield the same certification decision for A and B.

2.1.2 Certification and Residual Risk

Note that none of the sub-goals directly mention assurance of residual risk. This is due to the difficulty of quantitative system safety assessment. Historically, *compositional models* have proven to be useful in safety assessment. Compositional models are models that allow engineers to make system-level assessments by combining module-level assessments, according to the model. Ideally, we would like to use compositional models to assess safety, but modern software-intensive systems suffer from the following important fact:

There are no generally accepted compositional models for design faults in systems.

Detailed reliability growth models for software systems *do* exist, but no model is general enough to apply to all software systems [21]. This difficulty is primarily due to design faults being the *only* kind of fault in a software system, (or sub-system) [22]. That is, there are no degradation or Byzantine faults inherent in software systems (although such faults can certainly occur in larger systems that contain hardware as well as software sub-systems). Design faults are not random but *systematic* and thus not amenable to analysis using statistical models [4].

Without models to define combination, dependability attributes for large code bases cannot be determined in an acceptable way by combining dependability attributes of smaller parts of the system.

2.2 Current Practices

Recall from Chapter 1 that, because of the significant difficulty in product certification for dependability, developers and certifiers have worked together to create two different types of standards: *prescriptive* and *goal-based*.

2.2.1 Prescriptive Standards

A prescriptive standard is a document that dictates certain development practices usually demonstrated to help in creating acceptably safe systems in a certain domain.

Recall that a prescriptive standard is one that requires developers to adhere to a certain set of development practices in order to comply with the standard. Such standards can provide requirements for either or both of the following:

1. Specific types of work products that must be produced.
2. Specific development procedures that must be followed.

Examples of prescriptive standards are RTCA DO-178B [23], NASA-STD-8739.8 [24], UL 1998 [25], and SAE ARP 4754 [26]. In the prescriptive standard approach, certification of a system is achieved by the certifier assessing compliance with the standard. The certifier inspects the submission package for (a) the prescribed work products and (b) proof that the developer followed the prescribed development procedures.

Advantages and Disadvantages

Prescriptive standards provide developers and certifiers with several advantages:

1. Techniques that are specifically designed to effect avoidance or elimination of an entire class of faults can be included in a prescriptive standard. For example, mandating the usage of the Ada programming language will eliminate faults associated with ambiguous typing, because Ada is a type-safe language [27].
2. Guidance on process usage can be encoded in the standard requirements; this guidance can be designed to ensure that developers fulfill temporal and form requirements of certain critical process steps. For example, requiring a spiral model process for software development will ensure that developers evaluate development risk first and continue to periodically evaluate development risk.
3. Common reliance on a well-defined prescriptive standard means that all engineers on a project understand important details on how a project must proceed.
4. Prescriptive standards bring together a wealth of accumulated knowledge. No engineer is an expert in every subject area that his work may touch; standards serve to “fill in the gaps” in knowledge, thereby improving the efficacy of individual engineers and personnel in general.

However, prescriptive standards also exhibit two significant disadvantages: *inflexibility* and *reliance on the adherence assumption*.

Inflexibility. Prescriptive standards, by their very nature, are inflexible. Locking developers into one specific set of artifacts they must produce and/or processes they must execute in specific ways may not be the best approach for all of the systems the developers must create, especially if standards cross application domains. For example, ISO 14971 [28] is a standard extensively used in the broad medical device domain. Its stated scope is:

“This International Standard specifies a process for a manufacturer to identify the hazards associated with medical devices, including in vitro diagnostic (IVD) medical devices, to estimate and evaluate the associated risks to control these risks, and to monitor the effectiveness of the controls. The requirements of this International Standard are applicable to all stages of the life-cycle of a medical device.”

While this standard provides developers with valuable development and certification guidance on various necessary development activities, the standard applies to *all* medical devices. The characteristics of surgical robots [29] are vastly different from HIV diagnostic tests [30], automated external defibrillators [31], infusion pumps [16], etc. Specific systems may benefit from specific development strategies or technologies, but prescriptive standards effectively preclude such deviations, even if justified. Deviation from the standard may result in certification failure, which is costly in terms of immediate profit from the device, time spent in repairing or redesigning the device, and reputation.

Reliance on the adherence assumption. Prescriptive standards rely on the assumption that following the prescribed *processes* and producing the prescribed *work products* effects the *goal*: a certain level of residual risk, i.e., a system which is acceptably safe for a particular domain. However, there is no scientific basis for the adherence assumption; some work even suggests that certain standard prescriptions intended to effect higher dependability do not do so [5]. Moreover, there is no way of measuring the extent to which the developers followed the *intent* of a prescriptive standard.

Despite these disadvantages, prescriptive standards have worked fairly well across the world. Nevertheless, goal-based standards deal with these disadvantages.

2.2.2 Goal-Based Standards

An alternative approach to prescription is the goal-based approach. Goal-based standards set *goals* for the dependability of a product; the developers are then free to pursue any approach they choose to meet the goals. Typically, developers must provide their documentation for their belief in the claim that their system meets the necessary **dependability goal**.

Definition 2.7. Dependability goal. A *dependability goal* is a dependability requirement set by a particular certifier in a goal-based standard.

Since the certifiers in this discussion are interested in protecting the public interest in *safety*, this work focuses on one particular type of dependability goal, the **safety goal**. A set of documentation supporting a developer's belief that a system satisfies the requisite safety goal is called a **safety case**. Kelly's definition of safety case is sufficient for the purposes of this work.

Definition 2.8. Safety case. *A safety case is "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment" [14].*

Safety Cases

Safety cases have been built to document rationale for the belief in the safety of a variety of production systems, and are required by law for some systems in Europe by various agencies. Examples of goal-based standards are the UK Defence Standard 00-56 [9], UK Civil Aviation Authority CAP 670 [10], and FDA Draft Guidance for Infusion Pumps [15].

A safety case comprises three essential elements:

1. A safety goal that is required for the system.
2. Evidence that the safety goal has been satisfied.
3. An argument linking the evidence to the safety goal.

The argument should convince the reader that the evidence justifies the **safety claim**.

Definition 2.9. Safety claim. *The safety claim is the claim by a developer that a particular system satisfies the requisite safety goal.*

Safety arguments can be recorded in natural language, but natural language safety arguments suffer from similar problems to natural language software requirements specifications:

Clarity. Natural language, when crafted properly, can be quite clear and expressive. However, writing clear natural language is difficult, and the skill to do so is not easily learned. Engineers tasked with writing safety arguments may not receive *any* formal writing training. Even if they do receive writing training, technical training is an engineer's primary focus.

Lack of writing prowess is not a fundamental problem with natural language; poor writing skills can be improved with comprehensive training. In an ideal world, all engineers would successfully complete such training. However, these programs are prohibitively expensive, both in terms of finances and time.

Poor structure. Pure natural language lends itself to a linear narrative. However, safety arguments often require *cross-references* between the argument and pieces of evidence [13]. The more evidence required,

Prescriptive Standard Problem	Goal-Based Standard Solution
Prescriptive standards dictate details of development processes and products, resulting in inflexibility.	Developers adhering to a goal-based standard can follow any processes they want, as long as the dependability goals of the system are achieved.
Prescriptive standards rely on the unfounded assumption that a set of prescribed processes and products can effect the desired level of dependability.	Goal-based standards require an assurance case, a rigorous rationale for why the submitted system fulfills the dependability requirement of the certifier.

Table 2.1: Disadvantages of prescriptive standards and how goal-based standards deal with these disadvantages.

the more cross-references appear in the safety argument. Safety-critical systems rightfully require a great deal of evidence to support a safety claim; thus, safety arguments must include a great deal of cross-references to relevant pieces of evidence.

Succinctly:

*Natural language imposes neither clarity
nor structure on a safety argument.*

Because of unclear and poorly structured prose, natural language safety arguments are often ambiguous and difficult to understand [13]. Graphical notations have been designed to facilitate presentation of safety arguments in a manner that is easier (but not necessarily easy) for humans to understand. Examples of these notations include the Claims, Arguments, and Evidence notation (CAE) [8] and the Goal-Structuring Notation (GSN) [12]. The graphical components of each of these notations is a directed acyclic graph. This work uses GSN when treating safety arguments because GSN is one of the more widely used notations [32]; readers unfamiliar with GSN are referred to Kelly [12, 14].

Advantages and Disadvantages

Goal-based standards remedy both of the problems that arise with prescriptive standards, as shown in Table 2.1. However, goal-based standards introduce two new disadvantages:

Lack of quality metrics. Currently, there are no established quality metrics for safety cases. Despite the rigorous nature of safety cases, they are informal, so certifiers have difficulty assessing safety case quality *quantitatively* without attaching subjective judgments to the assessment.

If the argument in a safety case is flawed in some way, belief in the top-level goal might be unwarranted. This issue can be seen in UK Defence Standard 00-56, which states:

“The Safety Case shall consist of a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment.”

The standard requires a safety case that is *compelling*, *comprehensive* and *valid*. Despite this requirement, the standard does not define these terms, leaving in doubt how the requirements can be met and how a certifier will determine whether they have been met.

Significant variation in submission packages. The flexibility of the goal-based approach, while a strength when compared with the inflexibility of prescriptive standards, introduces a significant degree of variation into the potential submissions a certifier could see. The certifier could be presented with a safety case that does not necessarily facilitate review in any sense. Claims, arguments, and evidence in a safety case can take many forms, and the certifier must be prepared for any of them.

These issues cannot be easily addressed. Experts involved in certification at the FDA, for example, work on a variety of devices, so any solution must be broadly applicable both across domains and within each domain.

2.3 Summary

We have established in this chapter that both approaches to certification – prescriptive and goal-based – have problems. Two related questions remain:

Question 2.1. *How do these issues manifest in a particular certification mechanism?*

Question 2.2. *How can we evaluate the viability of a certification mechanism?*

Question 2 introduces the notion of **viability**.

Definition 2.10. Viable. *A viable certification mechanism is one that effectively protects the public interest.*

A regulator effectively protecting the public interest in safety means that the systems that are certified by the regulator are certifiably safe with a high degree of accuracy. A general model of certification would provide the first step to answering these questions; Chapter 3 describes a general model created by the author, the filter model.

Chapter 3

The Filter Model of Certification

Evaluating whether a given certification mechanism is viable is a difficult task. Ideally, we would study the mechanism empirically. To come to any conclusive results, such a study would necessarily be large scale, i.e., with participation of multiple regulatory agencies and many subject systems across different domains over many years. The resources required to conduct such a study are prohibitively high. In addition, deploying untested certification mechanisms is a dangerous activity. An untested mechanism could have a systematic flaw that allows certification of systems with certain critical defects. Accidents involving such systems could result in loss of life, the prime outcome against which certification should protect.

Another approach to determine the viability of a certification mechanism is to instantiate a general model of the mechanism itself and examine the model for flaws. Such an approach is far less resource-intensive, but requires a comprehensive, representative, and general model. To my knowledge, **no such models exist for certification mechanisms**. This chapter introduces the *filter model* of certification that models certification as a system which is designed to prevent systems with unacceptably high residual risk from being released to the public. That is, systems with unacceptably high residual risk are *filtered out* by certification. The filter model enables pre-deployment evaluation and repair of arbitrary certification mechanisms, which is necessary for highly reliable operation of certification mechanisms.

3.1 The Filter Model

3.1.1 Inputs and Outputs

The first step toward defining the filter model is characterizing the inputs and outputs of certification. We have established that the purpose of certification is to protect the public interest in safety. Generally,

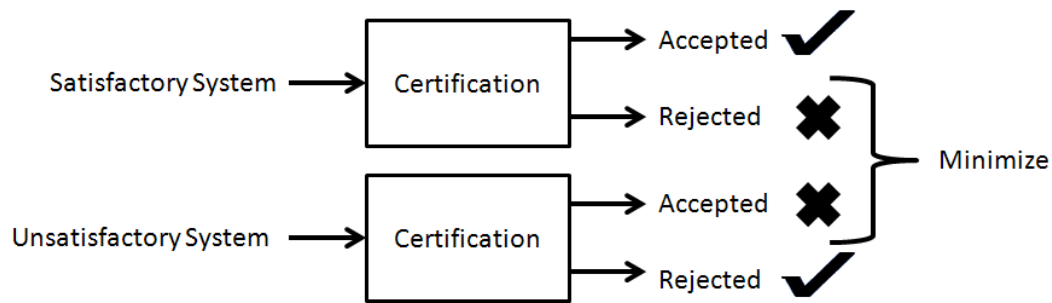


Figure 3.1: Inputs to certification and corresponding outputs.

certification mechanisms accomplish this goal with two **warranted outcomes**.

Definition 3.1. Warranted outcome. *A warranted outcome is a certification decision that properly protects the public interest.*

There are two types of warranted outcome: acceptance of an *adequately safe system* and rejection of a system that is not adequately safe. However, since certification is not perfect, **unwarranted outcomes** are also possible.

Definition 3.2. Unwarranted outcome. *An unwarranted outcome is a certification decision that does not properly protect the public interest.*

Thus, an unwarranted acceptance is the acceptance of an unsatisfactory system, and an unwarranted rejection is the rejection of a satisfactory system.

The inputs to certification are candidate systems, which are either satisfactory or unsatisfactory.

Definition 3.3. Candidate system. *A candidate system is one which has been deemed to require certification*

For simplicity, this work assumes that candidate systems have two important properties:

- A candidate system is either adequately safe or not adequately safe.
- A candidate system falls under the purview of a single regulatory agency.

The outputs of certification are warranted and unwarranted acceptances and rejections. Figure 3.1 shows the relationship between these inputs and outputs.

Both developers and certifiers would benefit from minimizing the unwarranted outcomes of certification. To do this, we must first examine how certification operates in a general sense.

3.1.2 General View of Certification

A sufficiently general view of certification must accurately reflect the commonalities between the two different types of certification. Fortunately, both types have an important commonality, that of **inspection**:

- With **prescriptive standards**, certification consists of the **inspection** of a collection of work products for the presence of various critical work products.
- With **goal-based standards**, certification consists of the **inspection** of a collection of work products for fulfillment of the goals set forth in the standard.

In either case, certification consists of the inspection of a representative set of work products. For simplicity, I will refer to these work products simply as the system, as their purpose is to proxy for the actual system. The inspection is usually a multi-stage process, with several different work products being examined during each stage.

A particular certification mechanism, *CM*, can be thought of as a filtration **system**. For the purposes of this work, I define a filtration system as having one or more *filters*. Each filter is designed for two complementary purposes:

1. Prevent a certain type of undesirable particle from being released past the filter. In the context of air filters, an example of an undesirable particle is a pollen particle.
2. Allow desirable particles to pass the filter. In the context of air filters, a desirable particle is any particle that constitutes clean air.

Thus, unwarranted outcomes are the *failures* of this system. The filter model of certification is based loosely on Reason's Swiss cheese model of system failure [33], shown in Figure 3.2. The different layers of the Swiss cheese model are designed to protect against a failure. The holes in the Swiss cheese model represent system **faults**, and the different layers of cheese represent the aggregate fault tolerance of the system. Adding layers of cheese is equivalent to increasing the fault tolerance of the system. Reducing the size of the holes, or elimination of the holes, is equivalent to fault reduction or fault elimination, respectively.

Most of the elements of the certification filter model are the same as the elements of the Swiss cheese model. The layers of Swiss cheese become the *filters*.

Definition 3.4. Filter. *In the context of the filter model, a filter is one of the different checks that comprise the certification mechanism.*

These filters correspond to the stages in the system examination process described above. Holes in the filters represent faults, just like holes in the Swiss cheese.

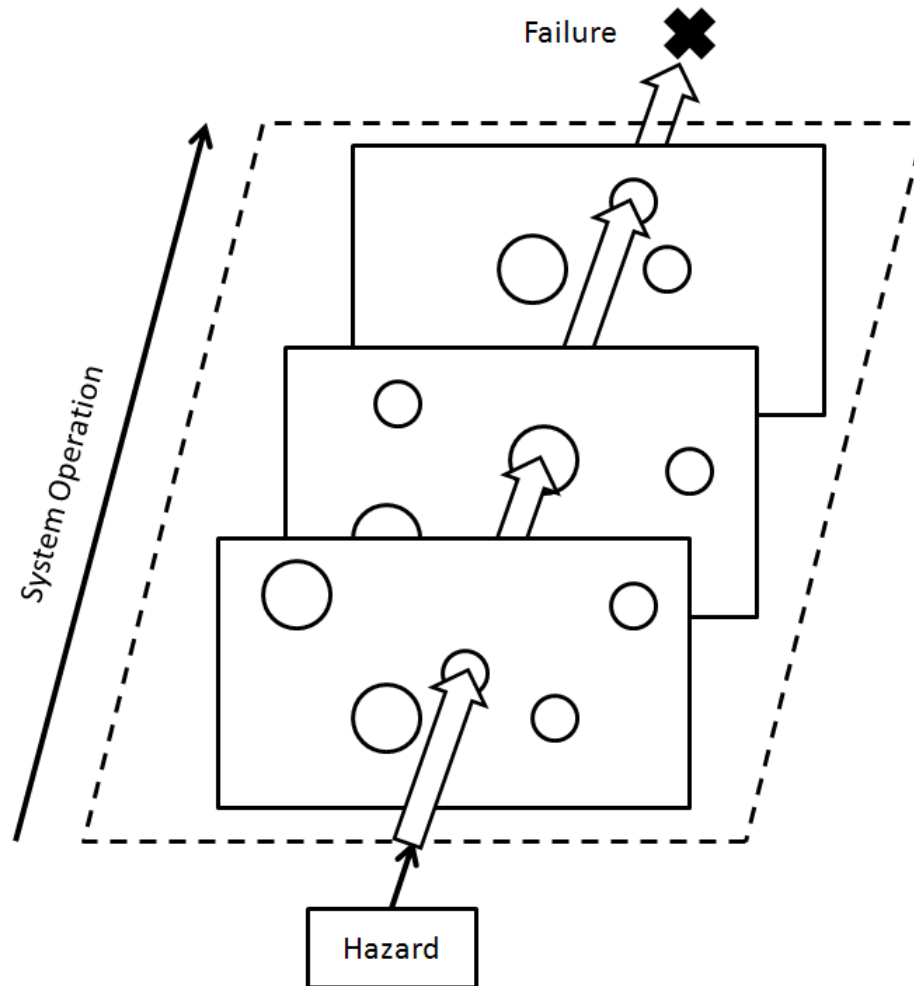


Figure 3.2: Reason's Swiss-cheese model of system failure.

However, the Swiss cheese model of the system assumes that the system is trying to prevent *everything* from passing through the layers of cheese to the other side, as the model is concerned with hazards, faults, and failures. In the filter model, a satisfactory system *should* pass through the filtration system, so a pre-determined way to pass each filter is necessary. This difference is shown in Figure 3.3. Figure 3.3 shows one warranted outcome of the certification process: warranted acceptance. The doors represent the correct way that a system should undergo a particular check during the filtering process, and the system passing through an opened door indicates that the system was deemed satisfactory for that particular check.

Figures 3.4 through 3.6 show the other outcomes. In Figure 3.4, the candidate system is unsatisfactory and is correctly rejected. The second door is closed, indicating that the certifier performed the second check correctly and rejected the candidate system on grounds of that result. Figure 3.5 shows the first unwarranted outcome of certification: an unsatisfactory system that is nevertheless accepted as a satisfactory system. The

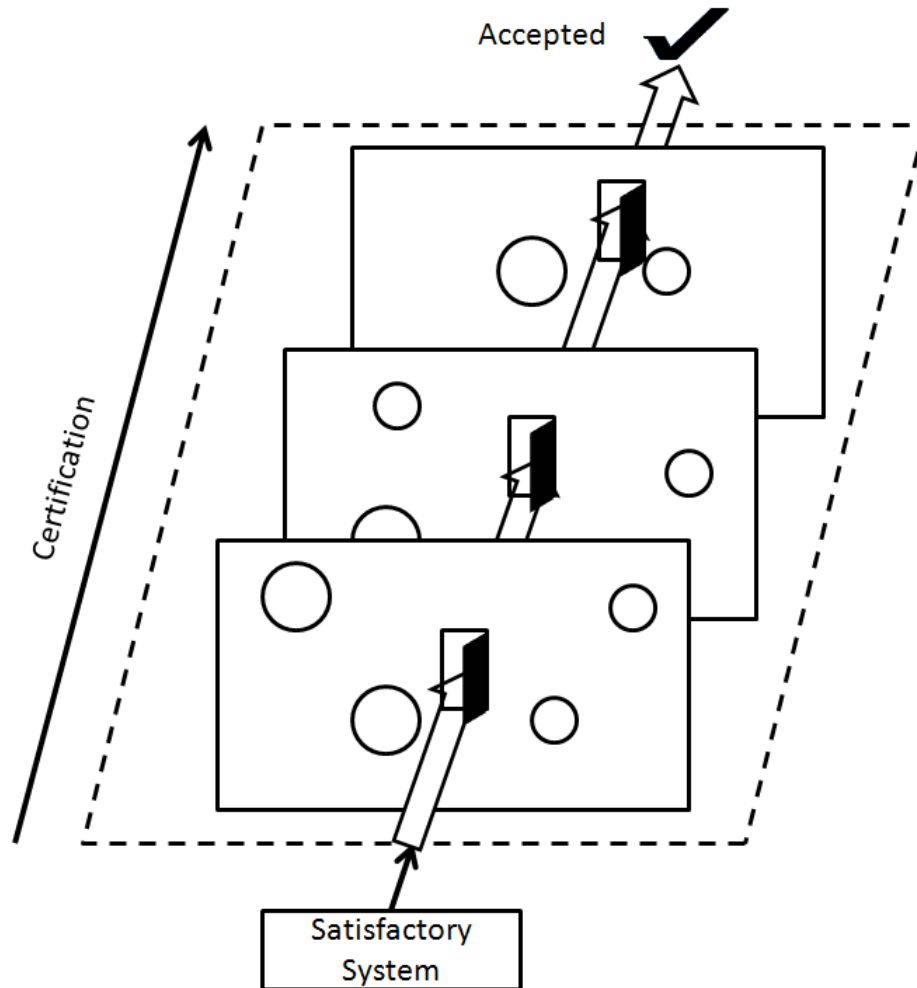


Figure 3.3: The filter model of certification, with a satisfactory system as input and a warranted acceptance as output.

system correctly passes the first check. However, instead of being correctly rejected by the second check, the system is able to bypass the correct execution of the check through a fault in the check. The system then proceeds to correctly pass the third check and be accepted as a certified system.

An exaggerated example of this would be a certifier forgetting to check a critical verification document that should contain proofs of worst-case execution time (WCET) for various calculations performed in a pacemaker. If the proofs are incorrect, then the system is unsatisfactory and should not have been certified. Despite being an exaggerated example, certifiers neglecting or forgetting to examine various documents is not entirely outside of the realm of possibility. A typical certification submission package contains thousands of pages of documentation, which must be examined and decided on within a short time period, e.g., 60 days. Coupled with the requirement of examining multiple submissions in a given 60-day period, certifiers might have approximately ten days to examine a single submission.

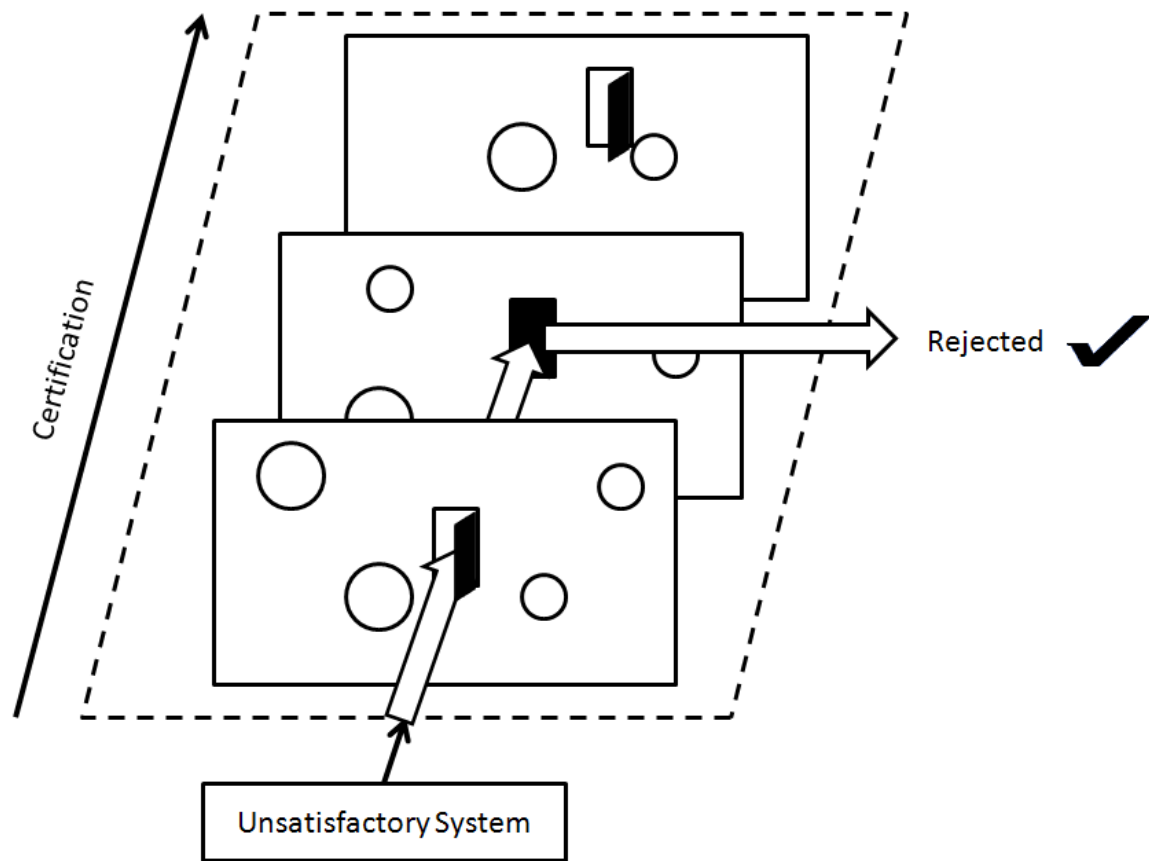


Figure 3.4: The filter model of certification, with an unsatisfactory system as input and a warranted rejection as output.

Examining **unwarranted acceptance** is the focus of this work. While unwarranted rejection is still a certification failure, an uncertified system will not be deployed and thus cannot cause harm. Finding the probability of unwarranted rejection is a subject for future work, discussed in Chapter 10. For completeness, Figure 3.6 shows unwarranted rejection of a satisfactory system. The system correctly passes the first check. However, instead of being correctly accepted by the second check, the system runs up against a part of the filter that is neither the door nor a fault. This represents an overly stringent requirement that the system, while being satisfactory, did not fulfill. Thus, the system is rejected.

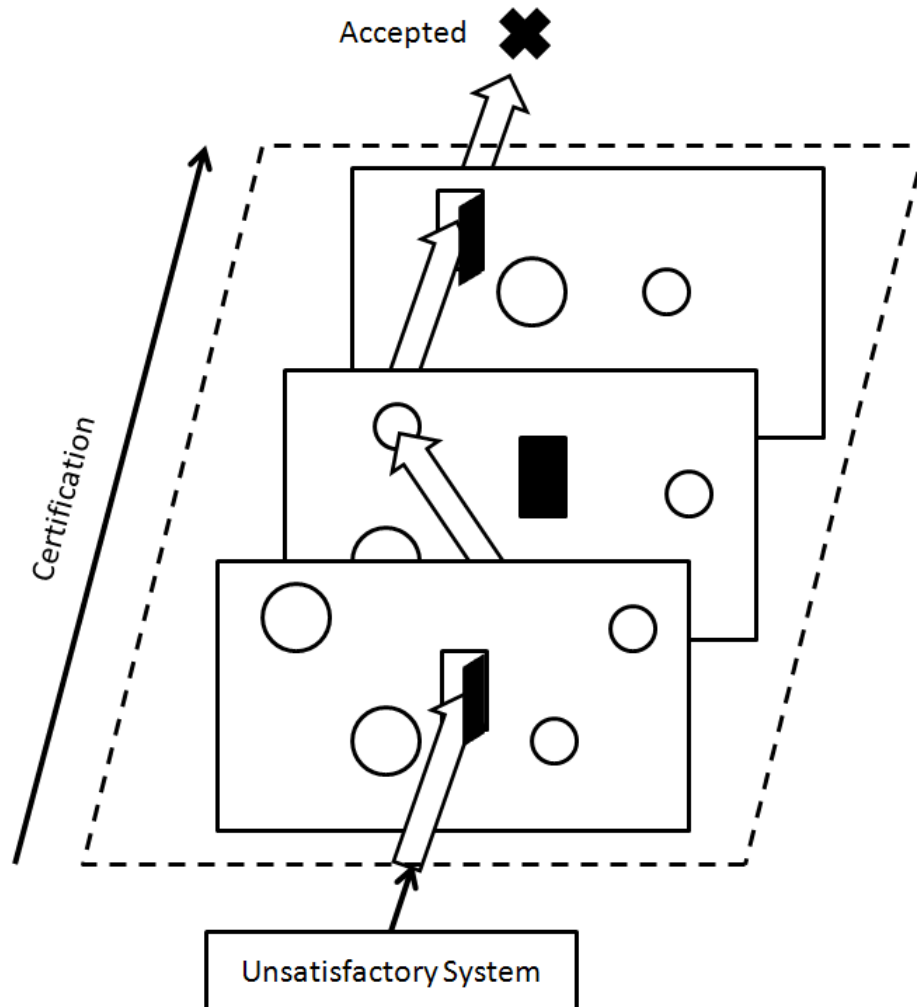


Figure 3.5: The filter model of certification, with an unsatisfactory system as input and an unwarranted acceptance as output.

3.2 System Safety Analysis

The similarity of system failure and certification failure raises the question: can existing techniques for reducing *system* failures be used to reduce *certification* failures, particularly unwarranted acceptances? Recall the thesis statement:

Existing hazard analysis techniques can identify faults in certification systems, which can then be mitigated by existing safety engineering techniques.

Figure 3.7 illustrates the thesis statement. Note that, as both the left- and right-hand sides of Figure 3.7 are the **same type**, i.e., they are both filtration systems, we can apply the safety engineering **iteratively**. The term for this continual repair of the certification mechanism in this work is the *filter repair cycle*.

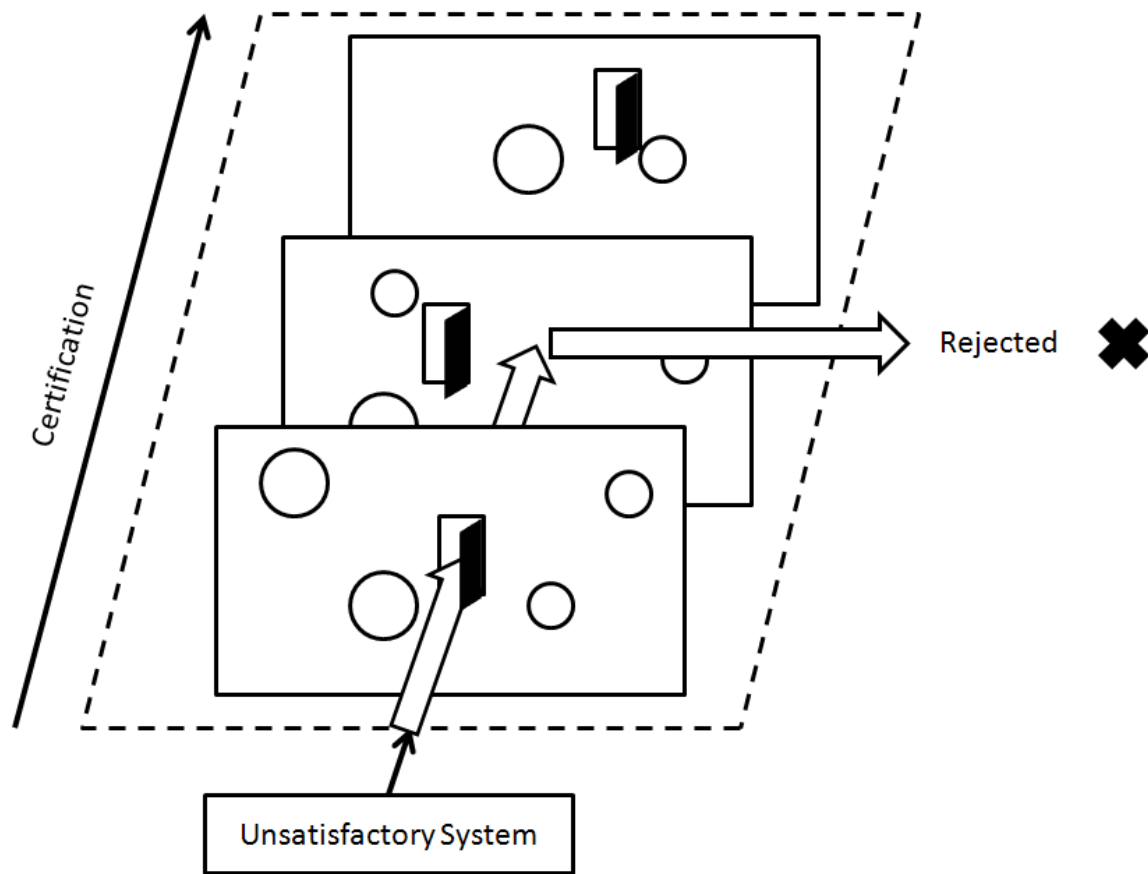


Figure 3.6: The filter model of certification, with a satisfactory system as input and an unwarranted rejection as output.

Definition 3.5. Filter repair cycle. *The filter repair cycle is the iterative application of safety engineering to a particular certification mechanism, in order to discover as many faults as possible and mitigate those faults.*

This section provides a brief background in established hazard analysis and safety engineering techniques.

3.2.1 Discovering Faults

Various hazard analysis techniques provide information about hazards; this information contributes to adjudging the cause of the hazards, i.e., discovering the faults that lead to hazards before corresponding failures happen. For discovering faults_{cert}, this work uses three popular techniques: fault tree analysis [34], failure modes and effects analysis [35], and hazard and operability studies [36].

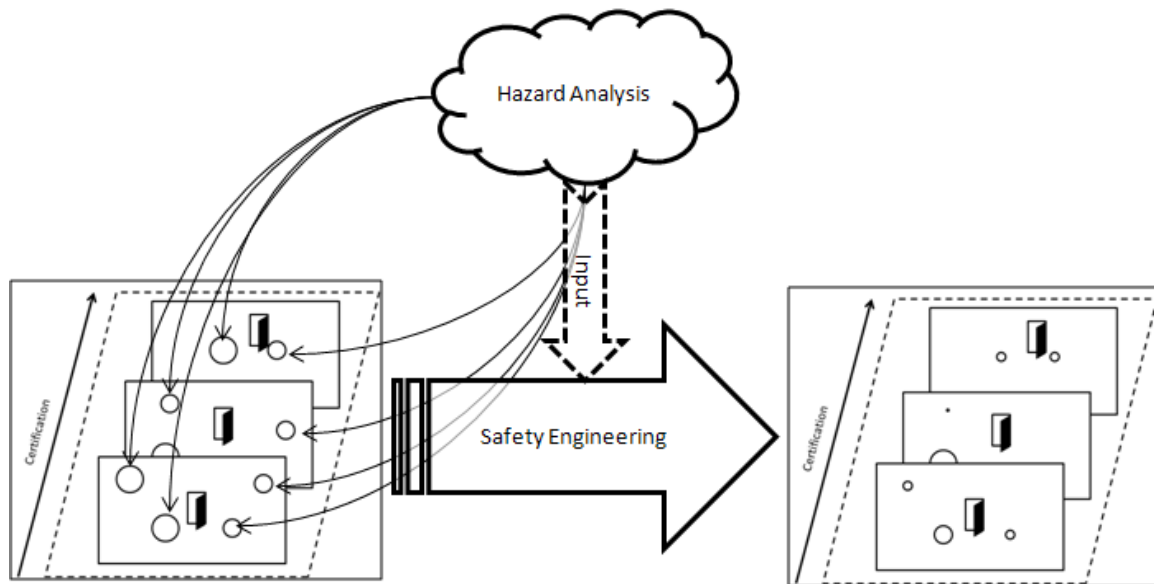


Figure 3.7: An illustrated version of the thesis statement.

Fault Tree Analysis

Fault tree analysis (FTA) is a top-down method of fault identifications. Analysts start with a particular hazard and work backward, determining the events that could lead to the hazard by recursively breaking down the possible events leading up to the hazard. Intermediate events are broken down using Boolean logic operations (most commonly AND and OR). Based on the level of detail needed, analysts conclude the event breakdown at basic events.

Typically, FTA is done graphically, although fault trees can also be represented textually. The most widely-used notation is similar to logic diagrams, which makes the fault trees easy to understand for anyone familiar with electronic circuits. Figure 3.8 shows an example fault tree for the software in a hypothetical morphine infusion pump. The fault tree breaks down the *compound event*, “Infusion pump delivers lethal dose of morphine.” This event is broken down into three contributing events. These are linked together with a *boolean OR gate*, indicating that the infusion pump will deliver a lethal dose of morphine if *any* of the contributing events occur.

Since the fault tree is directed toward the *software* in the infusion pump, “hardware failure” is regarded as a *basic event*, i.e., one that the fault tree does not break down any further. “User error” can be caused by poor user interface design and implementation, which is a software problem; thus, the “user error” element in the diagram is a *transfer* to another fault tree. Lastly, “software failure” is composed of two basic events that are linked together by a boolean OR gate.

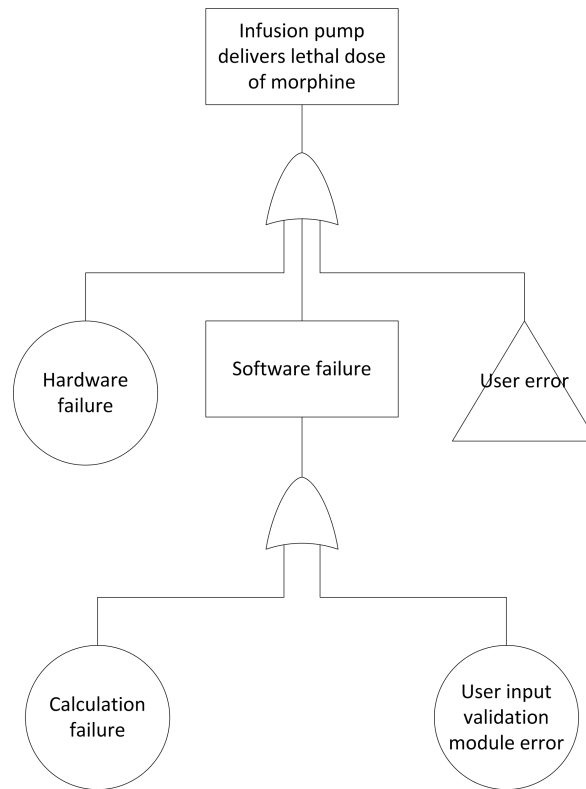


Figure 3.8: Example fault tree for the software in a hypothetical morphine infusion pump.

Note that the example fault tree shown in 3.8 is (a) hypothetical and (b) incomplete. As such, the example does not show all the notational intricacies possible in industrial FTA.

Failure Modes, Effects, and Criticality Analysis

Failure modes and effects analysis (FMEA) is a bottom-up method of discovering hazards. FMEA is often extended to Failure Modes, Effects, and Criticality Analysis (FMECA). In FMECA, analysts assess criticality information for each effect and include the information in the chart.

In FMEA, analysts examine individual components or functions within the system, in order to identify and investigate the failure modes of each component. Analysts consider possible causes and assess likely effects of each failure mode. Failure mode effects are assessed at both the component and system level. If the analysis is a FMECA, the analysts also classify the effects on a criticality scale. Optionally, analysts can suggest possible solutions.

Table 3.1 shows a simple, incomplete example of a FMECA table. Each row examines one failure mode for a particular component, elaborating the characteristics of the failure mode, the effects of the failure mode,

Component	Failure Mode	Effects	Criticality
Calculation module	Overflow error	Morphine dosage is set to minimum negative number, resulting in massive underdose.	Low; probability of patient death due to underdose is low.
	Underflow error	Morphine dosage is set to maximum positive number, resulting in massive overdose.	Extremely high; probability of patient death due to overdose is proportional to magnitude of overdose.

Table 3.1: Hypothetical FMECA results for the calculation module in a morphine infusion pump.

and the criticality of the effects. In this incomplete example, the FMECA examines the calculation module of the hypothetical morphine infusion pump. A complete FMECA would identify and examine as many failure modes as possible.

FMEA and FMECA are often used to provide input data for FTA; the two methods are complementary.

Hazard and Operability Study

A hazard and operability study, or HazOp, is an algorithmic method of discovering hazards. The HazOp technique was originally developed for hazard analysis in chemical and process control plants, but is used in a wide range of domains, including software [37, 38].

A HazOp analysis generates questions about a system based on parameters and guide words. Parameters are based on system components or attributes. The standard guide words are shown in Table 3.2 along with the meaning of each guide word.

HazOp questions take the general form of, “What happens when [*parameter*] [*phrase involving guide word*]?” For example, if *parameter* is “memory” and *guide word* is “more,” then a logical question to generate is, “What happens when the system uses memory more than specified?” or, phrased more clearly, “What happens when the system uses more memory than specified?”

HazOp question generation crosses all parameters with all guide words, so some of the resulting questions may be meaningless. Nevertheless, the answers to meaningful questions often directly provide hazards, which can be used as input to FTA. If not, the answers identify areas for further investigation.

3.2.2 Fault Mitigation

Given hazards and adjudged faults, safety engineering techniques and principles can protect against corresponding failures. To protect against accidents_{cert}, this work uses analogs of fault mitigation techniques. Ordinarily, engineers apply each of the following techniques in succession:

Guide Word	Meaning
No or Not	Complete negation of the design intent
More	Quantitative increase
Less	Quantitative decrease
As well as	Qualitative modification/increase
Part of	Qualitative modification/decrease
Reverse	Logical opposite of the design intent
Other than	Complete substitution
Early	Relative to the clock time
Late	Relative to the clock time
Before	Relating to order or sequence
After	Relating to order or sequence

Table 3.2: Standard HazOp guide words and their meanings.

1. **Fault avoidance.**
2. **Fault elimination.**
3. **Fault tolerance.**
4. **Fault forecasting.**

That is, engineers start with the application of fault avoidance. After applying fault avoidance techniques to the extent possible, engineers apply fault elimination techniques, then fault tolerance techniques, and finally fault forecasting techniques.

Chapter 8 discusses the difficulty of adapting these techniques to a novel area — certification — in more detail. Due to these difficulties, this work focuses on:

- Analogs of fault tolerance techniques, drawing specifically from **redundancy**.
- Analogs of fault-specific methods.

The following sections briefly introduce redundancy and fault-specific methods.

Redundancy

Fault tolerance is achieved through some form of redundancy. Different types of redundancy include:

- Hardware redundancy, e.g., triple modular redundancy to protect against random component failures.

- Software redundancy, e.g., recovery blocks to provide alternative software routines in the event of a software failure.
- Information redundancy, e.g., Hamming codes to detect and possibly correct transmission errors.
- Temporal redundancy, e.g., repeating calculations to detect and possibly ignore transient faults.

Fault-Specific Methods

Certain classes of faults have specific methods for mitigating or even eliminating their occurrence. For example, the usage of the Ada programming language [39] eliminates entire classes of faults that are possible in the standard C programming language [40]. In general, such methods are often found in prescriptive standards.

Faults_{cert} are at another level of abstraction, but similar techniques apply. For example, misunderstanding of domain-specific terminology used in a safety case can be mitigated by comprehensive and highly accurate glossaries.

3.3 Summary

Viewing a given certification mechanism as a system allows for systematic application of proven engineering techniques for fault detection and mitigation.

The next chapter describes how I evaluate fault detection and mitigation for certification systems.

Chapter 4

Evaluation Overview

This work evaluates the feasibility of the filter model with a case study. This chapter summarizes the structure and results of the case study. Chapters 5 and 6 elaborate on the components of the case study and Chapters 7 and 8 elaborate on the results.

4.1 Introduction

Recall the thesis statement from Chapter 1:

Existing hazard analysis techniques can identify faults in certification systems, which can then be mitigated by existing safety engineering techniques.

This evaluation uses a case study to demonstrate the two main points of the thesis:

- Existing hazard analysis techniques can identify faults in certification.
- Certification faults can be mitigated by applying safety engineering techniques.

The case study examines a **specimen certification mechanism** in two phases:

1. First, the certification mechanism is examined for faults_{cert} using hazard analysis.
2. Second, fault treatment is applied to the faults_{cert} discovered in the certification mechanism.

The case study uses the Graydon-Knight Green mechanism (GKG) for its certification mechanism, and for its target system, a novel system developed at the University of Virginia: the Diabetes Advanced Information System (DAIS). DAIS is the first of a new class of systems called a **safety-enhanced patient environment**, or SEPE. Chapter 6 defines SEPE and discusses the distinction between DAIS and SEPE.

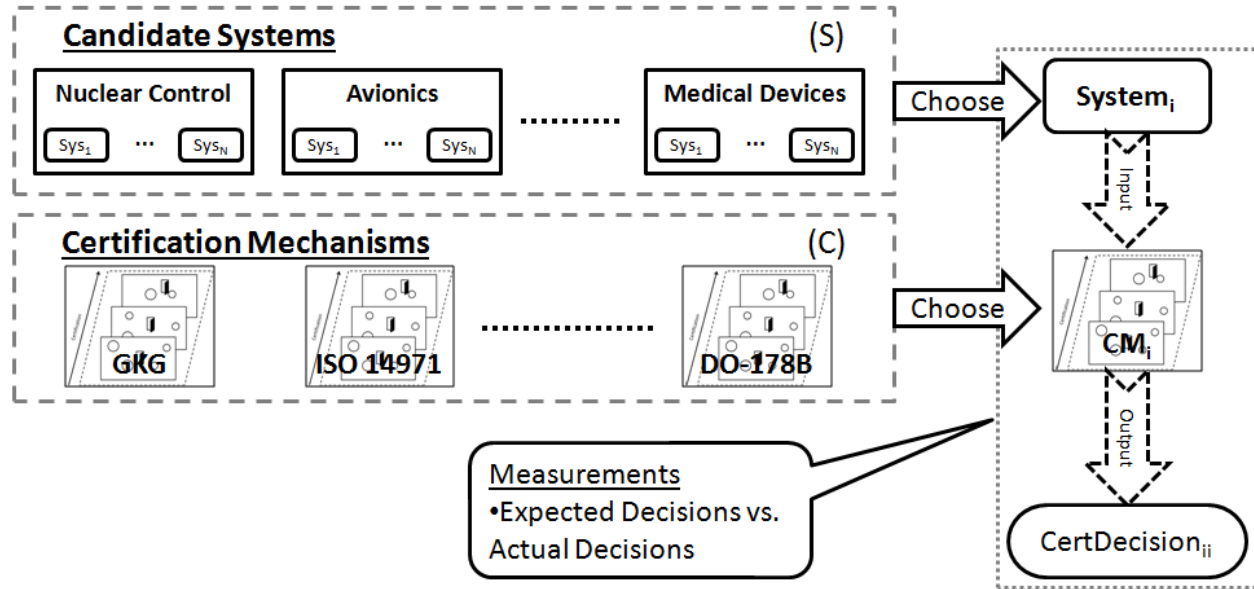


Figure 4.1: An ideal evaluation, comprising systems across many domains.

4.2 Evaluation

Section 4.2.1 presents one design for a comprehensive empirical evaluation of filter model analysis. However, such a comprehensive analysis is out of the scope of this work; Section 4.2.2 presents the design of the case study used in this work.

4.2.1 Ideal Evaluation

Given unlimited resources, an important property to measure about a certification mechanism is its probability of unwarranted acceptance. Ideally, a comprehensive empirical evaluation of a certification mechanism would use a significant number of systems across many domains. All such systems would need to have the following properties:

1. Safety-critical.
2. A member of a class of systems that resides primarily in the regulatory purview of one certifying agency.

In addition, examining multiple certification mechanisms would be beneficial for comparison purposes. Figure 4.1 shows the interaction of the many systems needed for an ideal evaluation. Figure 4.1 shows S system domains, each with N candidate systems, for a total of $S \times N$ candidate systems. In addition, the figure shows C certification mechanisms. Each experiment in the ideal evaluation chooses one certification

mechanism, CM_i , to evaluate and one candidate system, $System_i$, with which to evaluate CM_i . The certification outcome of $System_i$ being processed by CM_i is $CertDecision_{ii}$. The researchers conducting this ideal evaluation know the *expected decision* on each candidate system, i.e., whether each candidate system *should* be certified. To identify the probabilities of unwarranted acceptance and unwarranted rejection for, researchers compare the expected $CertDecision_{ii}$ to the actual $CertDecision_{ii}$.

This structure would imply that C certification mechanisms each examine $S \times N$ candidate systems, for a total of $C \times S \times N$ experiments. However, not every CM fits each candidate system domain; for example, the intended application of ISO 14971 is to *medical devices* [28]. In addition, some standards require the usage of other standards; for example, ISO 13485 requires the usage of ISO 14971 for risk management practices [41]. Thus, the amount of experiments is less than $C \times S \times N$ in practice.

Nevertheless, even one of these experiments is extremely costly. Examining one official certification mechanism requires the cooperation of a government regulator, and examining one industrial candidate system requires the cooperation of that system's developer. Given the resources available, this ideal analysis is not practical. Moreover, the safety cases (including all relevant documents, e.g., source code, software/hardware specifications, etc.) of industrial safety-critical systems are almost never released; they are regarded as trade secrets. The prohibitive expense of such an evaluation necessitates a more practical, small-scale approach.

4.2.2 Practical Evaluation

In lieu of directly measuring the probability of unwarranted acceptance, the case study evaluation presented in this work evaluates the feasibility of the filter model. In particular, the filter model enables the application of systematic *safety engineering* to certification mechanisms. Candidate system safety engineering can be divided into two parts: *hazard analysis* and *fault treatment*. Thus, this evaluation is structured around these two parts.

4.2.3 Aims

The case study shows the feasibility of the filter model through two aims:

Case Study Aim 4.1. *Hazard analysis.* Identify $faults_{cert}$ in GKG.

Case Study Aim 4.2. *Fault treatment.* Propose plausible $fault_{cert}$ mitigation strategies.

The case study is organized into two phases, each of which is designed to satisfy one aim. Figure 4.2 shows the overall structure. Section 4.2.4 elaborates on phase 1 and section 4.2.5 elaborates on phase 2.

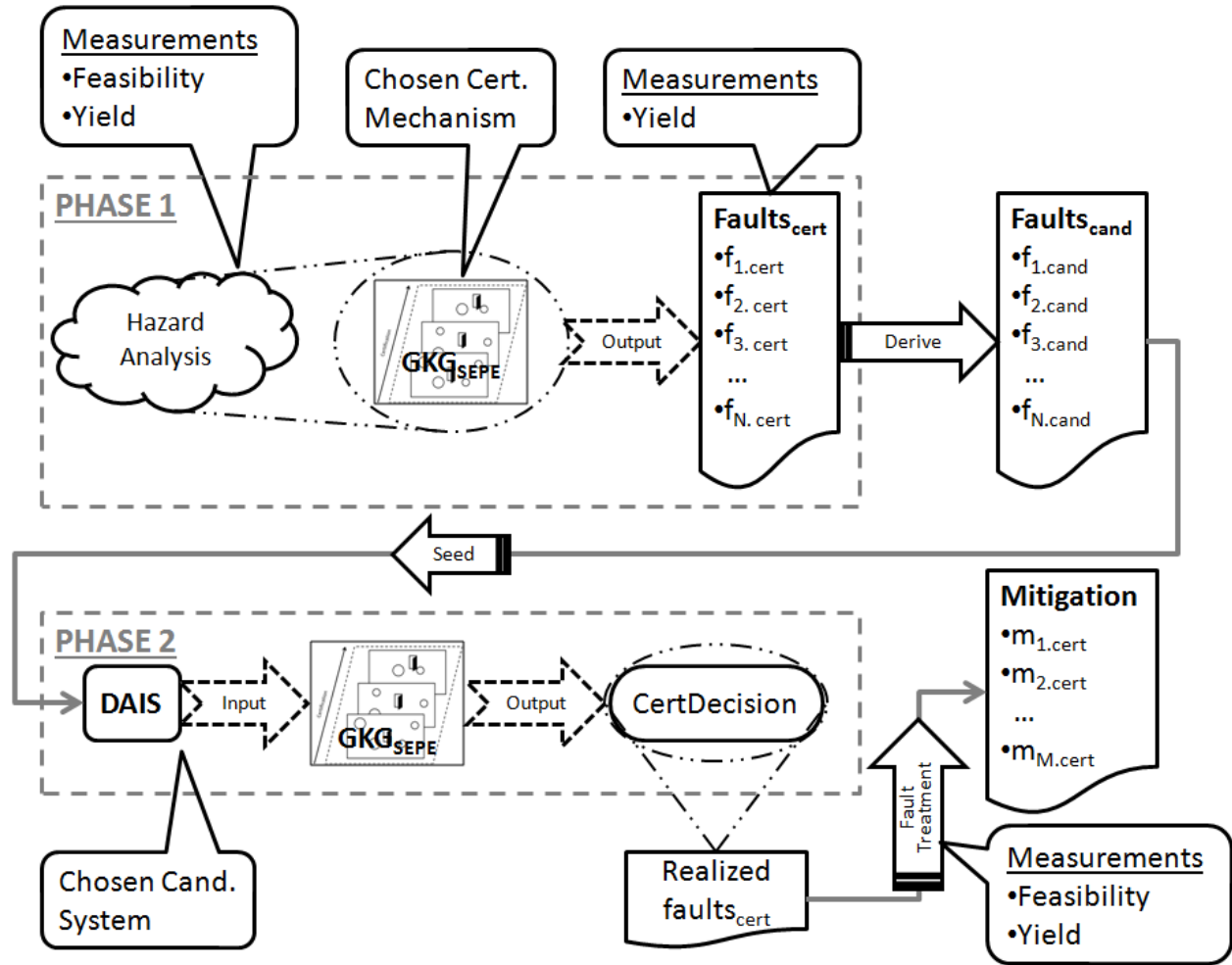


Figure 4.2: The two-phase structure of the case study evaluation used in this work.

4.2.4 Phase 1 - GKG Hazard Analysis

The first phase adapts and applies three *hazard analysis* techniques to the specimen certification mechanism. The specimen certification mechanism is an **instantiation** of the Graydon-Knight-Green certification mechanism; the instantiation is named GKG_{SEPE}. The three hazard analysis techniques used are hazard and operability study (HazOp), fault tree analysis (FTA), and failure modes, effects, and criticality analysis (FMECA). The results of phase 1 are two-fold:

- A set of measurements of the *feasibility* and *yield* of the applied hazard analysis techniques.
- A list of faults_{cert} in GKG_{SEPE}.

This section gives an overview on the input and results of phase 1. Chapter 3 gave an overview of HazOp, FTA, and FMECA.

Input - GKG_{SEPE}

Recall that the specimen certification mechanism is the Graydon-Knight-Green mechanism, which this work classifies as a domain-independent or **general certification mechanism**.

Definition 4.1. General certification mechanism. *A general certification mechanism is a certification mechanism that can be adapted to any candidate system domain.*

Related to the idea of a general certification mechanism is the idea of a **specific certification mechanism**.

Definition 4.2. Specific certification mechanism. *A specific certification mechanism is a certification mechanism that is adapted for a particular candidate system domain.*

A certification mechanism must be **specific** to be applied to candidate systems. Some certification mechanisms exist in a specific form, most often in the form of prescriptive standards that apply to a specific domain. For example, DO-178B [23] applies to software used in airborne systems. However, general certification mechanisms, e.g., GKG, must be adapted to a specific domain in order to be applied to candidate systems. This work terms the *result* of this adaptation process a **certification mechanism instantiation**, or simply **instantiation**.

Definition 4.3. Certification mechanism instantiation. *An instantiation of a general certification mechanism is a specific certification mechanism for a particular candidate system domain, derived from the general certification mechanism.*

An instantiation is derived by supplying the general certification mechanism with domain-specific information. As an example, consider a GKG instantiation concerned with software for drug infusion pumps, GKG_{SDIP} . To obtain GKG_{SDIP} , certifiers at the FDA would instantiate GKG with information from the drug infusion pump software domain, e.g., hazards, best practices, etc.

This work uses a GKG instantiation for the safety-enhanced patient environment domain; the instantiation is named GKG_{SEPE} . Chapter 5 elaborates on the characteristics of GKG_{SEPE} .

Results - Measurements

HazOp, FTA, and FMECA have been shown to be feasible and to have positive yield when applied to safety-critical systems, such as nuclear control systems. However, systematic hazard analysis has never been applied to certification mechanisms. Thus, this work provides an assessment of feasibility and yield of hazard analysis application *to certification mechanisms*. Phase 1 develops several metrics to measure feasibility and yield. With respect to these metrics, each technique was found to be feasible and to have positive yield.

Results - GKG_{SEPE} $Faults_{cert}$

Recall from Chapter 3 that hazard analysis techniques do not directly result in faults. Instead, faults are the *adjudged* causes of hazards. For each technique, phase 1 adjudges the $faults_{cert}$ that exist in GKG_{SEPE} . These $faults_{cert}$ are compiled for use in phase 2.

4.2.5 Phase 2 - GKG_{SEPE} Fault Treatment

The second phase adapts and applies several *fault treatment* techniques to GKG_{SEPE} . The overall goal of phase 2 is to generate $fault_{cert}$ mitigation strategies. Thus, the input to phase 2 is the list of $faults_{cert}$ found in phase 1.

To better understand how $faults_{cert}$ manifest themselves in a GKG instantiation and thus generate better $fault_{cert}$ mitigation strategies, phase 2 conducted several *hypothetical* certifications. GKG is a certification mechanism for safety case submissions; thus, each certification examined an *evaluation safety case*. Each of these safety cases was derived from the target system, the Diabetes Advanced Information System (DAIS). Moreover, each safety case was seeded with a $fault_{cand}$ that corresponds to a $fault_{cert}$ in the list from phase 1.

The results of phase 2 are three-fold:

- A set of examples of how unwarranted certification outcomes can occur.
- A list of $fault_{cert}$ mitigation strategies for GKG_{SEPE} .
- A set of measurements of the *feasibility* and *yield* of generating $fault_{cert}$ mitigation strategies.

This section gives an overview of the input and results of phase 2.

Input - GKG_{SEPE} $Faults_{cert}$

The set of $faults_{cert}$ found in phase 1 was used to create a corresponding set of $faults_{cand}$. These $faults_{cand}$ were codified into **faulty fragments**.

Definition 4.4. Faulty fragment. *A faulty fragment is a fragment in a safety argument that represents a $fault_{cand}$ in the corresponding system.*

These faulty fragments were seeded into a *baseline safety case*, SC_{base} . SC_{base} was constructed to reflect an idealized, high-assurance version of DAIS. The safety cases resulting from seeding the faulty fragments into SC_{base} are termed **evaluation safety cases**.

Results - Certification Examples

Phase 2 conducted a hypothetical certification on each evaluation safety case and compiled the results. Each certification provided one example of the outcome of a GKG-based certification effort.

Results - Fault_{cert} Mitigation

Based on the results of the hypothetical certifications, phase 2 generated several fault_{cert} mitigation strategies. These strategies were adaptations of fault_{cand} treatment strategies for *candidate systems*. This evaluation used primarily fault elimination and fault tolerance.

Results - Measurements

Fault mitigation methods have been shown to be feasible and to have positive yield when applied to candidate systems. However, systematic fault mitigation has never been applied to certification mechanisms. Thus, this work provides an assessment of the feasibility and yield of generating fault_{cert} mitigation strategies for *certification mechanisms*. Note that this work does *not* make claims about the efficacy of the fault_{cert} mitigation strategies.

4.2.6 Assumptions

The evaluation makes several assumptions for practicality.

Case Study Assumption 4.1. *The baseline DAIS safety case accurately represents a high assurance version of DAIS.*

This evaluation examines the GKG_{SEPE} filter, not the quality of the arguments, which are *inputs* to the filter.

Case Study Assumption 4.2. *Any tools used are fault-free.*

In the same vein, this work does not evaluate tool quality.

Case Study Assumption 4.3. *GKG fragment selection is fault-free.*

The problem of a filter (or filter component) that *changes* submissions is out of the scope of this analysis. At best, such a filter will change the submission in a benign or beneficial way, but negative change is a distinct possibility. Analysing this kind of deficient filter could be addressed by future work.

Case Study Assumption 4.4. *GKG conclusion combination is fault-free.*

A perfect conclusion combination process is assumed for the same reason as the perfect selection process assumption.

4.3 Summary

The following chapters present the pieces of the case study evaluation.

4.3.1 GKG Hazard Analysis

Chapter 5 describes the GKG mechanism in detail and the particular instantiation I use for the GKG analysis.

Chapter 7 describes the results of the GKG analysis in detail.

4.3.2 GKG Fault Mitigation

Chapter 6 describes the candidate system for certification, DAIS. Chapter 8 describes the results of using DAIS to examine the faults found in the hazard analysis. Chapter 8 also proposes the safety engineering techniques that correspond with the faults identified and examined in Chapters 7 and 8, respectively.

Chapter 5

Graydon-Knight-Green Certification

This chapter summarizes the Graydon-Knight Green mechanism (GKG) [42], and describes a partial instantiation of the GKG mechanism for the SEPE domain. GKG is an approach to product certification introduced by Graydon, et al. Recall Kelly’s definition of a safety case:

“A documented body of evidence that provides a **convincing** and **valid** argument that a system is **adequately safe** for a given application in a given environment [emphasis added]” [14].

Kelly uses the words “adequately,” “convincing,” and “valid,” but does not provide an actionable definition of those words. UK Def Stan 00-56 [9] provides a similar set of requirements:

“The Contractor shall produce a Safety Case for the system on behalf of the Duty Holder. The Safety Case shall consist of a structured argument, supported by a body of evidence, that provides a **compelling, comprehensible** and **valid** case that a system is safe for a given application in a given environment.”

The gist of this definition is similar to Kelly’s: “compelling,” “comprehensible,” and “valid” are terms that do not have inherently clear definitions, and are not defined by the standard. These terms are the *only quality metrics* provided to certifiers. In practice, these metrics are extremely difficult to measure; *The Nimrod Review* [43] states that the terms are “too amorphous to inject real rigour and focus into the [certification] process.”

Kelly proposed a four-step process for assurance case review [44]; since a safety case is a specific type of assurance case, the process applies to safety cases. The four steps are:

1. **Argument comprehension.** Arguments are composed of certain *elements*: claims, assumptions, context, argumentation strategy, and evidence. The certifier should check whether he understands the essential *elements* of the argument.
2. **Well-formedness checks.** Arguments can exhibit obvious structural defects, including circular arguments and claims that lack supporting evidence. The certifier should check for these obvious structural defects.

3. **Expressive sufficiency checks.** Arguments can often have *implicit* elements, e.g., strategies that are not fully explained. The certifier should assess whether certain elements are sufficiently expressed.
4. **Argument criticism and defeat.** Arguments are usually *inductive* (as opposed to *deductive*) [45]. As such, certifiers must be convinced that the argument is “strong enough” to support the safety claim. In this step, certifiers should *challenge* the argument and evidence by finding and presenting evidence that supports the *opposite* of the safety claim, i.e., that the system is *not* acceptably safe.

GKG was designed to expand Kelly’s recommendations into an actionable definition of “compelling.” The authors’ key insight was to make this definition an *operational* one, i.e., defined by the certification process itself.

5.1 Description

GKG is a certification mechanism for safety case submissions based on an *operational* definition of safety case quality metrics. Certifiers determine whether a system is adequately safe through a *structured dialog* between developers and certifiers. The two portions of the structured dialog are a *phased inspection* and a *dialectic*:

- **Phased inspection.** First, a *phased inspection* examines the safety argument for *essential argument qualities*. Section 5.1.1 expands on the characteristics of the GKG phased inspection.
- **Dialectic.** After the phased inspection is finished, the certifier and developer engage in a *dialectic*. In the dialectic, the certifier levies *challenges* against the safety argument to drive debate with the applicant about the truth of the argument. Section 5.1.2 expands on the characteristics of the GKG dialectic.

This process is the *operational definition* of “compelling” that GKG provides.

Figure 5.1 shows the outline of the GKG process, taken from the original paper. The structured dialog can begin at any point during the system development life cycle. In the structured dialog, certifiers treat the safety argument as a set of **fragments**.

Definition 5.1. Fragment. *A fragment is a small collection of related argument elements that support a single sub-claim.*

When all fragments have been examined and, if necessary, corrected, the certification is deemed complete.

5.1.1 Phased Inspection

Knight and Myers first introduced the idea of a *phased inspection* [46]. A phased inspection consists of multiple *phases*, each of which is a partial inspection that examines for a particular quality of a work product.

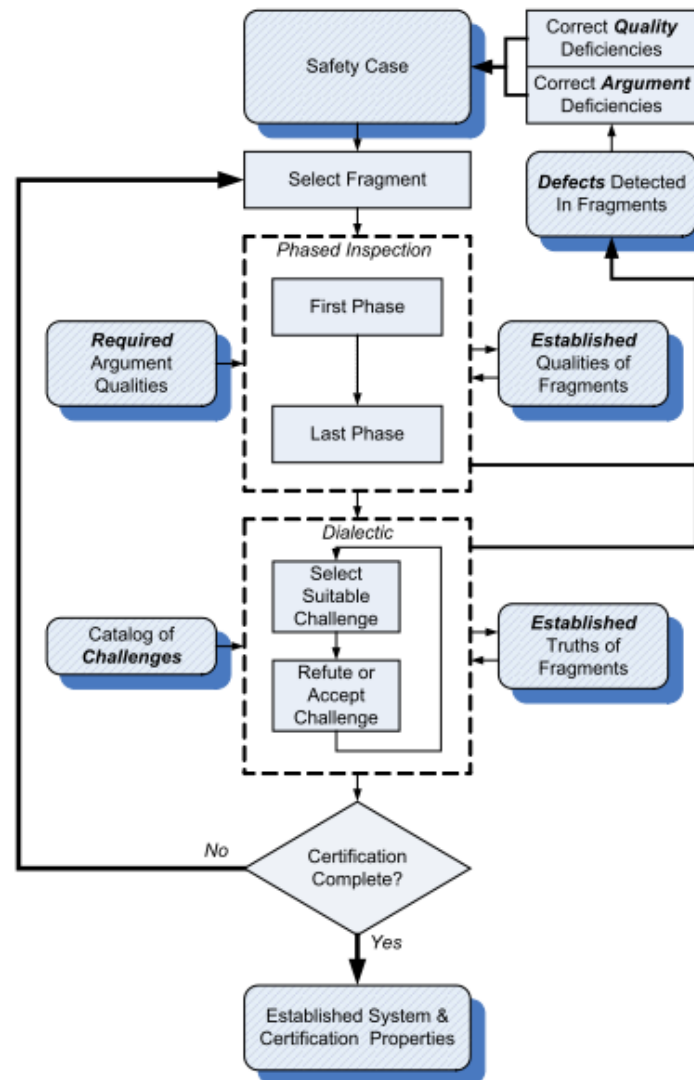


Figure 1. Certification based on a safety case.

Figure 5.1: GKG process outline: certification based on a safety case.

As an example of a work product, consider the source code of a program. In a phased inspection of a source program, the first phase could be devoted to examining layout, the second phase could be devoted to examining coding practices, and the third phase could be devoted to examining functional correctness.

At the end of each phase, inspectors recommend rework for defects related to the quality examined in that phase. When the rework is completed, inspectors can assume that the quality examined in that phase has been established. Thus, *the order of phases in a phased inspection matters*. Inspectors can thus structure a phased inspection to their advantage. Continuing with the previous example, a common coding practice

for which to inspect is initializing all variables at declaration. Assume that the inspection structures the “coding practices” phase, which includes checking for whether variables are initialized at declaration, before a “functional correctness” phase. After completion of the “coding practices” phase (including any necessary rework), inspectors can assume that no variables will be used before being initialized. This assumption allows inspectors in the “functional correctness” phase to focus on whether the code implies the specification, without inspecting for low-level errors.

The phased inspection in the GKG mechanism examines fragments for qualities that Graydon, et al. believe are essential to all safety cases; these qualities are listed below:

1. **Terminology understood identically by all readers.** In order to effectively evaluate a safety case, a certifier should understand the safety case in the same way that the developer does. If the certifier and developer understand a certain term differently and *are unaware of their misunderstanding*, the resulting certification decision could be unwarranted.
2. **Absence of vagueness.** The documents that comprise a safety case should be free from vagueness, as vagueness makes safety assessment difficult. A common source of vagueness is the natural language in the documents.
3. **Document syntactic validity.** A safety case consists of a variety of documents, each of which follows a pre-determined syntax. All documents should have the proper syntax; certifiers should examine documents for *content* and *relevance* to the argument, not *format*.
4. **Evidence availability and sufficiency.** Some of the documents in a safety case contain evidence; these documents are cited in the safety case to support various claims. An evidence citation should be a unique identifier of the cited artifact. Furthermore, the cited artifact should be sufficient to support the claim that the artifact purports to support.
5. **No unnecessary argument elements.** Threads of argument that are unnecessary should be removed. An example of unnecessary argument elements is a fragment that was developed regarding a deprecated part of the candidate system.
6. **Assumption necessity and reasonableness.** Any safety case will make assumptions about various system facets; prominent examples of assumptions are assumptions about operator capability and assumptions about system operating context. Each assumption should be checked to establish whether:
 - The assumption is necessary.
 - The assumption is plausible.
 - The certifier can show that the assumption is false.

If the assumption is plausible and the certifier *cannot* show that the assumption is false, the assumption should be considered reasonable.

7. **Freedom from well-known fallacies.** Fallacious reasoning in a safety argument can result in unwarranted belief in a safety claim, which leads to unwarranted certification. Greenwell, et al. developed a taxonomy of safety argument fallacies [47]. The certifier can examine the argument for the presence of well-known fallacies using the Greenwell taxonomy.

After a fragment in question has been inspected for the necessary qualities, the certifier and developer engage in the *dialectic*.

5.1.2 Dialectic

Chapter 2 established that safety cases cannot be evaluated quantitatively in general. Thus, certifiers must establish the truth of the claim in a fragment, and therefore the truth of the overall safety claim, in a qualitative way. GKG uses the notion of a **dialectic**, which the Merriam-Webster dictionary defines as:

“Discussion and reasoning by **dialogue** as a method of intellectual investigation; *specifically*: the Socratic techniques of **exposing false beliefs** and **eliciting truth**.” [48] [emphasis added]

The dialogue is driven by a catalog of domain-specific **challenges**.

Definition 5.2. Challenge. *In GKG, a challenge is a short statement or question that calls into question the truth of a fragment.*

During the dialectic, the certifier levies relevant challenges against each fragment. The developer must either refute or accept the challenge. Refutation of a challenge requires the developer to convince the certifier that the challenge should be dismissed. Acceptance of a challenge requires that the developer modify the argument in such a way that subsequent levying of the same challenge will result in refutation. The certifier may require modifications that include (but are not limited to):

- Generating additional evidence, e.g., formal verification of critical sections of code.
- Changing the argument, e.g., modification of contexts assumed in the argument.
- Changing the candidate system, e.g., adding redundant hardware components.
- A combination of the above modifications, e.g., changing an argument strategy to include a “formal verification” supporting leg *and* generating the formal verification evidence.

5.1.3 Challenge Categories

Certifiers can derive challenges from the definition of what is “reasonable” in the domain of interest. To help guide challenge derivation, Graydon, et al. identified predetermined challenge *categories*:

1. **Omission of expected practice.** Many critical domains have developed expected practices. Omission of an expected practice could indicate any of the following:
 - The argument has a defect.
 - The argument could benefit from including additional evidence.
 - The candidate system is a special case for which the expected practice does not apply.

In the last case, the developer must convincingly refute the need for the expected practice. This category contains challenges that reflect relevant expected practices.

2. **Common dependence of subarguments.** When a single argument does not provide adequate confidence in a claim, arguments often provide subarguments to increase confidence. These arguments are *intended* to be independent; if a certain circumstance exists that invalidates the independence assumption, the subarguments provide no extra confidence. This category contains challenges that reflect these circumstances.
3. **Negative experience.** Many established system domains have extensive histories of accidents. The foreknowledge of *why* a specific accident occurred can aid in avoiding similar accidents with *new* candidate systems. This category contains challenges that reflect relevant negative experience.
4. **Unrealistic assumptions.** Safety cases must often make assumptions about *candidate system properties*, e.g., component reliability. If these assumptions are incorrect, the argument could be unsound. This category contains challenges that reflect the bounds of realism on assumptions for the domain in question.
5. **Inapplicability.** Safety cases must also make assumptions about the *circumstances* in which the candidate system will be used; these assumptions must hold for the argument to be sound. This category contains challenges that reflect the bounds of applicability of various circumstances relevant to the domain in question.
6. **Inadequate strategy.** Safety argument *strategy* affects the assurance that a safety case can provide. Developers must choose a strategy to argue the safety of the candidate system; however, some strategies provide more assurance than others. This category contains challenges that reflect the certifier's knowledge of both the *presence* of inadequate strategy and the *absence* of adequate strategy.
7. **Improper use of patterns.** Safety case patterns [14, 11] are previously successful argument portions that have been preserved for posterity. Patterns encode the knowledge of engineers that were trying to solve the same problem as the candidate system. Developers can quickly create safety cases from appropriate patterns; however, the selection of patterns and their usage must be correct for the resulting safety case to be effective. This category contains challenges that reflect both *accepted* patterns (and

their usage) and *contra-indicated* patterns.

8. **Inadequate argument strength.** The strength of a fragment is defined by Graydon, et al. as the confidence that a certifier has in a fragment, i.e., how confident a certifier is about the *truth* of a fragment. This category contains challenges that reflect:

- Fragments that are known to be inadequate *in general*.
- Fragments that are known to be inadequate *in particular sets of circumstances*.

Other challenges could be derived based on the domain of interest, but most challenges would fall under one of the above categories.

5.2 Instantiation

The description of the GKG process throughout Section 5.1 is an *abstract definition* of the GKG process, i.e., every certifier that uses GKG must follow the general idea of:

1. Phased inspection of safety argument qualities.
2. Dialectic to establish truth of fragments through levying challenges and subsequent discussion.

In order to be used in a real certification, a certifier must supply the abstract GKG definition *domain information*: information that will enable certifiers to comprehensively examine arguments for systems in the certifier's purview. Section 5.2.1 discusses how to instantiate the GKG phased inspection, and Section 5.2.2 discusses how to instantiate the GKG dialectic.

5.2.1 Phased Inspection

The literature indicates that these properties are *necessary* for a convincing argument, but not whether these properties are *sufficient*. More properties may be needed for a sufficient set. Domain-specific argument properties may also be necessary for systems in a particular domain to have convincing arguments. The properties listed in Section 5.1.1 are domain-independent, i.e., *all* arguments across *all* certifications should have them. At this point, whether this set is complete (in either sense described above) is an open research question.

Open Question 5.1. *Are the GKG argument properties sufficient for all safety arguments?*

Until this question is answered definitively, a typical GKG instantiation should use the qualities detailed in Section 5.1.1.

5.2.2 Dialectic

The dialectic requires most of the domain information, because the challenges are domain-dependent. The description of the challenge categories in Section 5.1.3 implies sources for challenge derivation; these sources are much more easily explained through the use of an example. Section 5.3 provides an example of a GKG instantiation for the SEPE domain, discussed in Chapter 6. Section 5.3.2 describes example SEPE challenges for each challenge category.

5.3 SEPE Instantiation of GKG

This section introduces a *partial* instantiation of GKG for the SEPE domain. The rest of this work refers to this instantiation as “GKG_{SEPE}.” The evaluation of the filter repair cycle uses GKG_{SEPE} as its subject certification mechanism.

As in Section 5.2, any GKG instantiation must supply enough information to conduct certifications in a particular domain. The following sections describe the domain information used in creating GKG_{SEPE}. Note that the novelty of SEPE lies in the new *application* of existing technologies, specifically software. Thus, all software safety/quality standards apply; these standards are leveraged in the domain instantiation.

5.3.1 Qualities

Since the argument qualities, to the best of our knowledge, are domain-independent, GKG_{SEPE} includes them as in Section 5.1.1. The qualities are reproduced below.

1. Terminology understood identically by all readers.
2. Absence of vagueness
3. Document syntactic validity.
4. Evidence availability and sufficiency.
5. No unnecessary argument elements.
6. Assumption necessity and reasonableness.
7. Freedom from well-known fallacies.

5.3.2 Challenges

Because of the novel nature of the SEPE domain, the predefined challenge categories require some special care to use. Below is the list of categories, along with some example challenges. These example challenges

form an incomplete set of challenges for GKG_{SEPE} , in order to evaluate the filter model using GKG_{SEPE} as the certification mechanism.

The list of predefined challenge categories from GKG is:

- Omission of expected practice.
- Common dependence of subarguments.
- Negative experience.
- Unrealistic assumptions.
- Inapplicability.
- Inadequate strategy.
- Improper use of patterns.
- Inadequate argument strength.

The following sections expand on each of these categories in detail.

Omission of Expected Practice

As a novel domain, SEPE does not have any *new* expected practices. However, **prescriptive standards** include a wealth of expected practice for various types of software systems. Each of the following well-understood system domains applies in part to SEPE:

- Distributed systems [49].
- Safety-critical systems [23].
- Mobile device systems [50].
- Web services [51].
- Medical systems/devices [28].
- Software that must interoperate with external systems (both software and hardware) [52].

As shown in Figure 5.2, SEPE is at the intersection of all of those domains. Note that the fully intersecting Venn diagram shown in Figure 5.2 is *qualitative* in terms of the degree to which the various system domains overlap with each other. Additionally, SEPE may develop its own expected practices that do not derive from expected practices in other domains. The diagram is intended to convey the relationship the domains have to SEPE.

Note the important distinction between *using some of the provisions* of a prescriptive standard as *source material* for challenges and *adhering to a prescriptive standard for certification*. Recall from Chapters 1 and 2 that prescriptive standards rely on the unjustified **adherence assumption** instead of explicitly showing

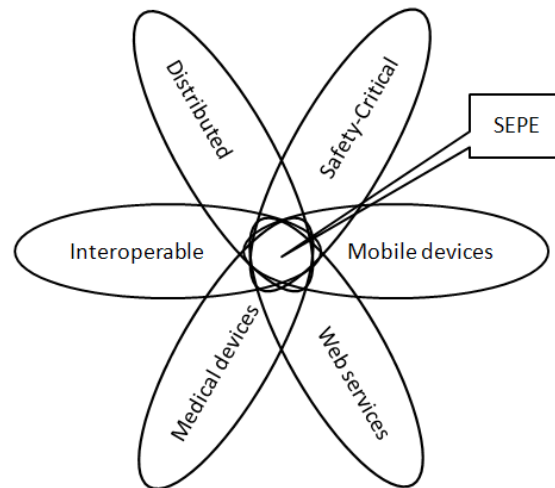


Figure 5.2: Well-understood system domains that contributed expected practices to SEPE.

dependability. However, using individual standards provisions as source material for challenges allows certifiers to pose challenges of the following form to developers:

Challenge. “Explain why you did not follow [expected practice] from a standards provision.”

[Expected practice] is a general placeholder for any given expected practice for which the certifier issues a challenge. In the ensuing dialectic, the certifier and developer can arrive at a decision about whether the developer should have included [expected practice]:

- [Expected practice] was necessary for this particular candidate system, so the certifier will not issue a positive certification decision until [expected practice] is fulfilled. For example, the FDA has issued guidance on off-the-shelf (OTS) software use in medical devices [52]; one provision states, “Provide the results of the testing [of the OTS software].” In most candidate systems that incorporate OTS software¹, providing testing results is a prudent decision; OTS software often does not come with guarantees of dependability, forcing developers of candidate systems to derive this dependability themselves.
- [Expected practice] was not necessary for this particular candidate system, so the certifier will issue a positive certification decision. Continuing with the previous example, *not* providing the results of OTS software testing would be justifiable if the OTS software provided a legal guarantee of high reliability from the OTS software developer. Hypothetically, a legal guarantee of sufficient reliability would

¹From a naive point of view, one might argue that not many systems have a large OTS software component. However, third-order programming languages are either (a) compiled or (b) interpreted; compilers and interpreters are almost invariably OTS software. So, the vast majority of candidate system that incorporate any software *at all* contain or depend on OTS software.

convince the FDA certifier that the OTS software did not require testing to be included as OTS software in another device.

Given the source of these challenges – *all* relevant prescriptive standards – developing a comprehensive set of challenges for any GKG instantiation is beyond the scope of this work. However, the analysis requires some challenges in order to approach accurate simulation of an instantiated GKG mechanism. Challenges 5.1 through 5.4 reflect standards provisions. Each challenge provides a reference to the standard from which the challenge is derived.

Challenge 5.1. *“Explain why you did not provide a testing report for this OTS software.” (FDA Off-The-Shelf Software Use in Medical Devices [52])*

Challenge 5.2. *“Explain why you did not provide a formal specification for your software.” (IEC 61508 [53])*

Challenge 5.3. *“Explain why your software data exchanges do not include checksums.” (ISO 14971 [28])*

Challenge 5.4. *“Explain why you did not obtain independent verification and validation (IV & V) for your software.” (NASA-STD-8739.8 [24])*

Note that despite the disparate origins of the standards cited in the example challenges, each standard applies to some domain of software systems, and high-assurance software systems in general.

Common Dependence of Subarguments

Independent subarguments are often used to provide additional assurance where one argument does not provide sufficient assurance. However, if subarguments that are surmised to be independent have a *common dependence*, their assurance is reduced. If the common dependence is, for example, software requirements, then any requirements error will percolate to *both* subarguments, reducing the advantage of the independent subargument construction. The fragment shown in Figure 5.3 illustrates this problem. Hypothetically, the two legs have a high probability of being dependent, as both the formal specification and the test cases in the test plan are derived from the *same requirements analysis*, in most cases. Challenge 5.5 reflects this difficulty.

Challenge 5.5. *“Explain why your verification argument legs are independent.”*

For more on the two-legged construction shown in Figure 5.3, see the below section on **Inadequate Strategy**.

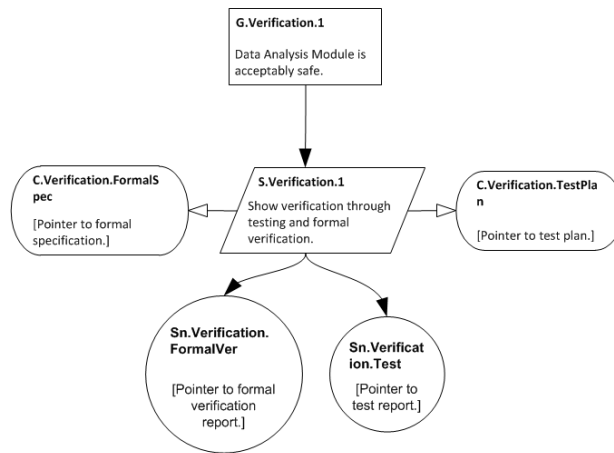


Figure 5.3: A fragment with two potentially dependent subarguments.

Negative Experience

As a novel domain, SEPEs have not accumulated significant negative experience (if any). However, certifiers can again draw from related software system domains. This time, the source material for challenges is different collections of failure data, whether in the form of official report narratives [54, 55] or databases with failure data records [56].

Related SEPE domains (e.g., medical devices) have accumulated extensive collections of failure data. Deriving a comprehensive set of challenges for this category is out of the scope of this work; as in other areas, this work instead provides some example challenges below. Challenges 5.6 through 5.8 reflect negative experience. Each challenge provides a reference to the negative experience from which the challenge is derived.

Challenge 5.6. *“Explain why you did not ensure that converting a variable from a floating point representation to an integer representation will not cause a catastrophic failure.” (Ariane 5 Flight 501 Failure Report [55])*

Challenge 5.7. *“Explain why you did not ensure that all hardware components provide an adequate degradation failure warning at the time of the failure.” (Boeing Company 777-200 9M-MRG In-flight Upset Event Report [54])*

Challenge 5.8. *“Explain why you did not ensure that the software will not crash when a physician tries to access diagnostic data.” (FDA Manufacturer and User Facility Device Experience database [56])*

Unrealistic Assumptions

Any developer must make assumptions about the context and usage of the system under development. These assumptions can vary widely in form and content; especially in a novel domain, it would be impractical to

enumerate all unrealistic assumptions that could be made in the SEPE domain.

An example of an assumption category is **rate of user error**.

Definition 5.3. Rate of user error. *Rate of user error is the average number of user errors per unit time.*

In order to scope the amount of error detection and correction the system must perform, a SEPE developer could assume a bound on the rate of user error. One can conceive of a hypothetical, exaggerated situation in which the developer bases the bound on the *average* rate of user error. This assumption could be too optimistic for the most error-prone patients, making the system unsafe for those patients. Challenge 5.9 reflects this unrealistic assumption.

Challenge 5.9. *“Explain why you chose the value you chose for your bound on rate of user error.”*

Inapplicability

SEPEs imply a certain set of circumstances under which normal operation is justified. Safety arguments rely on the details of these circumstances [42], so defining a clear set of applicable circumstances is important. A general, comprehensive set of criteria for what makes a circumstance acceptable is beyond the scope of this work. An example of a criterion is the **age** of the patient. A child under a certain certifier-determined age may not have the motor skills or cognitive faculties to properly use a SEPE. Without loss of generality, assume that the minimum age for a patient to qualify for using a SEPE is fourteen. Challenge 5.10 reflects this inapplicable circumstance of a patient below the age of fourteen using a SEPE.

Challenge 5.10. *“Explain why no patient under the age of fourteen (14) will use this system.”*

Inadequate Strategy

The strategies used in safety arguments affect the maximum overall level of assurance that the safety argument can reflect. Argumentation is not an exact science; there is a human element in *choosing* particular argumentation strategies. Different individuals are often more comfortable arguing using one strategy over another.

However, some strategies do not provide as much assurance as others. Consider the top-level strategies shown in the safety argument fragments in Figure 5.4. The strategy on the left, $S.DAIS.1$, directly addresses hazards; the strategy on the right, $S.DAIS.2$, argues conformance to a standard. In contrast with $S.DAIS.2$, $S.DAIS.1$ provides a much more direct argument that SEPE hazards have been mitigated. Furthermore, $S.DAIS.1$ is a more flexible strategy than $S.DAIS.2$; recall that one of the disadvantages of prescriptive

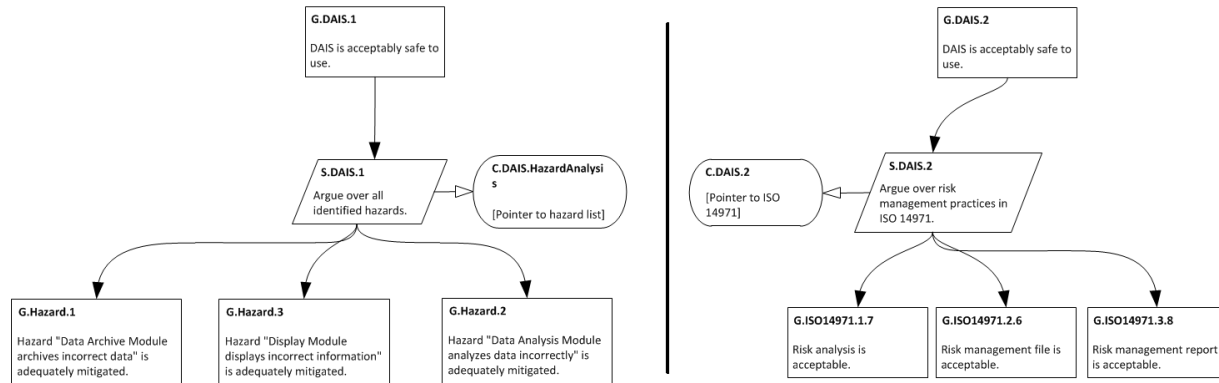


Figure 5.4: Example safety argument fragments exhibiting two differing strategies of top-level argumentation. One strategy argues over all known hazards, while the other argues conformance to a standard.

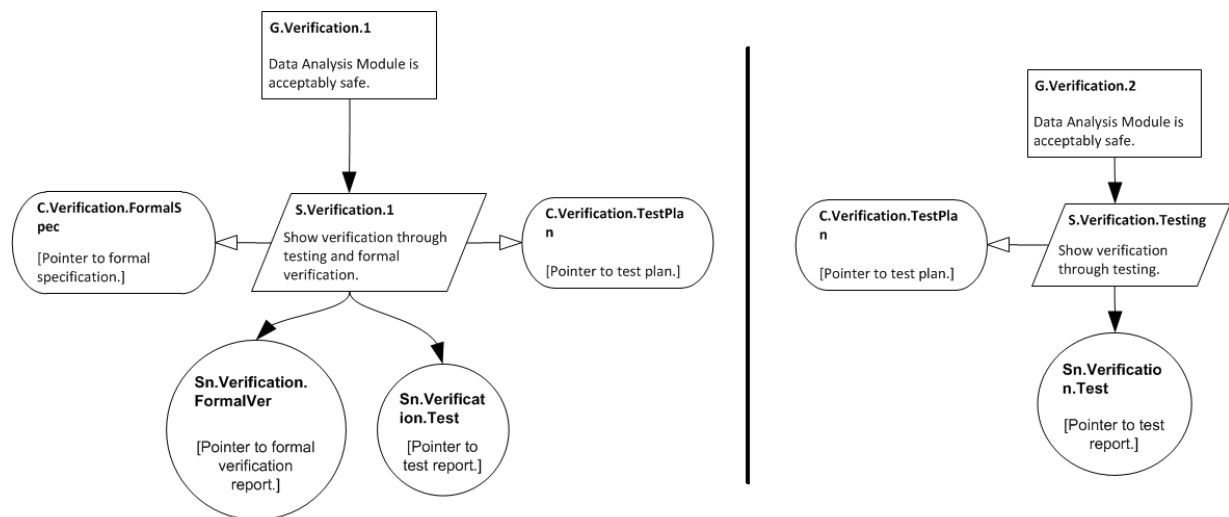


Figure 5.5: Example safety argument fragments exhibiting two differing strategies of arguing about verification. One strategy argues that only testing is sufficient, while the other argues that verification is *also* necessary.

standards is their inflexibility. While *S.DAIS.2* argues conformance to a standard, *S.DAIS.1* allows the developers to choose what they deem to be the best strategy for mitigating each hazard.

Challenge 5.11 reflects the general superiority of arguing over all hazards compared to arguing standards conformance.

Challenge 5.11. “Explain why you based your safety argument on conformance to a standard.”

Challenge 5.11 is a challenge at the highest level of the argument; another useful illustrative example is one at a lower level. Consider the strategies in the fragments shown in Figure 5.5.

The strategy on the left of Figure 5.5, `G.Verification.1`, argues that adequate dependability is shown by both testing and formal verification, while the strategy on the right, `S.Verification.2`, argues that only testing is required for adequate dependability. `S.Verification.1` is a more rigorous verification strategy; formal verification has been shown to be complementary to testing [57, 58]. Moreover, Butler and Finelli [59] showed the infeasibility of employing *only* statistical life testing to show that critical software systems fulfill ultra-high dependability criteria.

UK Def Stan 00-56 [9] refers to the construction including `S.Verification.1` as a **diverse two-legged argument**. Def Stan 00-55 [60] suggests that one leg should be based on statistical testing and another on logical proof of correctness, i.e., formal verification. Thus, this work uses the following specialized definition of a diverse two-legged argument.

Definition 5.4. Diverse two-legged argument. *A diverse two-legged argument is a safety argument fragment that argues adequate verification with two legs: one based on testing and one on formal verification.*

Challenge 5.12 reflects the inadequacy of solely testing as a verification strategy.

Challenge 5.12. *“Explain why you argued that testing alone is sufficient for verification.”*

Improper Use of Patterns

Safety case patterns [14, 11] are an important asset for safety case creation. Patterns encode previously useful safety argument fragments and corresponding usage guidance; however, they must be used correctly to impart the assurance that they are intended to impart. Incorrect usage could lead to, among other things, a confused safety case that is not representative of the system for which it is a proxy.

For example, consider the software safety pattern shown in Figure 5.6, the “Hazardous Software Failure Mode Decomposition” pattern [61]. Weaver states the following about `SWContribIdent`:

This context gives the safety requirements which are related to the software. These can be either through software causes or through **derived requirements due to cross dependencies** [emphasis mine].

Blind application of this pattern without reading the guidance about “cross dependencies” could result in subtle deficiencies in the safety case, where both the developer and certifier miss the existence of requirements generated from cross dependencies. Challenge 5.13 reflects this difficulty.

Challenge 5.13. *“Explain why your safety requirements are all identified, including requirements generated from cross dependencies.”*

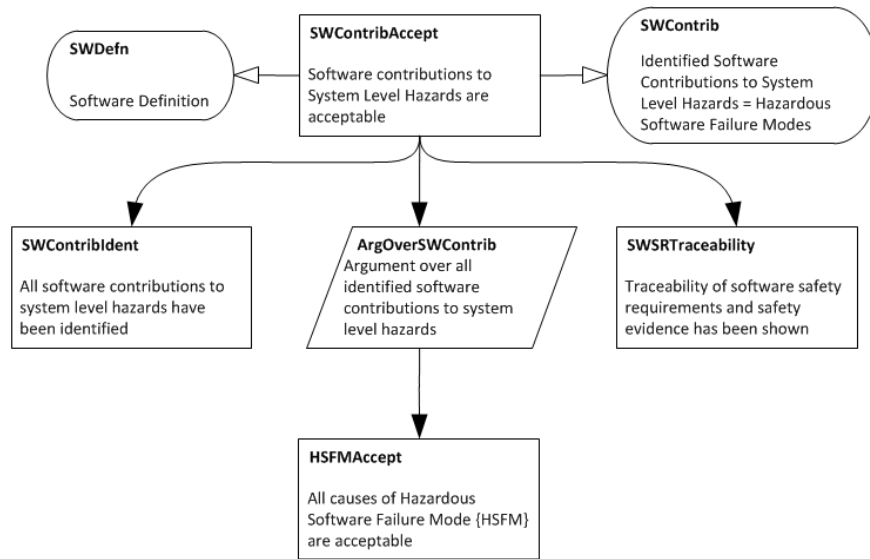


Figure 5.6: The Hazardous Software Failure Mode Decomposition pattern.

Inadequate Argument Strength

The original GKG paper states that the “strength of an argument fragment is basically the confidence that the certifier has in the fragment.” Measuring confidence in safety arguments is area of current work [62, 63, 64, 65], so there is no established answer to the question of how exactly a certifier can measure his confidence in a fragment’s strength. One practical way to approach such a measurement is to allow the certifier to issue Challenge 5.14. Challenge 5.14 reflects the the confidence that the certifier has in the fragment.

Challenge 5.14. *“Explain why this fragment is of adequate strength.”*

5.4 Summary

This chapter summarized the GKG mechanism and provided an example instantiation, GKG_{SEPE}. Chapters 7 and 8 present the results of a case study evaluation of the filter model approach. This evaluation uses GKG_{SEPE} as the subject certification mechanism and DAIS as the specimen candidate system. GKG_{SEPE} is necessarily a partial instantiation; nevertheless, the subsequent chapters show that a filter model analysis of GKG_{SEPE} provides non-trivial results and many future directions.

Chapter 6

Diabetes Advanced Information System

This chapter describes how the Diabetes Advanced Information System (DAIS) is used in evaluating the filter model analysis presented in this work. In particular:

- DAIS is the *candidate system* that is supplied as input to GKG_{SEPE} , in order to measure GKG_{SEPE} 's response to a system requiring certification.
- As discussed in section 5.3, GKG was instantiated for the SEPE domain to form GKG_{SEPE} . DAIS is a system in this domain.

Section 6.1 expands on the SEPE domain and introduces DAIS. Section 6.2 expands on the role of DAIS in this work. Section 6.3 describes how the safety case representing DAIS was developed.

6.1 System Overview

DAIS is a software system inspired by Lin's work on environmental hazard analysis for Type I diabetics [1]. Recall from Chapter 1 that DAIS is an example of a new type of system, which this work calls a **safety-enhanced patient environment** (SEPE). SEPEs are information systems that store, analyze, and present information about a patient's condition. DAIS is designed to improve the safety of the living environment of a Type I diabetic using an insulin pump and continuous glucose monitor (CGM). Instead of exclusively focusing on the patient, the insulin pump, or the interaction between the patient and pump, DAIS considers the environment as a whole, holding the patient and insulin pump as given factors. Figure 6.1 illustrates this view.

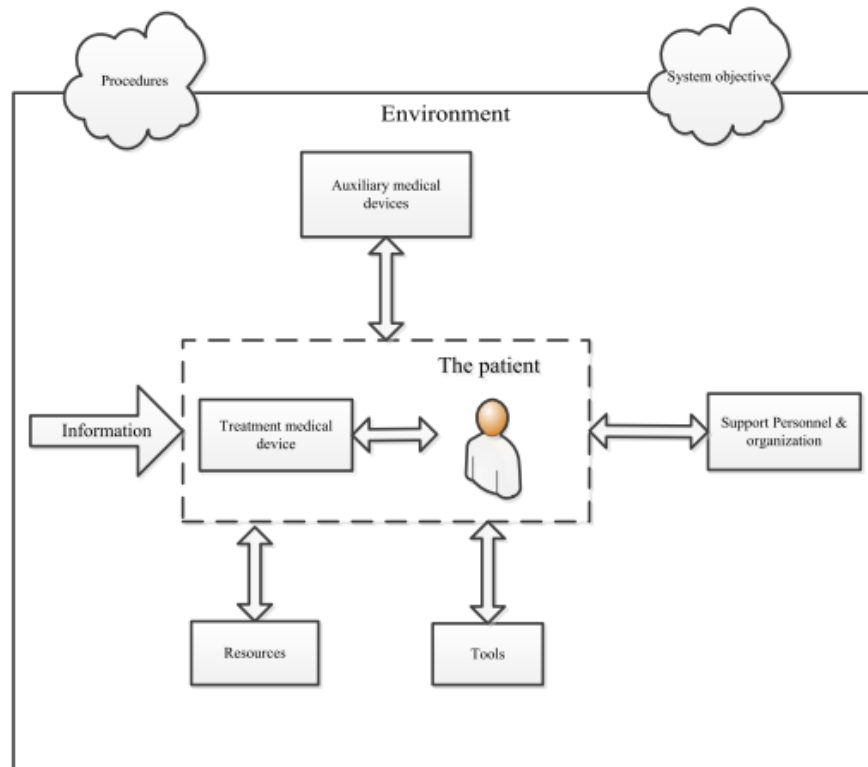


Figure 6.1: The DAIS application model (from Lin [1]).

6.1.1 DAIS Information Collection

DAIS collects, organizes, and analyzes information of various natures. The sources of this information are:

1. User input.
2. External devices.
3. Environmental sensors.
4. Trusted external databases.

Each of the following sections describes the type(s) of information DAIS records from each source.

User Input

The patient can provide DAIS with various forms of user input. Diabetics must monitor various types of information that affect blood glucose levels. Typically, patients monitor this data manually; DAIS provides the capability to electronically record supplies, food intake, medication, exercise, sickness, stress, infusion sites, and emergency contact information. All this information is available to the patient, allowing the patient to make informed decisions.

Supplies. The patient can organize and catalog *supplies* (e.g. syringes, insulin, food) using DAIS, such that DAIS can provide and maintain a comprehensive view of the living environment.

Food intake. The patient can track *food intake* using DAIS facilities. Typically, the patient requires a bolus of insulin after a meal. The amount of insulin in this bolus is determined by the nutritional characteristics of the foods in the meal, especially carbohydrate content. DAIS records the nutritional information of default meals; the patient enters in the type and quantity of foods eaten in the meal and DAIS calculates the correct bolus for that specific meal, eliminating the usual guesswork. DAIS allows patients to set a default meal schedule and default meals.

DAIS also provides for deviations from the default schedule and meals. The patient can record both:

- **Deviations from default meals**, i.e., different meals, with corresponding nutritional information. For example, the patient may eat out at a restaurant. A restaurant meal consisting of a 225 g steak with 500 g of broccoli and 100 g of brown rice may not be a default meal. DAIS can record new meals and, if the patient desires, add the new meals to the list of default meals.
- **Deviations from the default schedule**, i.e., eating a meal at an unscheduled time. For example, the patient may have a default meal – lunch – set for 12:00 p.m., but may not eat that meal until 2:00 p.m. on one particular day. DAIS can record this deviation, and, if the patient desires, change the default schedule to set lunch to 2:00 p.m.

Medication. The patient can record all current *medication*. Various medications can be prescribed for treating diabetes, as well as other diseases. These medications may interact with each other. DAIS can combine this information with information about drug interactions, e.g., from a trusted external database (see below), informing the patient of possible drug interactions as soon as possible.

Exercise. Physicians often recommend that diabetes patients embark on an *exercise* program. Exercise can affect blood glucose levels and insulin sensitivity [66]; DAIS can record the type and duration of exercise.

Illness and stress. *Illnesses* can also affect blood glucose levels [67], as can *stress* [68]. DAIS can record patient illness and stress levels.

Infusion sites. The patient uses an insulin pump to deliver infusions of insulin to the subcutaneous tissue. Insulin pumps deliver insulin through an *infusion site*, i.e., the location of the pump's connection to the skin of the patient. The patient can change infusion sites for various reasons, including periodic change to prevent scarring [69]. DAIS can record both periodic and on-demand change of infusion site.

Emergency contact information. DAIS manages the patient's emergency response mechanisms. For example, the patient designates several people as *emergency contacts*. If DAIS, through monitoring of the

CGM, senses that the patient has entered hypoglycemia and is unresponsive, DAIS can place telephone calls to people on the lists, notifying them of the emergency.

External Devices

DAIS assumes that the patient uses *external devices*; the minimum assumption is that the patient uses a continuous glucose monitor (CGM) and an insulin pump. The patient can upload the data from these devices into DAIS for DAIS to record and analyze.

Insulin pump. The patient uses an *insulin pump* to deliver boluses of insulin when necessary. The bolus amounts and times are stored in the memory of the device, which the patient can transfer to a computer. DAIS can process and record the pump data.

Continuous glucose monitor. The patient uses a continuous glucose monitor (CGM) to keep track of blood glucose levels; based on blood glucose levels, the patient may need to deliver a bolus of insulin or eat some high-glycemic-index food (food that raises blood sugar). The CGM memory records the glucose readings, which the patient can transfer to a computer. DAIS can process and record the CGM data.

Other devices. DAIS does not currently interface with other devices, but type 1 diabetics use a variety of other devices, including infusion sets, test strips, and symlin pens. With minimal effort, DAIS could accommodate data from devices such as these as well.

Sensors and Databases

Lin's work [1] mentions the possibility of DAIS interacting with embedded environmental sensors and trusted external databases. This section describes:

- How DAIS would interact with external sensors and databases.
- Examples of such sensors and databases.

Environmental sensors. Lin mentions several types of environmental sensors possible: emergency push buttons, motion sensors, and cameras. DAIS does not currently interface with any such sensors, but designs for such interfaces are in place.

Trusted external databases. Interoperability with trusted external databases is built into DAIS; however, these databases are not yet implemented. Possible databases from which DAIS could draw data are:

- **FDA database with alerts relevant to diabetes.** The FDA maintains a "Recalls, Market Withdrawals, & Safety Alerts" database [70]. DAIS could interface with a relevant subset of the extant FDA database – one that selects alerts pertaining to diabetes type 1.

- **Database with drug side-effects and interactions.** Cerner Multum, Inc. provides a “Drugs Interactions Checker” [71]; DAIS could interface with this database and other similar databases to calculate drug interactions and side effects.
- **Nutritional information database.** SELFNutritionData [72] is one of many online, searchable databases of nutrition data. Interfacing with this database would provide DAIS with the capability to auto-calculate the nutritional information of default meals, as opposed to manual entry of the nutritional information by the patient.
- **Exercise information database.** CyberSoft offers software that can track nutrition and fitness; part of this software is a database of calorie expenditure per type of exercise [73]. DAIS could interface with this database or similar databases to facilitate patient entry of exercise into DAIS.
- **FDA Manufacturer and User Facility Device Experience database.** The FDA Manufacturer and User Facility Device Experience (MAUDE) database contains failure information for medical devices. DAIS could interface with MAUDE to derive and present contra-indicated uses of the patient’s devices.

6.1.2 DAIS Information Analysis

In addition to information *collection*, DAIS provides displays to facilitate patient *analysis* of data. The patient can customize charts to display by choosing various characteristics, including:

- Type of chart, e.g., line chart.
- Data to plot, e.g., CGM glucose readings.
- Time constraints, e.g., “from May 5, 2013 to May 12, 2013.”

Figure 6.2 shows an example of a line chart with data series from CGM readings and insulin pump boluses, and with a time constraint from April 1, 2012 to April 8, 2012. Displays like the ones in Figure 6.2 are useful for the patient to discern trends in physiological response to various disturbances, whether from new meals, new medicines, illness, exercise, or other sources.

6.2 Role in Evaluation

This work uses DAIS for two purposes in filter model evaluation. Recall from Chapter 4 that:

- GKG must be initialized with *domain-specific* information, in order to be effective for a particular candidate system domain. Section 6.2.1 explains how the SEPE domain fulfills this role. As the only

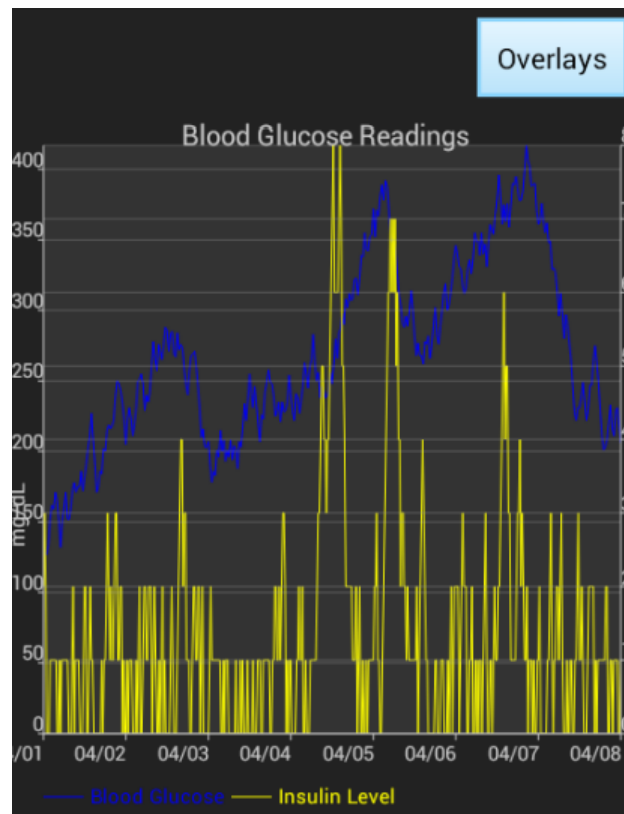


Figure 6.2: DAIS display, showing CGM readings and insulin boluses on the same chart.

extant system in the SEPE domain, DAIS serves as the system from which generalizations about the SEPE domain are drawn.

- GKG is a certification mechanism based on *safety case* examination. Thus, the evaluation must employ a full complement of safety cases, derived from *one* candidate system. This candidate system must be:
 1. Safety-critical.
 2. A member of a class of systems that resides primarily in the regulatory purview of one certifying agency.

Section 6.2.2 explains how DAIS fulfills the role of candidate system.

6.2.1 GKG Candidate System Domain - SEPE

SEPEs are designed to aid patients with diseases requiring extensive home care, i.e., patients with such diseases must “take care of themselves” to a large degree. Typically, such diseases are chronic; diabetes is one such disease. Often, a mistake in self-care can lead to serious injury or death. Such a system, because of

the nature of the information it manipulates, is certainly safety critical and would fall under the regulatory purview of the FDA. Thus, SEPE serves as the candidate system domain for the partial GKG instantiation used in the evaluation, GKG_{SEPE} .

6.2.2 Evaluation Candidate System - DAIS

DAIS serves as the candidate system in the evaluation. The safety case for DAIS serves as the baseline safety case; this safety case is used to create several evaluation safety cases. However, DAIS and its safety case have the following limitations:

- Currently, DAIS is an experimental prototype. Such a system is not representative of an industrial safety-critical system. Instead of using the system as-is, this work hypothesizes a high-assurance version of DAIS: a version that is an ideal candidate for certification.
- DAIS, as an experimental prototype, was not constructed in accordance with a typical software engineering process. As such, DAIS does not have a corresponding complete set of development artifacts to include in a safety case, nor was its safety case constructed with the “early and often” [13] method recommended by the safety engineering community. Instead of creating a safety case representative of DAIS as-is, this work assumes the existence of various artifacts and, where appropriate, elaborates the contents of these artifacts.

Section 6.3 expands on how the baseline safety case was constructed. Recall from Chapter 4 that the evaluation safety cases are created by seeding $faults_{cand}$ into the baseline safety case. These $faults_{cand}$ correspond to $faults_{cert}$ found in GKG. Chapter 8 summarizes the $faults_{cert}$ and the corresponding $faults_{cand}$.

6.3 DAIS Baseline Safety Case

Recall from Chapter 4 that this work uses a set of **evaluation safety cases** to evaluate the filter model. The DAIS safety cases are this set. The baseline safety case, used to construct the evaluation safety cases, reflects a *high-assurance version* of DAIS. This section expands on how the baseline safety case was constructed.

The main strategy for the safety case was:

Argue that all hazards defined in the hazard log are mitigated.

Section 6.3.1 presents the hazard analysis used as the basis of the safety case.

SEPE Requirement	DAIS Requirement
The probability that the system, in and of itself, causes mistreatment is acceptably low.	DAIS is a software system that handles information and, as such, can only cause serious harm through providing misinformation (as discussed previously) —whether through internal data mis-handling or security breaches.
The probability that the system’s interoperability with other devices causes mistreatment is acceptably low.	DAIS is designed to work with other device domains, including insulin pumps and CGMs. In addition, DAIS runs on a number of interconnected pieces of computer hardware through a network. Any communication DAIS conducts with external devices must not affect the correct operation of the devices.
The probability that the system allows confidential information to be accessed by unauthorized parties is acceptably low.	DAIS includes connections to remote databases (e.g., the insulin pump’s record or the FDA alerts database) over the Internet. If these connections are compromised, attackers could access confidential information (e.g., the patient’s other prescriptions).

Table 6.1: FDA SEPE requirements and their instantiation in DAIS.

6.3.1 DAIS Hazard Analysis

Lin’s work identified two major hazardous states possible through the environment: severe hypoglycemia and severe hyperglycemia of the patient. Chapter 7 of Lin’s work divides the information system into *Data Analysis*, *Data Archive*, and *Interface* modules.

6.3.2 Requirements

DAIS would be under the regulatory purview of the FDA, in the SEPE domain. SEPE is not yet recognized as an official FDA domain, due to its novelty. Thus, this work *hypothesizes* FDA requirements for all SEPE systems, shown on the left side of Table 6.1. DAIS requirements are derived from these hypothesized SEPE requirements, and are shown on the right side of Table 6.1.

The two hazards that can occur in the patient’s *environment* are:

1. Severe hypoglycemia.
2. Severe hyperglycemia.

These two *environmental* hazards can be caused by factors external to DAIS, but this work will focus on how DAIS can bring about these hazards. All of the DAIS modules can generate events that contribute to the environmental hazards. Usually, these events have similar geneses:

The path from $fault_{cand}$ to $hazard_{cand}$ for a SEPE will include providing misinformation to the patient in some form.

Thus, the main hazard_{cand} for DAIS is **misinformation**. Misinformation can come in two forms: **omission of necessary information** or **supply of incorrect information**.

Omission of necessary information. Consider a situation in which a patient adds a new prescription into DAIS. DAIS should monitor the interactions of the drugs that the patient is prescribed; if DAIS fails to warn the patient of a lethal interaction involving the new drug and an existing drug, the patient could be misled into thinking DAIS is protecting him from hurting himself by taking either of the interacting drugs. In this case, DAIS is providing misinformation through **omitting the necessary information** for a decision that the patient must make.

Incorrect information. Consider the situation in which a patient enters a drug that has no adverse interactions with existing prescriptions, but DAIS decision support notifies the patient that the drug *does* have adverse interactions. In this case, DAIS **supplies incorrect information** directly to the patient.

Because DAIS is designed to work with all insulin pumps and CGMs, rather than just one brand, interoperability is also a major concern. Interfaces and data formats could be different from manufacturer to manufacturer. A complete implementation of DAIS should have the following two properties:

- DAIS should be able to communicate effectively with any insulin pump.
- DAIS should not interfere with the correct operation of the insulin pump or CGM.

Therefore, a complete safety case should *argue* that the implementation fulfills these requirements. The corresponding certification of the DAIS implementation using this safety case should determine whether the argument is *sufficient*.

In addition, confidentiality is a major concern with any networked system, and a complete implementation of DAIS should not allow an unacceptable amount of confidential data to be accessed by attackers. A security breach of DAIS or the insulin pump [74] could certainly cause mistreatment. However, if attackers are merely *reading*, not *writing*, confidential data, then confidentiality concerns may be better suited to a *security* case, i.e., an assurance case for security [75], rather than a *safety* case. Confidential information could be used to blackmail a patient, but such activities and their possible physical consequences are well beyond the scope of treatment. While effectively independent from safety concerns, confidentiality concerns are always present, and could be addressed via a separate security certification on a security case for the system. GKG is designed to be property-agnostic, i.e., the certification aim could be safety, security, availability, or any other property. Thus, GKG could be used to certify information security as a separate property, or together, using a combined safety/security case. Security cases and their combination with other properties, e.g., safety, is a current area of research [75].

A complete safety case would address all of the SEPE requirements, as all the requirements represent domain hazards_{cand}. However, evaluating security and interoperability is beyond the scope of this work. Thus, this case study focuses only on the first SEPE requirement in Table 6.1, **lack of mistreatment**. Correspondingly, the hazard_{cand} analysis for the DAIS safety case focuses on how **omission of necessary information** and **incorrect information** – the two types of **misinformation** – contribute to patient mistreatment.

6.3.3 Hazard Analysis Techniques

Creating a plausible baseline evaluation safety case requires a hazard_{cand} analysis – any ideal safety case would at least consider all known hazards in the domain. The DAIS hazard analysis employs HazOp and FTA. HazOp was chosen to *identify* hazards, and FTA was chosen to *elaborate* these hazards into events.

The hazard analysis techniques used raise the following open question:

Open Question 6.1. *What hazard analysis techniques work best for SEPEs?*

Appendices D and E present the full DAIS HazOp and FTA, respectively. The resulting events are summarized here:

Event 1 The Data Archive module fails to supply the relevant requested information at an acceptable rate.

Event 2 The link between Data Analysis and Interface Modules fails at an unacceptable rate.

Event 3 The Data Analysis Module fails to procure relevant information at an unacceptable rate.

Event 4 The Interface Module display fails at an unacceptable rate.

Event 5 The Data Analysis module fails to send a relevant remote alarm signal at an acceptable rate.

Event 6 The link between patient device and Data Analysis Module fails at an unacceptable rate.

Event 7 The Data Analysis Module verification fails at an unacceptable rate.

Event 8 The patient fails to enter data at an acceptable rate.

Event 9 The device fails to present correct data at an acceptable rate.

Event 10 The database has incorrect information at an unacceptable rate.

Event 11 The Data Analysis Module calculates incorrectly at an unacceptable rate.

Event 12 The Interface Module fails at an unacceptable rate.

6.3.4 Safety Case

The safety case is designed to show hazard mitigation with direct evidence. The safety case uses the construction of a **diverse two-legged argument** for each event, as discussed in Chapter 5; each event sub-argument has a testing leg and proof leg [64, 60]. This analysis *hypothesizes* the existence of a formal

specification and a test plan; creating a full-fledged formal specification and test plan for DAIS is beyond the scope of this work.

Implicit in the two-legged construction is the following assumption:

Case Study Assumption 6.1. *The two-legged argument construction shows a high degree of reliability by virtue of extraordinary amounts of testing, in addition to a formal verification.*

This assumption is generally accepted, but there are certain edge cases that may be problematic; these are identified by Littlewood and Wright [64]. The caveat regarding argument leg independence also applies. However, this analysis does not use the safety case to derive a probability of failure, so independence is not needed for any probability calculations. This analysis makes only a qualitative claim about the reliability afforded by the diverse two-legged construction.

Figure 6.3 shows an example of one of the argument legs. For the goal node, `G_LackOfInformation.1`, the text for **Event 1**, “The Data Archive module *fails* to supply the relevant requested information *at an acceptable rate*,” is reversed to its positive, such that the goal node text reads, “The rate of the Interface Module displaying incorrect data is *acceptably low*.” The solution nodes, `Sn.1` and `Sn.2`, reference two documents that support the goal:

- Sn.1** A formal proof report, “FormalProof.pdf.” This document shows the results of a formal verification with respect to a formal specification, “FormalSpec.pdf.” The formal specification is referenced in the context node, `C_LackofInformation.1.2`.
- Sn.2** A testing report, “TestReport.pdf.” This document shows the results of a test battery. The test cases for this test battery, “TestCases.pdf” are referenced in the context node, `C_LackofInformation.1.1`.

The other eleven events are structured similarly. Appendix F shows the full DAIS safety argument in GSN.

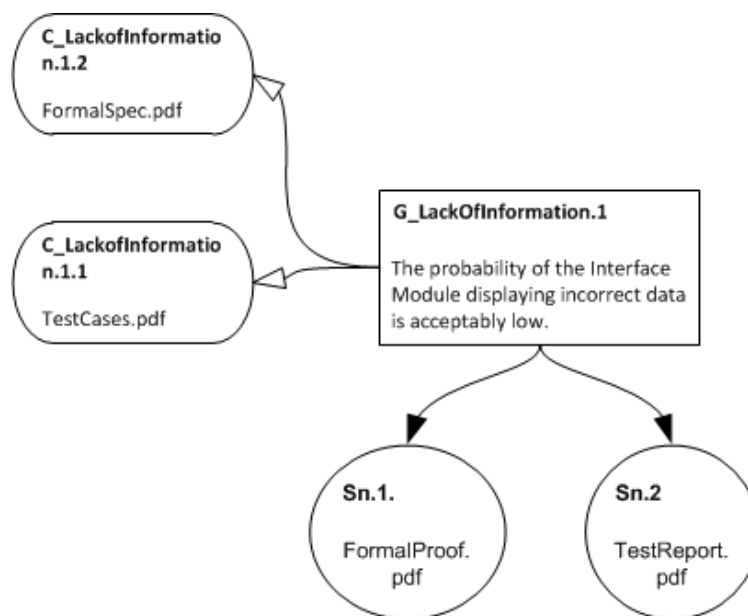


Figure 6.3: A portion of the baseline safety case. This GSN structure is a diverse two-legged argument, designed to show partial mitigation for Event 1.

Chapter 7

GKG Hazard Analysis

7.1 Introduction

Recall from Chapter 4 that the case study evaluation is structured in two phases. This chapter presents the results of Phase 1: a hazard analysis on GKG_{SEPE} . Chapter 6 introduced the idea of safety-enhanced patient environments (SEPEs) as a *candidate system domain*, with the Diabetes Advanced Information System (DAIS) as a system in the SEPE domain. Chapter 5 introduced GKG_{SEPE} as the Graydon-Knight-Green *certification mechanism* instantiated for certification of SEPE candidate systems. Figure 7.1, reproduced here from Chapter 4, shows the roles of DAIS and GKG_{SEPE} in the evaluation, along with how phase 2 uses the output of phase 1.

The goal of phase 1 is to analyze GKG_{SEPE} for $hazards_{cert}$ using common hazard analysis techniques. Chapter 8 presents the results of phase 2: safety engineering of GKG_{SEPE} , based on the $faults_{cert}$ found in phase 1.

This chapter presents the results of three hazard analysis techniques:

- Hazard and operability study (HazOp). Section 7.2 presents the HazOp results.
- Fault tree analysis (FTA). Section 7.3 presents the FTA results.
- Failure modes, effects and criticality analysis (FMECA). Section 7.4 presents the FMECA results.

Each technique provided two categories of results, **feasibility** and **yield**. Each of the hazard analysis techniques as applied to GKG_{SEPE} provided some results in both categories. The discussions of results focus on these categories in the following ways:

Feasibility. Given the novelty of the application domain (certification mechanisms), this work measured the feasibility of application of each of the three hazard analysis techniques using the following metrics:

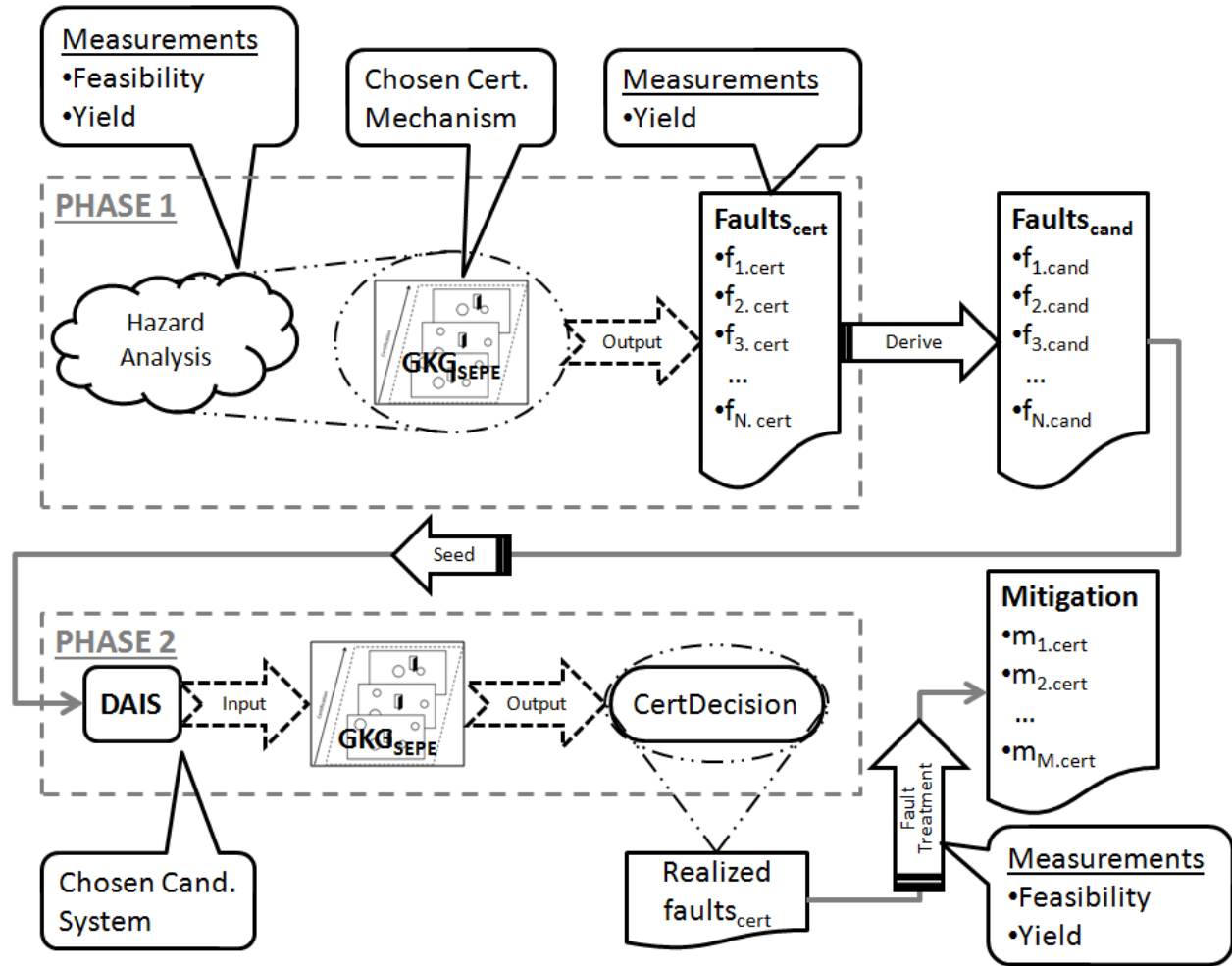


Figure 7.1: Summary of the evaluation used in this work (from Chapter 4).

- Time taken to perform the analysis.
- Whether or not the analysis yielded any non-trivial results.

Yield. Each technique helps to analyze hazards in a different way. This work categorizes the results of each technique into the type of $\text{fault}_{\text{cert}}$ that would cause a corresponding $\text{hazard}_{\text{cert}}$. In addition, several metrics are developed for each technique; each results subsection describes these metrics in greater detail.

Most of these metrics suffer from the fact that **this analysis is the first time hazard analysis techniques have been applied to certification mechanisms**. Thus, there are many open questions relating to the **acceptability criteria** for these metrics. Each results subsection details these questions in relation to the relevant metrics.

Ideally, feasibility results and yield results would be independent. However, the two properties are not

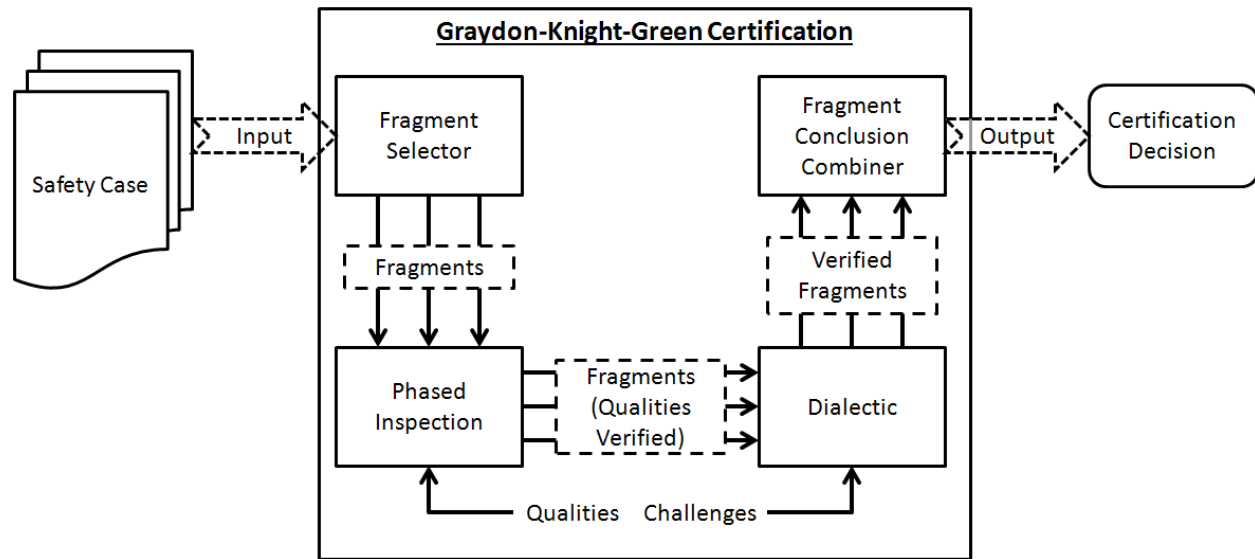


Figure 7.2: The functional block diagram used for the safety analyses.

easily separated and controlled for without extraordinary resources. Moreover, the limited analysis presented in this work prevents large-scale declarations of feasibility or non-feasibility. This analysis concerns only GKG_{SEPE} :

- **Feasibility.** If a technique, T , yields **non-trivial results**, then T is feasible, but *only for* GKG_{SEPE} . No claims can be made about whether T is feasible for other certification mechanisms.
- **Non-feasibility.** If T yields **no non-trivial results**, then T is non-feasible, but only for GKG_{SEPE} . No claims can be made about whether T is non-feasible for other certification mechanisms.

Thus, each section includes a discussion of feasibility and yield together.

Throughout the following sections, I note various **open questions** about limitations of the hazard analysis techniques used (HazOp, FTA, and FMECA), the results metrics, and the methods used to collect results. Chapter 10 discusses these and other open questions as possibilities for future work.

To facilitate the safety analyses, a functional block diagram of GKG was developed, shown in Figure 7.2. The block diagram describes how information flows through the filter.

7.2 HazOp

Recall the nature of HazOp analyses:

- HazOp is an algorithmic question generation strategy that helps analysts combine *guide words* and *parameters* to form questions about various system components.
- Guide words are not system-specific; usually, there is an industry standard set of HazOp guide words.
- Parameters are system specific, and are often derived from a functional block diagram or other diagram depicting system flow.
- HazOp-generated questions are sometimes irrelevant or nonsensical.

This section describes the conduction of the GKG_{SEPE} HazOp. Below is the HazOp process followed in this work.

1. Select guide word set.
2. Construct parameter set.
3. Construct the cross-product set of guide words and parameters and generate questions from this set.
4. Classify questions as *relevant/not relevant*.
5. Answer the relevant questions and classify the answers as *representing a fault_{cert}/not representing a fault_{cert}*.
6. Assign fault designations to each fault_{cert}.

Sections 7.2.1 through 7.2.5 elaborate on the conduction of each step in this process, and section 7.2.6 presents the results. Appendix A contains the full GKG_{SEPE} HazOp analysis.

7.2.1 Guide Word Selection

The GKG HazOp uses the standard guide words, described in Table 3.2. The Chemical Industries Association of the UK issued guidance for HazOp in 1977 [76] that included the standard guide words. There are other guides for other domains, including software [77, 78], but which set of guide words to apply for any given filter model analysis is an open question. A special case of this question is whether *new* guide words should be developed for filter model analyses.

Open Question 7.1. *In a filter analysis, what are the criteria for selection of HazOp guide words?*

Open Question 7.2. *Should analysts develop new guide words specifically for filter analysis?*

For simplicity, this HazOp analysis uses the standard guide words as a starting point.

7.2.2 Parameter Construction

Parameters for candidate system HazOp analysis are derived from a flow model of the system. One advantage of the filter model is that the model views the certification mechanism as a flow of *information*. This is a fundamental property of certifications:

*Analysts can derive an information flow model
for any certification mechanism.*

The correctness and/or complexity of the flow model may vary greatly; such variance may affect the parameters selected. How the parameters change is an open question.

Open Question 7.3. *In a filter model analysis, how much do the correctness and/or complexity of the flow model of the certification mechanism affect which HazOp parameters analysts generate?*

The best way to generate parameters for filter model HazOp is another open question.

Open Question 7.4. *In a filter analysis, what criteria should analysts use to (a) generate HazOp parameters and (b) determine if the generated parameters are acceptable?*

GKG HazOp parameters are derived from the flow control blocks and inputs in the functional block diagram from Figure 7.2:

1. Safety case.
2. Fragment selector.
3. Phased inspection.
4. Quality.
5. Dialectic.
6. Challenge.
7. Conclusion combiner.

These parameters may not be *optimal* parameters for analysis, because they might be generated based on a faulty assumption: the flow control blocks and inputs are a sufficient source for parameter generation. In addition, these parameters may not be a complete set of parameters. This analysis arrived at non-trivial results using this selection of parameters, but further investigation of the effect of parameter selection on results is needed.

7.2.3 Cross Product and Question Generation

Questions for candidate system HazOp analysis are generated by following these two steps:

1. **Cross-product construction.** The input to question generation is the cross product of (a) the set of guide words and (b) the set of questions. Formally:

- Let G be the set of selected guide words:

$$G = \{NO\ OR\ NOT, MORE, LESS, AS\ WELL\ AS, \\ PART\ OF, REVERSE, OTHER\ THAN, EARLY, \\ LATE, BEFORE, AFTER\} \quad (7.1)$$

- Let P be the set of constructed parameters:

$$P = \{Safety\ Case, Fragment\ Selector, Phased\ Inspection, \\ Quality, Dialectic, Challenge, Conclusion\ Combiner\} \quad (7.2)$$

- Let C be the input set to question generation:

$$C = G \times P \quad (7.3)$$

$$C = \{\{NO\ OR\ NOT, Safety\ Case\}, \\ \dots \\ \{AFTER, Conclusion\ Combiner\}\} \quad (7.4)$$

The process of constructing C is entirely mechanical.

2. **Question generation.** After generating C , analysts use the pairs in C to generate a set of questions about the system, Q :

$$Q = questionGeneration(C), \quad (7.5)$$

where *questionGeneration* is the process through which analysts generate a question, $q_i \in Q$, from a pair, $c_i \in C$. Question generation is largely guided by:

- Experience with the subject system domain.
- Experience with applying HazOp.
- Intuition.

Thus, the content of the questions in Q depends on the experience and intuition of the analyst applying HazOp.

In this work, question generation depended on the author's intuition and experience. Given the same guide words and parameters, analysts with different backgrounds and experience may have generated different questions; the effect of expertise/background variance on the content of the questions in Q is not known.

Open Question 7.5. *To what extent does analyst expertise in the particular subject domain of certification affect Q in a filter model analysis, i.e., on a certification mechanism?*

Open Question 7.6. *To what extent does analyst expertise in general application of HazOp affect Q in a filter model analysis?*

7.2.4 Relevancy Classification

Because of the cross-product construction, not all candidate system HazOp questions are relevant. Analysts must classify each question from Q as *relevant* or *irrelevant*. The relevant questions are collected into a new set, R :

$$R \subseteq Q \quad (7.6)$$

$$R = \text{relevancyClassification}(Q) \quad (7.7)$$

R contains the set of questions that analysts answer. Relevancy classification is dependent on the analysts' backgrounds and experience.

As an example of relevancy classification, consider the pair, c_1 :

$$c_1 = \{NO \ OR \ NOT, Safety \ Case\} \quad (7.8)$$

The parameter, *NO OR NOT*, conveys "complete negation of the design intent" (cf. Table 3.2). The resultant pair, c_1 , generated the following question, q_1 :

``What if the submitted safety case does not address risk?''

This question was classified as *irrelevant* by the author's intuition and expertise: safety cases are *meant* to address overall system risk. However, analysts with different backgrounds and experience could interpret q_1 differently, possibly resulting in q_1 being classified as *relevant*. The effect of background/experience on question classification is not known.

Open Question 7.7. *To what extent does analyst expertise affect R in a filter model analysis?*

7.2.5 Answers and Fault Classification

After completing relevancy classification, analysts answer the questions in R and decide whether the answers imply faults. Similarly to question generation and relevancy classification, question answering and fault classification are guided by analyst experience and intuition. The effect of background/experience on question answering and fault classification is not known.

Open Question 7.8. *To what extent does analyst expertise affect question answering and $\text{fault}_{\text{cert}}$ classification in a filter model analysis?*

7.2.6 Results

This section presents the results of the HazOp analysis on GKG_{SEPE} . The results are presented in the following order:

1. Measurements of HazOp metrics. Table 7.1 presents:
 - The metrics this analysis uses to assess the feasibility and yield of applying HazOp.
 - The values of these metrics.
 - The category of each metric.
2. Discussion of HazOp metrics and measurements, including a qualitative discussion of acceptability.
3. Discussion of the limitations of these results, including open questions raised by these results.
4. Discovered $\text{faults}_{\text{cert}}$. Tables 7.2 and 7.3 present a summary of the $\text{faults}_{\text{cert}}$ in GKG_{SEPE} discovered by the HazOp analysis, including open questions about the effects of GKG fragment selection and conclusion combination on the conclusions drawn from the hazard analysis.

Feasibility and Yield Assessment

This analysis uses several metrics to assess feasibility and yield. Table 7.1 shows these metrics, the value at which each metric was measured, and what property each metric is designed to assess. We cannot meaningfully compare the measurements in Table 7.1 to the same measurements for HazOp analyses of candidate systems. However, the HazOp did achieve non-trivial results, generating 88 questions and discovering 9 potential $\text{faults}_{\text{cert}}$ based on those questions in 1.5 hours. Both feasibility metrics – **time** and **generation of non-trivial results** – are fulfilled.

In terms of yield, 9 potential $\text{faults}_{\text{cert}}$ out of 88 generated questions results in 10.227% of generated questions identifying $\text{faults}_{\text{cert}}$. Again, we cannot meaningfully compare this statistic to others like it, but

Metric	Value	Category
Hours to complete HazOp	1.5	Feasibility
Total number of questions and answers	88	Yield
Questions generated and answered per hour	58.667	Feasibility/Yield
Number of irrelevant questions	59	Yield
Number of relevant questions	29	Yield
Number of questions with answers that imply faults	9	Yield
Number of questions with answers that do not imply faults	20	Yield
Potential faults _{cert} discovered per hour	13.333	Feasibility/Yield
Percentage of faults _{cert} in relation to total questions	10.227%	Feasibility/Yield

Table 7.1: Measurements for the HazOp analysis on the GKG_{SEPE} certification mechanism.

we can state that HazOp has a **non-zero yield** for discovering faults_{cert}, which is important both in terms of feasibility and yield.

Limitations

These results imply a number of open questions relating to the application of the HazOp technique to GKG in addition to Open Questions 7.1 through 7.8. In particular, the metrics and their acceptability criteria are not well-understood.

Open Question 7.9. *In a filter model analysis, what metrics should analysts measure to determine the feasibility and yield of HazOp analysis?*

Open Question 7.10. *In a filter model analysis, what are the acceptability criteria for the metrics chosen to determine feasibility and yield of HazOp analysis?*

Discovered Faults_{cert}

Out of the 29 relevant questions produced in the HazOp analysis, 9 had answers that pointed to faults_{cert} in the GKG mechanism. Tables 7.2 and 7.3 presents the guide words and parameters that produced each of the 9 questions, along with the conditions under which the fault_{cert} can be exercised. The table also assigns a unique **fault designation** to each result. Appendix A contains the whole HazOp analysis.

Fault Designation	Parameter	Guide Word	Question	Fault Exercised When:
$F_{cert-frag.oth}$	Fragment	Other Than	What if the current fragment is corrupted?	The fragment contains a fault and the corruption masks the fault.
$F_{cert-qual.oth}$	Quality	Other Than	What if the current quality is corrupted?	The corruption changes the quality into a quality not exhibited by sound arguments.
$F_{cert-chal.oth}$	Challenge	Other Than	What if the current challenge is corrupted?	The corruption changes the challenge such that the challenge no longer accurately represents a domain hazard.
$F_{cert-insp.oth}$	Phased inspection	Other Than	What if the phased inspection is not conducted correctly?	The phased inspection is done incorrectly, e.g., by inexperienced inspectors, in the improper order, etc.
$F_{cert-dial.le}$	Dialectic	Less	What if the dialectic issues fewer challenges than necessary?	The dialectic fails to issue critical challenges.
$F_{cert-dial.rev}$	Dialectic	Reverse	What if the dialectic does not challenge the fragment?	Certifier and developer agree that the fragment has been sufficiently challenged.
$F_{cert-dial.oth}$	Dialectic	Other Than	What if the dialectic is corrupted?	The dialectic fails to issue challenges that, if issued, would uncover faults _{cand} .

Table 7.2: Results of HazOp analysis on the GKG certification mechanism.

As per Case Study Assumptions 4.3 and 4.4, I deemed questions about the fragment selection and conclusion combination irrelevant because analyzing either mechanism is out of scope of this analysis. This analysis assumes that the effects of fragment selection and conclusion combination are null for simplicity. However, these effects are unknown.

Open Question 7.11. *What are the effects of GKG fragment selection on the meaning of the fragments?*

Open Question 7.12. *What are the effects of GKG conclusion combination on the meaning of the conclusions?*

These questions are analogous to questions about compositional models for dependability assessment of candidate systems:

Fault Designation	Parameter	Guide Word	Question	Fault Exercised When:
$F_{cert-sc.oth}$	Safety Case	Other Than	What if the submitted safety case is corrupted?	The safety case changes in such a way that it masks faults _{cand} , allowing the candidate system to be certified.
$F_{cert-qual.po}$	Quality	Part Of	What if the current quality is not stringent enough?	A degenerate case of $cert.qual.oth$: equivalent to corruption of the quality from a more stringent version to a less stringent version.

Table 7.3: Faults_{cert} discovered in the GKG certification mechanism using HazOp analysis.

Fragment selection as decomposition. Decomposition of a candidate system to its component subsystems may be difficult, due to high degrees of coupling between subsystems. Forcing decomposition of two tightly coupled subsystems may change the behavior of each subsystem.

Conclusion combination as composition. Recall from Chapter 2 that there are no generally accepted compositional models for dependability assessment of candidate systems. Thus, composition of the dependability assessments of subsystems into an overall system-wide dependability assessment is difficult.

7.3 Fault Tree Analysis

Recall the nature of fault tree analyses from Chapter 3:

- Fault tree analysis (FTA) is a top-down method of hazard analysis, starting with a *hazard* of interest.
- The goal of FTA is to determine all the *events* that could lead to the hazard. The two types of events are *compound* and *basic* events. Basic events are events that can be treated as indivisible actions *in the system context*; compound events are logical combinations of basic events.
- Analysts break down the hazard into events, *refining* compound events until basic events are reached.
- The determination and refinement of events is *informal*. Effective FTA requires *insight* about the system in question and *experience* in the system domain.

This section describes the conduction of the GKG_{SEPE} FTA. Below is the FTA process followed in this work.

1. Start with the hazard of interest and hypothesize events that could cause GKG_{SEPE} to reach this hazard.
2. Hypothesize causal events for the events recorded in # 1. Iterate until events can be classified as basic.

3. Adjudge the faults_{cert} that cause the basic events.
4. Assign fault designations to each fault_{cert}

Sections 7.3.1, 7.3.2, and 7.3.3 elaborate on the conduction of each step in this process and section 7.3.4 presents the results. Appendix B contains the whole GKG_{SEPE} FTA.

7.3.1 Initial Hazard and Event Hypothesizing

The initial hazard in a fault tree is called the **top-level hazard**. Recall from Chapter 3 that any certification mechanism, *CM*, presented with a candidate system, *CS*, can have two **unwarranted outcomes**:

- **Unwarranted rejection.** The rejection of *CS* is **unwarranted** if *CS* is acceptably safe, but *CM* decides to reject *CS*.
- **Unwarranted acceptance.** The acceptance of *CS* is **unwarranted** if *CS* is unacceptably unsafe, but *CM* decides to accept *CS*.

Both of these outcomes constitute *hazards* for *CM*. This work focuses only on unwarranted acceptance. Thus, the top-level hazard for the GKG_{SEPE} FTA was:

```
``GKG allows certification of a product with unacceptable residual
risk.``
```

After determining the top-level hazard, causal events for the top-level hazard are hypothesized. The events are combined with Boolean logic operators; the highest-frequency operators are *AND* and *OR*, but other operators can be used.

This work based the first-iteration events on the components derived from the flow model in Figure 7.2. Each first-iteration event is combined together with the rest using an *OR* operator. Below is an example of one event derived from the *quality* component of the flow model.

```
``Quality omitted from qualities list.``
```

This analysis uses the flow model components as a simple starting point. This starting point may not be *optimal*, because the hypothesized events might be (a) incomplete or (b) incorrect. In general, hypothesizing events in a fault tree is partially based on the experience and intuition of the analyst. Furthermore, determining how the events combine is also based on experience and intuition.

Open Question 7.13. *What criteria should analysts use to hypothesize events and combinations thereof when conducting FTA in a filter model analysis?*

7.3.2 Event Hypothesizing and Iteration

Subsequent hypothesizing of events is done according to the ability of the analyst. The analyst continues to iteratively refine the events in the fault tree until he decides that the events are **basic events**. Basic events are events that analysts can treat as indivisible actions.

An event is basic if the analyst *does not need* to refine a hypothesized event any further. An event, E , can *always* be refined, but depending on analyst's context, E might not *need* to be refined.

As an example, consider the Amazon Web Services Java library. A hypothetical fault tree for this library could include an event based on the class, *DBInstance*:

```
``DBInstance fails.``
```

An analyst at a company simply *using* the library would not need to refine this event any further, as *DBInstance* would be used as a “black box.” However, analysts at Amazon would be interested in how exactly *DBInstance* fails, necessitating further refinement of the event.

Deciding whether to classify an event as basic requires some level of analyst judgment. The FTA in this work is the product of the author's judgements; other analysts could make different judgments. The effects of intuition and experience on fault tree event refinement is not known.

Open Question 7.14. *To what extent does analyst expertise affect fault tree event refinement in a filter model analysis?*

7.3.3 Fault Determination

Recall from Chapter 1 that a **fault** is the *adjudged* cause of an erroneous state, which could be a hazard. The events in a fault tree are *not* faults in themselves; *some* of the events in a fault tree happen because of faults. Deriving faults from a fault tree requires an additional analyst judgment.

Adjudging faults is difficult enough for well-established domains: despite analysts having extensive experience in a certain domain, adjudging faults is still an informal and, at times, imprecise activity. The novelty of applying FTA to certification mechanisms makes this difficulty even more apparent. This work adjudges faults by adapting the second-level events in the GKG_{SEPE} fault tree. As an example, consider the this second-level event:

```
``Quality stated incorrectly in qualities list.``
```

This sentence fragment can be interpreted as an event, i.e., “An analyst states a quality incorrectly.” After this event occurs, however, the event *creates a fault* in the GKG mechanism: the qualities list that will be used in the phased inspection now has an incorrect quality. This kind of understanding is applied to all the second-level events because beyond the second level, the events do not seem to be amenable to such understanding. However, analysts with different intuition and experience could adjudge faults differently.

Open Question 7.15. *To what extent does analyst expertise affect fault adjudging in a filter model analysis?*

7.3.4 Results

This section presents the results of the fault tree analysis on GKG_{SEPE} in the following order:

1. Measurements of FTA metrics. Table 7.4 presents:
 - The metrics this analysis uses to assess the feasibility and yield of applying FTA to GKG.
 - The values of these metrics.
 - The category (*feasibility* and/or *yield*) of each metric.
2. Discussion of FTA metrics and measurements, including a qualitative discussion of acceptability.
3. Discussion of the limitations of these results, including open questions raised.
4. Discovered faults_{cert}. Table 7.5 presents a summary of the faults_{cert} in GKG_{SEPE} discovered by the FTA.

Following is a discussion of the open questions raised by these results.

Feasibility and Yield Assessment

This analysis uses several metrics to assess feasibility and yield of applying FTA to GKG_{SEPE}. Table 7.5 shows these metrics, the value at which each metric was measured, and what property each metric is designed to assess. We cannot meaningfully compare the measurements in Table 7.4 to the same measurements for FTA of candidate systems. However, the FTA achieved non-trivial results: a fault tree with 57 total events and 7 total faults was created in 32 hours. Both feasibility metrics – **time** and **generation of non-trivial results** – are fulfilled.

In terms of yield, 7 potential faults_{cert} out of 57 total events results in 12.281 % of events being designated as ones that create faults in GKG_{SEPE}. Compared to the HazOp analysis from section 7.2, the percentage of faults_{cert} found per element (question for HazOp and event for FTA) is slightly higher. However, the FTA took significantly more time to complete, resulting in only 0.122 faults_{cert} being found per hour. Nevertheless, 5 of these 7 faults were distinct from the faults discovered using HazOp; thus, we can state that FTA has a **non-zero** yield for discovering faults_{cert}.

Limitations

In addition to the limitations stated in the Open Questions in this section (Open Questions 7.13 through 7.15), the limitations on metric acceptability criteria stated for HazOp in section 7.2.6 also apply to this FTA and its metrics. Additionally, the existence of overlapping raises the issue of feasibility of applying

Metric	Value	Category
Hours to complete FTA	32	Feasibility
Total number of events	57	Yield
Intermediate events	12	Yield
Basic events	45	Yield
Events generated per hour	1.781	Feasibility/Yield
Number of faults _{cert}	7	Yield
Percentage of faults _{cert} to events	12.281%	Yield
Faults _{cert} generated per hour	0.122	Yield
Number of faults overlapping with HazOp analysis	3	Feasibility
Percent of faults overlapping with HazOp analysis	42.857	Feasibility

Table 7.4: Measurements for the fault tree analysis on the GKG certification mechanism.

one technique *relative* to another. Whether a 42.857% overlap between two techniques is acceptable for the resources spent on the more expensive technique is unknown.

Open Question 7.16. *To what extent do overlapping results affect the feasibility of a hazard analysis technique?*

Discovered Faults_{cert}

The FTA discovered 7 faults_{cert}. Of these 7, 3 were classified as similar to faults_{cert}:

1. $F_{cert-dial.rev}$. The HazOp question, “What if the dialectic does not challenge the fragment?” has as its answer an FTA result, “The dialectic leads to unwarranted agreement.”
2. $F_{cert-chal.oth}$. The HazOp question, “What if the current challenge is corrupted,” is the statement, “Challenge in challenge list stated incorrectly,” but in question form.
3. $F_{cert-frag.oth}$. The HazOp question, “What if the current fragment is corrupted?” is the statement, “Current fragment corrupted,” but in question form.

These three faults_{cert} use the same fault designation in both analyses. Each of the other four faults_{cert} receives a fault designation. Table 7.5 presents the faults_{cert} discovered by the GKG FTA, and the events from which the faults_{cert} were derived.

Fault Designation	Event that Creates Fault:
$F_{cert-qual.est.inc}$	Quality X established incorrectly (i.e., it is established, but is not actually present).
$F_{cert-qual.omi}$	Quality omitted from qualities list.
$F_{cert-qual.oth}$	Quality stated incorrectly in qualities list.
$F_{cert-dial.rev}$ (same as GKG HazOp)	Dialectic leads to unwarranted agreement.
$F_{cert-chal.oth}$ (same as GKG HazOp)	Challenge in challenge list stated incorrectly.
$F_{cert-chal.omi}$	Challenge in challenge list omitted.
$F_{cert-frag.oth}$ (same as GKG HazOp)	Current fragment corrupted.

Table 7.5: Results of FTA on the GKG certification mechanism.

7.4 Failure Modes, Effects, and Criticality Analysis

Recall the nature of failure modes, effects, and criticality analyses from Chapter 3:

- Failure modes, effects, and criticality analysis (FMECA) is a bottom-up, inductive method of hazard analysis, starting with the *system components*.
- The goal of FMECA is to determine:
 - *Failure modes*, i.e., under what conditions components can fail.
 - *Effects*, i.e., what happens when components fail.
 - *Criticality*, i.e., which effects, and thus failure modes, are the most dangerous.
- Determining the failure modes, effects, and criticality relies to some extent on analyst intuition and experience.

This section describes the conduction of the GKG_{SEPE} FMECA. Below is the FMECA process followed in this work.

1. Start with the flow model elements from Figure 7.2, using the elements as FMECA components.
2. Hypothesize failure modes for each component.
3. Hypothesize effects for each failure.
4. Hypothesize circumstances under which effects become critical.

Sections 7.4.1 through 7.4.3 elaborate on the conduction of each step in the process and section 7.4.4 presents the results. Appendix C contains the whole GKG_{SEPE} FMECA.

7.4.1 Module Selection

The starting point for FMECA is all the components of interest. In candidate systems, the components are equated on the system *modules*, as defined in the system design documents. For a software system, UML class diagrams detail the different modules and module relationships. Unfortunately, UML class diagrams do not directly apply to certification mechanisms; in fact, certification mechanisms suffer from the following important fact:

There exist no generally accepted design methods for certification mechanisms.

Recall from section 7.2.2 that an *information flow model* can be derived from any certification mechanism. In the absence of a comprehensive, generally accepted design document, this flow model's modules become the components for the GKG_{SEPE} FMECA. Whether using a flow model is optimal as a source for FMECA components is an open question.

Open Question 7.17. *What criteria should analysts use to select FMECA components in a filter model analysis?*

7.4.2 Failure Mode Determination

Determining the different ways in which components can fail relies to some extent on analyst intuition and experience. The less “concrete” the components are, the more analysts must rely on intuition and experience. Informally, “concreteness” indicates how tied to *physical reality* a module is, i.e., how closely a module functions to hardware. Inductive reasoning about failure modes is often easier for hardware modules than software modules, due to how well hardware failures are understood compared to software failures.

The components in the GKG_{SEPE} flow model are constructs that relate only to information; there are no physical sensors or actuators in GKG_{SEPE}. Therefore, this analysis deems the components as non-concrete, and relies on the author's intuition and experience with certification. Analysts with different intuition and experience could determine different failure modes for the components.

Open Question 7.18. *To what extent does analyst expertise affect the determination of failure modes in a filter model analysis?*

7.4.3 Effect and Criticality Determination

Like the determination of failure modes, the determination of effects of failure modes is based to some extent on analyst intuition and experience. The effects, being tied to their failure modes, are also related to information and therefore not concrete. Thus, the effects are determined through the author's intuition and

experience. Analysts with different intuition and experience could determine different effects for the failure modes.

Open Question 7.19. *To what extent does analyst expertise affect the determination of effects in a filter model analysis?*

Criticality analysis unifies the probability of failure modes and the severity of the effects. However, the novelty of analyzing GKG_{SEPE} as a system resulted in the following problems:

- There is no accepted way to derive probabilities for the failure modes, and no historical data from which to derive these probabilities.
- There is no accepted way to measure the criticality of the effects of the failure modes.

Thus, the analysis treats criticality *qualitatively*, like the treatment of risk in Chapter 1. Whether certification mechanisms are amenable to quantitative criticality analysis is an open question. Moreover, this analysis depends on the author's intuition and experience. Analysts with different intuition and experience could derive different criticality to each effect.

Open Question 7.20. *Can certification mechanisms be subjected to quantitative criticality analysis as part of a filter model analysis?*

Open Question 7.21. *To what extent does analyst expertise affect the derivation of criticality in a filter model analysis?*

7.4.4 Results

This section presents the results of the failure modes, effects, and criticality analysis on GKG_{SEPE} in the following order:

1. Measurements of FMECA metrics. Table 7.6 presents:
 - The metrics this analysis uses to assess the feasibility and yield of applying FMECA to GKG.
 - The values of these metrics.
 - The category (*feasibility* and/or *yield*) of each metric.
2. Discussion of FMECA metrics and measurements, including a qualitative discussion of acceptability.
3. Discussion of the limitations of these results, including open questions raised.

Note that the results do not include discovered faults_{cert}; the FMECA did not discover any new faults. Instead, the FMECA discovered:

- Failure modes for the faults_{cert} in the components; these faults were discovered in the HazOp and FTA.
- The possible effects of some of the failure modes.
- The circumstances under which a failure caused by a fault would be critical.

Feasibility and Yield Assessment

This analysis uses several metrics to assess feasibility and yield of applying FMECA to GKG_{SEPE}. Table 7.6 shows these metrics, the value at which each metric was measured, and what property each metric is designed to assess. We cannot meaningfully compare the measurements in Table 7.6 to the same measurements for FMECA of candidate systems.

The FMECA took 27 hours to complete and discovered no new faults_{cert}. However, the FMECA did generate non-trivial results:

- **Organization of the faults_{cert} discovered with HazOp and FTA.** The FMECA organized 8 of the 13 discovered faults_{cert} by the components that could harbor each fault_{cert}.
- **Failure modes and effects of the faults_{cert}.** The FMECA found at least one failure mode and effect for each of the 8 faults_{cert}.

Thus, the FMECA fulfilled both feasibility metrics – **time** and **generation of non-trivial results**. In this analysis, the FMECA achieved a **zero** yield for faults_{cert} and a **non-zero** yield for the other non-trivial results shown above.

Limitations

In addition to the limitations stated in the Open Questions in this section (Open Questions 7.17 through 7.20), the limitations on metric acceptability criteria stated in section 7.2.6 also apply to this FMECA and its metrics. The scope of the case study also limited this FMECA; recall that Case Study Assumptions 4.3 and 4.4 deem questions about fragment selection and conclusion combination out of scope of this analysis. The FMECA components included the “Fragment Selector” and “Fragment Conclusion Combiner” blocks from the GKG functional block diagram in Figure 7.2, but the failure modes and effects from these components are not examined further. Furthermore, faults_{cert} found in these components are not subjected to adapted safety engineering techniques in phase 2 of the evaluation. Thus, this work will not analyze $F_{cert-frag.oth}$ any further.

Metric	Value	Category
Hours to complete FMECA	27	Feasibility
Total number of FMECA rows	23	Feasibility
FMECA rows produced per hour	0.852	Feasibility
Total number of failure modes	22	Yield
Average faults _{cert} per component	2	Yield
Average percentage of total faults _{cert} per component	22.222%	Yield
Total number of effects	23	Yield
Average number of effects per failure mode	1.045	Feasibility/Yield
Average number of effects per fault _{cert}	2.556	Feasibility/Yield
Percentage of faults _{cert} organized by FMECA	61.538%	Feasibility

Table 7.6: Measurements for the FMECA on the GKG_{SEPE} certification mechanism.

7.5 Summary

This chapter presented:

- Measurements of feasibility and yield for various hazard analysis techniques that were adapted for use on certification mechanisms.
- The faults_{cert} discovered in the GKG certification mechanism using established hazard analysis techniques.

The sections throughout this chapter note 21 open questions about the analysis presented in this chapter.

The questions fell into two broad categories:

1. **Variance introduced by analyst expertise.** This analysis was conducted by the author. The author is not an expert in (a) the subject area of certification or (b) application of HazOp, FTA, and FMECA. The validity and utility of the results of *candidate system* hazard_{cand} analysis depend on the expertise of the analyst; by analogy, the results of *certification mechanism* hazard_{cert} analysis *also* depend on the expertise of the analyst.
2. **Fundamental unknowns.** Due to the novelty of this analysis, the optimal way to apply HazOp, FTA, and FMECA to certification mechanisms is unclear. In particular, the metrics used to assess feasibility and yield might not be the right metrics, and the acceptability criteria for the metrics presented are unknown.

Analyst Expertise Variance	Fundamental Unknowns
7.5, 7.6, 7.7, 7.8, 7.14, 7.15, 7.18, 7.19, 7.21	7.1, 7.2, 7.3, 7.4, 7.9, 7.10, 7.11, 7.12, 7.13, 7.16, 7.17, 7.20

Table 7.7: Categorization of the open questions in this chapter.

Table 7.7 shows to which category each Open Question belongs.

Chapter 8 provides a comprehensive summary of the faults_{cert} found in the analysis presented in this chapter, as these faults_{cert} are used in phase 2 of the evaluation. Chapter 8 also presents the results of phase 2 of the evaluation: seeding faults_{cert} into the evaluation, recording the results, and adapting safety engineering techniques to complete the filter repair cycle.

Chapter 8

GKG Fault Mitigation

Recall from Chapter 4 that the aim of Phase 4.2 of the case study was to generate $\text{fault}_{\text{cert}}$ mitigation strategies as part of the **filter repair cycle**. This chapter presents the results of phase 2.

Chapter 7 provided a list of $\text{faults}_{\text{cert}}$. In *candidate system* safety engineering, the next step is *fault treatment*. Chapter 3 briefly introduced the following broad categories of safety engineering techniques:

1. Fault avoidance.
2. Fault elimination.
3. Fault tolerance.
4. Fault forecasting.

Typically, *candidate system* engineers apply each in succession. First, fault avoidance is applied. After fault avoidance has been applied to the extent possible, engineers move on to fault elimination, and then to fault tolerance. Finally, when all the other techniques have been applied to the extent possible, fault forecasting is applied.

In this work, $\text{fault}_{\text{cert}}$ mitigation strategies are generated through adaptation of techniques in the above categories. Unfortunately, *how* to adapt these techniques is not established because of the novelty of this analysis. Thus, this analysis focuses mostly on fault elimination and fault tolerance. **Fault-specific** methods and **redundancy** are two techniques from each category:

- **Fault-specific methods** are *fault-elimination* techniques designed to eliminate specific classes of faults.
- **Redundancy** is a *fault-tolerance* technique that uses multiple copies of *identical* system modules in place of a single module. Simple redundancy mitigates random and transient faults in candidate systems.

This chapter presents adaptation and application of these safety engineering techniques to generate $\text{fault}_{\text{cert}}$ mitigation strategies.

Below is the safety engineering process followed in this work.

1. Start with faults_{cert} discovered in Chapter 7. Place these faults_{cert} into the GKG_{SEPE} mechanism.
2. Where appropriate, place corresponding faults_{cand} into the baseline safety case, creating an evaluation safety case.
3. Conduct certifications using GKG_{SEPE} and evaluation safety cases.
4. Generate fault mitigation strategies using the safety engineering literature.

Sections 8.1 through 8.4 elaborate on the conduction of each step in the process.

This chapter also presents the following results:

- **Outcome of GKG_{SEPE} certifications.** This evaluation examines the certification decisions of GKG_{SEPE} on:
 - The baseline DAIS safety case.
 - The evaluation safety cases.
- **Fault mitigation strategies.** Based on the outcome of the GKG_{SEPE} certifications, this analysis generates various fault mitigation strategies.

Section 8.5.1 presents the GKG_{SEPE} certification decisions, and section 8.5.2 presents the fault mitigation strategies.

8.1 GKG_{SEPE} Faults_{cert} Placement

The starting point for phase 2 of the evaluation was the faults_{cert} discovered in phase 1. Table 8.1 summarizes each of the faults_{cert} discovered in Chapter 7. These faults_{cert} are generally applicable to all instantiations of GKG. The specimen certification mechanism in this analysis is GKG_{SEPE}; therefore, the faults_{cert} should be instantiated for GKG_{SEPE}. Table 8.2 shows how this analysis places faults_{cert} into GKG_{SEPE}. Each table entry contains:

- The fault_{cert} designation.
- The part of GKG_{SEPE} in which the fault_{cert} is placed.
- How the fault_{cert} is placed.

The two following faults_{cert} shown in Table 8.1 are not analyzed in this chapter, because their scope is too large:

Fault Designation	Description
$F_{cert-insp.oth}$	The phased inspection is conducted incorrectly.
$F_{cert-qual.oth}$	A quality is corrupted.
$F_{cert-qual.est.inc}$	A quality is established incorrectly.
$F_{cert-qual.omi}$	A quality omitted from qualities list.
$F_{cert-dial.less}$	The dialectic issues fewer challenges than necessary.
$F_{cert-dial.rev}$	The dialectic does not challenge the fragment.
$F_{cert-dial.oth}$	The dialectic is corrupted.
$F_{cert-chal.oth}$	A challenge is corrupted.
$F_{cert-chal.omi}$	A challenge is omitted.

Table 8.1: Summary of faults_{cert} found in Phase 1 of the case study.

1. $F_{cert-insp.oth}$. Phased inspections can be conducted incorrectly in a wide variety of ways, with a wide variety of effects; one way is $F_{cert-qual.est.inc}$. As an example, consider a phased inspection conducted by non-experts. Such an inspection could be faulty by virtue of the inspectors missing details that can only be learned with extensive experience. These details can range from knowing the importance of the phase order to minute details about the qualities for which the inspectors examine. Most of these effects are difficult to predict, especially those of not conducting the phased inspection in the correct order. Thus, with the exception of $F_{cert-qual.est.inc}$, this analysis will not analyze faults_{cert} related to *conduction* of the phased inspection.
2. $F_{cert-dial.oth}$. The dialectic can be corrupted in a wide variety of ways; two of the ways are $F_{cert-dial.rev}$ and $F_{cert-dial.le}$. Other ways in which the dialectic can be corrupted are not clear from the analysis conducted in phase 1, because of the generality of $F_{cert-dial.oth}$.

Thus, phase 2 of the case study introduces two new assumptions:

Case Study Assumption 8.1. *The phased inspection process is conducted perfectly with supplied materials.*

Case Study Assumption 8.2. *$F_{cert-dial.rev}$ and $F_{cert-dial.le}$ are a reasonable approximation of $F_{cert-dial.oth}$.*

Each of the faults_{cert} in Table 8.2 could be placed into GKG_{SEPE} in many ways. This analysis chooses only one way for each fault_{cert}. The effect of placing the same fault_{cert} into a certification mechanism in multiple ways is an open question.

Open Question 8.1. *What is the effect of multiple placing of the same class of fault_{cert} into a certification mechanism?*

Fault Designation	Placement Method	Example
$F_{cert-qual.oth}$	Make a plausible corruption to the quality.	Change Q4: “Evidence availability and sufficiency” to “Evidence availability.”
$F_{cert-qual.est.inc}$	Make a plausible but faulty assumption.	Use Q6: “Assumption Necessity and Reasonableness.” Assume testing is sufficient to establish verification.
$F_{cert-qual.omi}$	Omit a quality.	Omit Q7, “Freedom from well-known fallacies.”
$F_{cert-dial.le}$	Select a fragment that requires multiple challenges. Fail to levy one of those challenges.	G.Misinformation.6 could have two challenges levied: 5.1 and 5.12. Fail to issue 5.12.
$F_{cert-dial.rev}$	Do not issue any challenges to a particular fragment.	Fail to issue 5.12 to a fragment that dictates issuing of 5.12.
$F_{cert-chal.oth}$	Make a plausible corruption to a challenge.	Change 5.11 to “Explain why you did not base your safety argument on conformance to a standard.”
$F_{cert-chal.omi}$	Omit a challenge.	Omit Challenge 5.11.

Table 8.2: How the discovered faults_{cert} are placed into GK_{GSEPE}.

8.2 Evaluation Safety Case Construction

In order to construct the evaluation safety cases for this analysis, a *fault mapping* and *faulty fragments* were needed. Section 8.2.1 expands on the fault mapping, section 8.2.2 expands on the construction of the faulty fragments, and section 8.2.3 how the faulty fragments were used to create the evaluation safety cases.

8.2.1 Fault Mapping

Recall the definition of **unwarranted acceptance** from Chapter 3: an unwarranted acceptance is the acceptance of an unsatisfactory system. In order for a certification mechanism, CM , to arrive at an unwarranted acceptance of a candidate system, CS , both of the following must be true:

- CM must contain at least one fault_{cert}.
- CS must contain at least one fault_{cand} that is not filtered out by the (faulty) CM .

In particular, faults_{cand} in CS can be *mapped* to the faults_{cand} in CM that produces an unwarranted acceptance for CS , and all other candidate systems with the same faults_{cand}. Figure 8.1 illustrates this concept. In this figure, F_{cert-1} is a fault_{cert}, and $F_{cand.1}$ through $F_{cand.3}$ are some possible faults_{cand} in the particular system domain for which the certification mechanism, CM , is designed. If a candidate system, $CS_{1,2}$, contains either

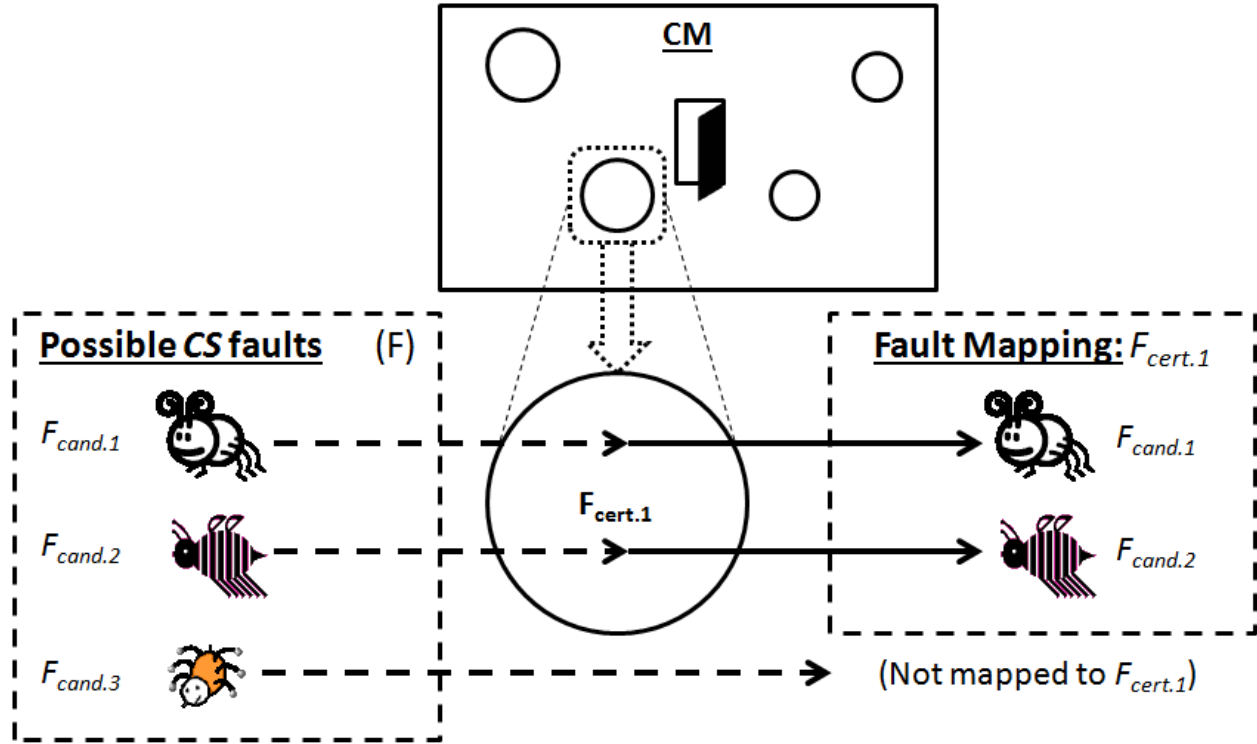


Figure 8.1: Hypothetical simple fault mapping for one $\text{fault}_{\text{cand}}$ in a hypothetical certification mechanism.

of $F_{\text{cand}.1}$ or $F_{\text{cand}.2}$, CM will not filter $CS_{1,2}$ out because of $F_{\text{cert}.1}$, i.e., CM will produce an unwarranted acceptance for CS . Thus, $F_{\text{cand}.1}$ and $F_{\text{cand}.2}$ map to $F_{\text{cert}.1}$. A different candidate system containing only $F_{\text{cand}.3}$, CS_3 , would not exercise $F_{\text{cert}.1}$. Thus, $F_{\text{cand}.3}$ does not map to $F_{\text{cert}.1}$.

Note that the example in the figure is (a) simple and (b) incomplete; in practice, $F_{\text{cand}.3}$ would be mapped to a $\text{fault}_{\text{cert}}$ of its own. Moreover, $\text{faults}_{\text{cand}}$ could be mapped to multiple $\text{faults}_{\text{cert}}$ and vice versa. Being neither one-to-one nor onto, this mapping is not a function, but a relation. Nevertheless, the term is useful for this discourse:

Placing $\text{faults}_{\text{cand}}$ that map into the $\text{faults}_{\text{cert}}$ discovered in phase 1 enables examination of these $\text{faults}_{\text{cert}}$ in a certification.

This work terms such a mapping a **fault mapping**.

Definition 8.1. Fault mapping. A fault mapping is a mapping of $\text{faults}_{\text{cert}}$ to $\text{faults}_{\text{cand}}$; the mapping maps $\text{faults}_{\text{cert}}$ to the $\text{faults}_{\text{cand}}$ that, if present in a candidate system, are not filtered out by the faulty certification mechanism.

Established system domains, e.g., infusion pumps, have extensive failure data derived from certified systems. For example, the FDA's MAUDE database [56] has over 2.5 million manufacturer reports, spanning

Fault_{cert}	Corresponding Fault_{cand}
$F_{cert-qual.oth}$	Evidence insufficiency in fragment solution elements.
$F_{cert-qual.est.inc}$	No formal specification/verification in fragment.
$F_{cert-qual.omi}$	Argument from ignorance in fragment.
$F_{cert-dial.le}$	No formal specification/verification in fragment.
$F_{cert-dial.rev}$	No formal specification/verification in fragment.
$F_{cert-chal.oth}$	N/A
$F_{cert-chal.omi}$	Arguing conformance to a standard in fragment.

Table 8.3: The partial GKG_{SEPE} fault mapping used in this analysis.

from 1996 to 2012¹. These collections of failure data provide ample information for constructing a comprehensive fault mapping. However, the SEPE domain is novel, so the only mapping from faults_{cert} to faults_{cand} is the partial mapping presented in Table 8.3. The only fault_{cert} that did not receive a corresponding fault_{cand} was $F_{cert-chal.oth}$; the reason for this absence is explained in section 8.3.

The fault mapping shown in Table 8.3 is incomplete, and the author selected the faults_{cand}. There are many possible faults_{cand}. The chosen set of faults_{cand} might not be the optimal set of faults_{cand} with which to:

- Demonstrate faults_{cert}.
- Engage in filter repair.

These concerns raise the following open questions:

Open Question 8.2. *What is the effect of an incomplete fault mapping on the results of the filter repair cycle?*

Open Question 8.3. *Which faults_{cand} are optimal for use in a filter model analysis?*

8.2.2 Faulty Fragments

The fault mapping in Table 8.3 enables the creation of faulty safety case *fragments*. These fragments can then be passed through GKG_{SEPE} to form the basis for mitigating the faults_{cand} exhibited in the fragments. Each of these fragments exhibits one fault_{cand}; each fault_{cand} is mapped to one fault_{cert} as per Table 8.3. The following sections expand on each of the faulty fragments. Each section contains:

- The faulty fragment, represented in GSN.

¹The FDA has even more records in the predecessor of the MAUDE database, the Medical Device Reporting (MDR) database, spanning from 1984 to 1996. The MDR database contains over 600,000 records.

- The **entrance node** of the faulty fragment in the baseline DAIS safety case. This entrance node is where the fragment is inserted *instead* of the baseline fragment.
- A description of the faulty fragment, including the deficiency represented in the faulty fragment.
- The faults_{cert} that map to the fault_{cand} exhibited in the faulty fragment.
- Additional notes, if necessary.

Evidence Insufficiency Fragment

Entrance node: `G_LackOfInformation.1.`

Description: Figure 8.2 shows the *Evidence Insufficiency* fragment, $Frag_{EI}$. $Frag_{EI}$ attempts to argue high assurance via the two-legged construction. The *argument* portion of the fragment is the same as the fragment in the baseline safety case. However, the difference between $Frag_{EI}$ and the original fragment is the following assumption:

The testing plan does not sufficiently stress the system.

An example of such a testing plan is a plan based around the use of **statement coverage**. Statement coverage is regarded as the weakest of test coverage metrics [79]; such a test plan would not be suitable for a high-assurance system.

Faults_{cert} that map to this fault_{cand}:

- $F_{cert-qual.oth.}$

Additional notes: None.

Lack of Formal Verification Fragment

Entrance node: `G_LackOfInformation.1,G_Misinformation.6.`

Description: Figure 8.3 shows the *Lack of Formal Verification* fragment, $Frag_{LOFV}$. $Frag_{LOFV}$ attempts to argue high assurance via testing *alone*. Recall from Chapter 5 that (a) formal verification is complementary to testing [57, 58] and (b) testing *alone* is not sufficient to establish ultra-high dependability [59].

Faults_{cert} that map to this fault_{cand}:

- $F_{cert-qual.est.inc}$
- $F_{cert-dial.le}$
- $F_{cert-dial.rev}$

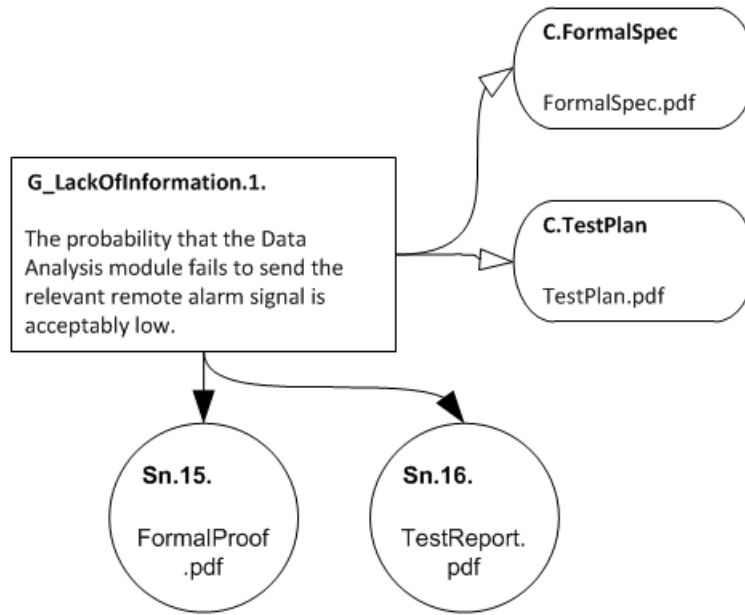


Figure 8.2: The Evidence Insufficiency Fragment, $Frag_{EI}$.

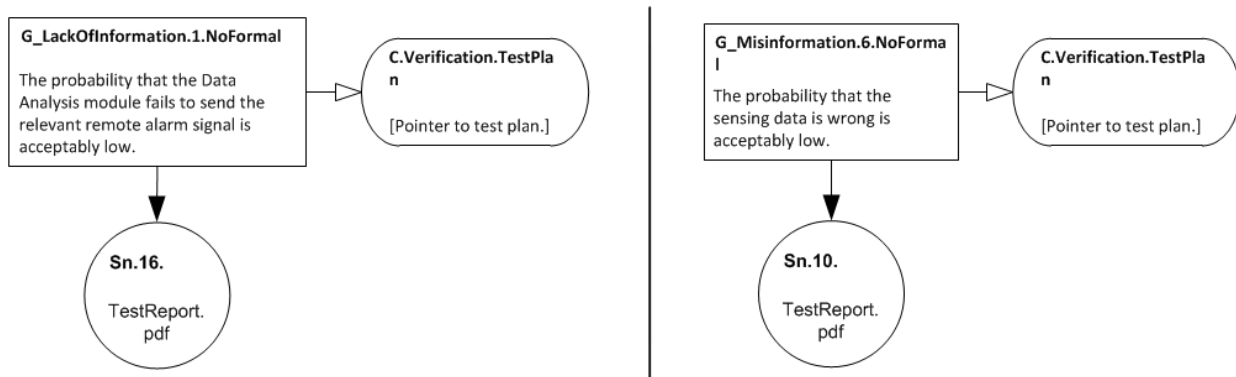


Figure 8.3: The Lack of Formal Verification Fragment, $Frag_{LOFV}$.

Additional notes: $Frag_{LOFV}$ is used in two different places in the argument,

`G_LackOfInformation.1` and `G_Misinformati.on.6`. Figure 8.3 shows two versions of this fragment; the two versions are identical except for the entrance node.

Argument from Ignorance Fragment

Entrance node: `G_Misinformati.on.5`.

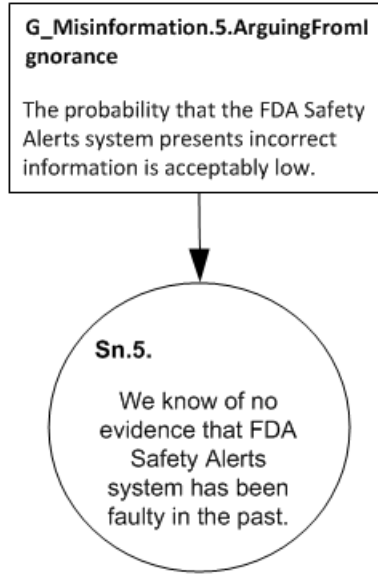


Figure 8.4: The Argument From Ignorance Fragment, $Frag_{AFI}$.

Description: Figure 8.4 shows the *Argument from Ignorance* fragment, $Frag_{AFI}$. $Frag_{AFI}$ attempts to argue for the safety of an *external component*, the FDA Safety Alerts system. The verification of external components is a non-trivial issue for critical systems [80, 81]; developers cannot simply assume that external components have the desired dependability attributes. $Frag_{AFI}$ argues that the FDA Safety Alerts system is sufficiently dependable because the writer of the fragment is unaware of any errors. This is an *argument from ignorance*; the presence of an argument from ignorance is a fallacy, as per the taxonomy of safety case fallacies described by Greenwell, et al. [47]

The FDA Safety Alerts system is a large, human-centric system, with many industrial collaborators. The probability that such a system is sufficiently dependable is unknown. Moreover, the probability that the system is *correct* at any given instant about the safety of a certain product is unknown; the purpose of the system is to rectify errors in FDA approval of products. Thus, patients could use products for which safety alerts are issued well after the product approval date [82]. Such usage is dangerous to the patient.

Faults_{cert} that map to this fault_{cand}:

- $F_{cert-qual.omi}$

Additional notes: None.

Conformance Argument Fragment

Entrance node: `Top`.

Description: Figure 8.5 shows the *Conformance Argument* fragment, $Frag_{CA}$. $Frag_{CA}$ attempts to argue overall safety via conformance to a standard. Recall from Chapter 5 the inadequacy of this argumentation strategy: arguing over hazards is (a) more direct and (b) more flexible than arguing conformance to a standard.

Faults_{cert} that map to this fault_{cand}:

- $F_{cert-chal.omi}$

Additional notes: The previous faulty fragments presented in this section were **leaf fragments**:

Definition 8.2. Leaf fragment. *A leaf fragment in a safety case is a fragment that contains leaf nodes, i.e., solutions or undeveloped elements.*

Leaf fragments are concluding fragments, i.e., there are no argument elements that continue *past* a leaf fragment. However, $Frag_{CA}$ is *not* a leaf fragment; `G_DAIS.2.Standard` replaces `Top`. The entire strategy of the safety case, and thus the content of the safety case itself, is changed. There is no corresponding exit node into the baseline DAIS safety case.

8.2.3 Evaluation Safety Cases

Each of the faulty fragments presented above are placed into *one* evaluation safety case. The effect of the presence of additional faulty fragments in a safety case is an open question.

Open Question 8.4. *What is the effect of multiple faulty fragments in a safety case?*

This work names each of these safety cases by the placed fragment and entrance node, e.g., $SC_{EI.G.LackOfInformation.1}$ is the baseline safety case with $Frag_{EI}$ placed at entrance node `G_LackOfInformation.1`. The baseline safety case is termed SC_{base} . Table 8.4 maps each faulty fragment to its corresponding evaluation safety case.

8.3 GKG_{SEPE} Certification

The next step in phase 2 was to use GKG_{SEPE} to conduct hypothetical certifications of each evaluation safety case; faults_{cert} were placed into GKG_{SEPE} in section 8.1. The certification for each fault_{cert} used one evaluation safety case from Table 8.4. For each certification, Table 8.5 shows:

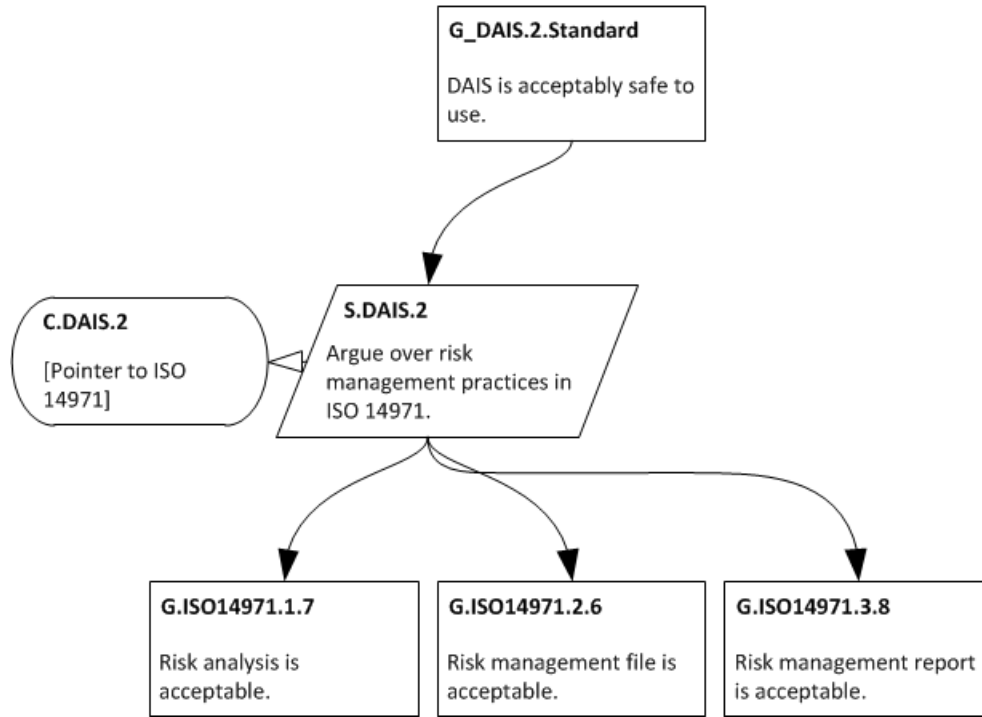


Figure 8.5: The Conformance Argument Fragment, *Frag_{CA}*.

Safety Case Name	Faulty Fragment	Entrance Node
SC_{base}	None	N/A
$SC_{EI.G_LackOfInformation.1}$	$Frag_{EI}$	G_LackOfInformation.1
$SC_{LOFV.G_LackOfInformation.1}$	$Frag_{LOFV}$	G_LackOfInformation.1
$SC_{LOFV.G_Misinformation.6}$	$Frag_{LOFV}$	G_Misinformation.6
$SC_{AFI.G_Misinformation.5}$	$Frag_{AFI}$	G_Misinformation.5
$SC_{CA.Top}$	$Frag_{CA}$	Top

Table 8.4: The names of each evaluation safety case and the faulty fragments contained in each evaluation safety case.

- A unique **certification designation**.
- The fault_{cert} being examined.
- The fault_{cand} example paired with that fault_{cert}. The original fault_{cand} example text from Table 8.1) is augmented with the name of the entrance node of the faulty fragment that the certification examines; this augmentation is shown in **bold**.
- The evaluation safety case used to exercise the fault_{cert}.

Cert. Des.	Fault Designation and Placement Method	Safety Case Used
C_1	$F_{cert-qual.oth.}$ Change Q4: “Evidence availability and sufficiency” to “Evidence availability.” Examine G.LackOfInformation.1.	$SC_{EI.G.LackOfInformation.1}$
C_2	$F_{cert-qual.est.inc.}$ Use Q6: “Assumption Necessity and Reasonableness.” Assume testing is sufficient to establish verification. Examine G.LackOfInformation.1. NoFormal.	$SC_{LOFV.G.LackOfInformation.1}$
C_3	$F_{cert-qual.omi.}$ Omit Q7: “Freedom from well-known fallacies.” Examine G.Misinformation.5.ArguingFromIgnorance.	$SC_{AFI.G.Misinformation.5}$
C_4	$F_{cert-dial.le.}$ G.Misinformation.6 could have two challenges levied: 5.1 and 5.12. Fail to issue 5.12.	$SC_{LOFV.G.Misinformation.6}$
C_5	$F_{cert-dial.rev.}$ Examine G.LackOfInformation.1.NoFormal. Fail to issue 5.12 to fragment.	$SC_{LOFV.G.LackOfInformation.1}$
C_6	$F_{cert-chal.oth.}$ Change 5.11 to “Explain why you did not base your safety argument on conformance to a standard.” Examine Goal.DIP.1.Mistreatment.	SC_{base}
C_7	$F_{cert-chal.omi.}$ Omit Challenge 5.11. Examine G.DAIS.2.Standard.	$SC_{CA.Top}$

Table 8.5: The characteristics of the certifications used in this analysis: the faults_{cert} each certification focuses on, the fault_{cand} example, and the evaluation safety case examined in the certification.

Recall from Chapter 6 that the baseline DAIS safety case, SC_{base} , is *assumed* to be high-assurance:

- SC_{base} argues directly over all identified hazards, an accepted high-assurance safety argumentation strategy.
- Each hazard is shown to be mitigated using a two-legged construction, an accepted high-assurance construction.
- The evidence documents that SC_{base} links to, e.g., test plans and test reports, accurately represent a high-assurance version of DAIS.

The construction of SC_{base} differed from typical industrial construction practices in the following ways:

- SC_{base} was constructed after most of the system development had been finished; industry professionals typically construct safety cases in parallel with system development.

- SC_{base} represents a research system, DAIS. As DAIS is a novel system, the requirements and identified hazards may be incorrect. Established types of industrial systems have a clearer picture of requirements and domain hazards.
- SC_{base} relies heavily on the two-legged construction. However, some of the identified hazards may not be amenable to mitigation via formal verification and/or testing.

Whether an industrial safety case would provide better results for filter repair is an open question.

Open Question 8.5. *What kind of safety cases should be used to examine $faults_{cert}$ in a filter model analysis?*

Each evaluation safety case, except SC_{CA} , uses mostly SC_{base} fragments: one faulty fragment replaces the corresponding baseline fragment at the entrance node (discussed in section 8.2.2). Essentially, fragments are examined as proxies for entire safety cases. SC_{base} was constructed specifically to pass GKG_{SEPE} certification. Industrial safety cases could be constructed differently, thus potentially containing more than one $fault_{cand}$. Moreover, GKG_{SEPE} is incomplete; a complete, FDA-ready GKG_{SEPE} instantiation (or other instantiations of GKG) could be affected in unpredictable ways by complex fault mappings. In addition, the author conducted all the hypothetical certifications in this analysis. A team of professional certifiers using GKG_{SEPE} (e.g., at the FDA) might have produced different results, due to expertise with certification and the candidate system domain. The effects of using industrial safety cases, a complete certification mechanism, and a team of professional certifiers are open questions.

Open Question 8.6. *What are the effects of using safety cases with multiple $faults_{cand}$ in a filter model analysis?*

Open Question 8.7. *What are the effects of examining a complete certification mechanism in a filter model analysis?*

Open Question 8.8. *To what extent does certifier expertise in certification affect the results of a filter model analysis?*

Open Question 8.9. *To what extent does certifier expertise in the candidate system domain affect the results of a filter model analysis?*

8.4 $Fault_{cert}$ Mitigation Strategy Generation

The last step in phase 2 was to generate $fault_{cert}$ mitigation strategies. Ideally, strategies would be drawn progressively from the following sources:

1. Fault avoidance.
2. Fault elimination.
3. Fault tolerance.

4. Fault forecasting.

The author generated strategies by adapting existing safety engineering techniques using his own expertise. The $\text{fault}_{\text{cert}}$ mitigation strategies were adapted from fault-specific methods, which effect fault elimination, and redundancy, which effects fault tolerance. Professional safety engineers with different expertise could generate different mitigation strategies. Moreover, different mitigation strategies could be generated by adapting additional existing techniques, as discussed in Chapter 3:

- Fault avoidance techniques.
- Different fault elimination techniques,
- Different fault tolerance techniques, including design diversity and different kinds of redundancy, e.g., temporal.
- Fault forecasting techniques.

The effects of engineer expertise and adapting additional techniques are open questions.

Open Question 8.10. *To what extent does engineer expertise affect the generation of $\text{fault}_{\text{cert}}$ mitigation strategies in the filter repair cycle?*

Open Question 8.11. *What additional $\text{fault}_{\text{cert}}$ mitigation strategies could be derived by adapting different safety engineering techniques?*

8.5 Results

This section presents the results of GKG_{SEPE} safety engineering in the following order:

1. Certification results. Section 8.5.1 elaborates on each certification. For each certification, Table 8.6 presents:
 - The safety case used.
 - The $\text{fault}_{\text{cert}}$ being examined.
 - The decision of the certification: **positive** or **negative certification**.
 - The type of **certification accident** caused by the certification decision.
2. Section 8.5.1 also contains a discussion of the certification results.
3. Section 8.5.2 presents the generated $\text{fault}_{\text{cert}}$ mitigation strategies. Table 8.7 shows the $\text{faults}_{\text{cert}}$ and the strategies generated to mitigate them.
4. Section 8.5.2 also presents the measurements of safety engineering metrics. Table 8.8 shows:

Certification	Fault _{cert} Examined	Safety Case Used	Decision	Certification Accident?
C_1	$F_{cert-qual.oth}$	$SC_{EI.G.LackOfInformation.1}$	Positive	Unwarranted Acceptance: YES
C_2	$F_{cert-qual.est.inc}$	$SC_{LOFV.G.LackOfInformation.1}$	Negative	Warranted Rejection: NO
C_3	$F_{cert-qual.omi}$	$SC_{AFI.G.Misinformation.5}$	Positive	Unwarranted Acceptance: YES
C_4	$F_{cert-dial.le}$	$SC_{LOFV.G.Misinformation.6}$	Positive	Unwarranted Acceptance: YES
C_5	$F_{cert-dial.rev}$	$SC_{LOFV.G.LackOfInformation.1}$	Positive	Unwarranted Acceptance: YES
C_6	$F_{cert-chal.oth}$	SC_{base}	Negative	Unwarranted Rejection: YES
C_7	$F_{cert-chal.omi}$	$SC_{CA.Top}$	Positive	Unwarranted Acceptance: YES

Table 8.6: Certification results for all evaluation certifications.

- The metrics this analysis uses to assess the feasibility and yield of adapting safety engineering techniques to GKG_{SEPE}.
- The values of these metrics.
- The category (*feasibility* and/or *yield*) of each metric.

8.5.1 Certification Outcomes

Table 8.6 shows the certification decisions for C_1 through C_7 . The following sections elaborate on how each certification decision was reached and the characteristics of the certification decision.

C₁: F_{cert.qual.oth}

C_1 examined fault_{cert} $F_{cert-qual.oth}$, the fault_{cert} created when a GKG quality is corrupted. The GKG quality chosen was Q4:

“Evidence availability and sufficiency.”

The corruption was to remove “sufficiency,” resulting in the following quality:

“Evidence availability.”

C_1 used evaluation safety case $SC_{EI.G.LackOfInformation.1}$, the safety case with faulty fragment $FrageI$ inserted at $G_LackOfInformation.1$.

Recall from section 8.2.2 that $Frag_{EI}$ showed an evidence *insufficiency* in the testing plan. However, examining the fragment only for evidence *availability* would not uncover this $fault_{cand}$. Thus, C_1 arrived at a **positive** certification decision for a system that has not been adequately tested, resulting in an **unwarranted acceptance**.

C₂: F_{cert.qual.est.inc}

C_2 examined $fault_{cert} F_{cert-qual.est.inc}$, the $fault_{cert}$ exercised when the phased inspection incorrectly establishes a GKG quality. The GKG quality chosen was Q6:

``Assumption Necessity and Reasonableness.``

The assumption made was:

``Assume testing is sufficient to establish verification.``

C_2 used evaluation safety case $SC_{LOFV.G_LackOfInformation.1}$, the safety case with faulty fragment $Frag_{LOFV}$ inserted at $G_LackOfInformation.1$.

Recall from section 8.2.2 that $Frag_{LOFV}$ argued for verification using only testing-based evidence. Given that Q6 is established *incorrectly*, the assumption that testing is sufficient is *incorrectly* deemed necessary and reasonable. However, $Frag_{LOFV}$ dictates levying Challenge 5.12. $Frag_{LOFV}$ failed Challenge 5.12. Thus, C_2 arrived at a **negative** certification decision for a system that made an unreasonable assumption, resulting in a **warranted rejection**.

C₃: F_{cert.qual.omi}

C_3 examined $fault_{cert} F_{cert-qual.omi}$, the $fault_{cert}$ created when a GKG quality is omitted. The GKG quality chosen was Q7:

``Freedom From Well-Known Fallacies.``

C_3 used evaluation safety case $SC_{AFI.G_Misinformation.5}$, the safety case with faulty fragment $Frag_{AFI}$ inserted at $G_Misinformation.5$.

Recall from section 8.2.2 that $Frag_{AFI}$ argued *from ignorance* that an external component, the FDA Safety Alerts system, was safe. The **argument from ignorance** fallacy is described in the safety case fallacy taxonomy created by Greenwell, et al. [47]. However, omitting Q7 caused the certification to not examine the fragment for common fallacies, including fallacies in the Greenwell taxonomy. Thus, C_3 arrived at a **positive** certification decision for a system in which an external component was not properly verified, resulting in an **unwarranted acceptance**.

C₄: F_{cert.dial.le}

C_3 examined $\text{fault}_{\text{cert}} F_{\text{cert-dial.le}}$, the $\text{fault}_{\text{cert}}$ exercised when the dialectic issues less challenges than necessary for a fragment. C_4 used evaluation safety case $SC_{LOFV.G.Misinformation.6}$, the safety case with faulty fragment $Frag_{LOFV}$ inserted at `G.Misinformation.6`. Of the 14 challenges in GKG_{SEPE} , 2 are immediately applicable to $Frag_{LOFV}$: 5.1 and 5.12. These challenges are reproduced below:

Challenge. 5.1. *Explain why you did not provide a testing report for this OTS software.*

Challenge. 5.12. *Explain why you argued that testing alone is sufficient for verification.*

In `G.Misinformation.6`, the argument makes reference to “sensing data,” which come from an external instrument. The certification deemed Challenge 5.1 passed because of the assumption that DAIS was built using high-assurance components. High-assurance sensor arrays would come with manufacturer guarantees of reliability, obviating the need for an OTS testing report by DAIS developers. However, as $Frag_{LOFV}$ argues verification of sensor data by testing only, Challenge 5.12 would *not* have been passed *if the challenge had been issued*. Because Challenge 5.12 was not issued, C_4 arrived at a **positive** certification decision for a system that had not been adequately verified, resulting in an **unwarranted acceptance**.

C₅: F_{cert.dial.rev}

C_5 examined $\text{fault}_{\text{cert}} F_{\text{cert-dial.rev}}$, the $\text{fault}_{\text{cert}}$ exercised when the dialectic fails to issue *any* challenges for a fragment. C_5 used evaluation safety case

$SC_{LOFV.G.LackOfInformation.1}$, the safety case with faulty fragment $Frag_{LOFV}$ inserted at `G.LackOfInformation.1`.

`G.LackOfInformation.1` argues the safety of a DAIS software module, the remote alarm signaling sub-module of the Data Analysis module. Inserting $Frag_{LOFV}$ at `G.LackOfInformation.1` means that the safety of the remote alarm signal module was not properly verified. Sn.16 points to a testing report as direct evidence of verification, so certifiers should issue *at least* Challenge 5.12 against this fragment. If Challenge 5.12 were issued, the fragment would fail. However, no challenges were issued due to $F_{\text{cert-dial.rev}}$; the fragment passed certification. Thus, C_5 arrived at a **positive** certification decision for a system that had not been adequately verified, resulting in an **unwarranted acceptance**.

C₆: F_{cert.chal.oth}

C_6 examined $F_{\text{cert-chal.oth}}$, the $\text{fault}_{\text{cert}}$ exercised when a challenge is corrupted. C_6 used evaluation safety case SC_{base} , the baseline safety case. The challenge chosen was Challenge 5.11:

Challenge. 5.11. *Explain why you based your safety argument on conformance to a standard.*

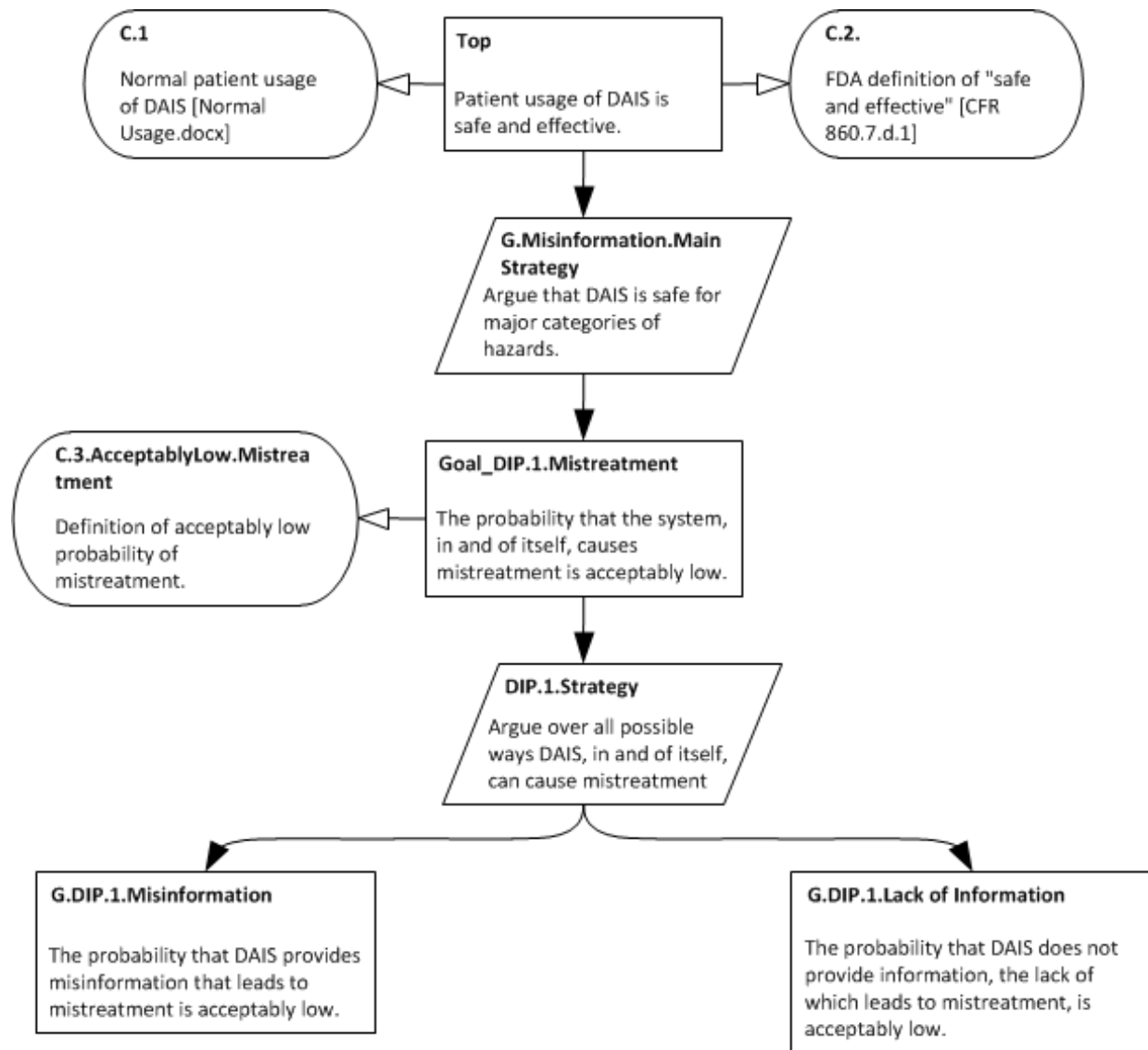


Figure 8.6: The top-level fragment of SC_{base} , against which Challenge 5.11 should have been issued.

Challenge 5.11 was designed to be issued against the top-level fragment, because the top-level fragment contains the overall argumentation strategy of the safety case. Figure 8.6 shows the top-level fragment of SC_{base} . The corruption to the challenge was to reverse the meaning, adding “did not” between “you” and “base[d].”

Challenge 8.1. *Explain why you did not base your safety argument on conformance to a standard [emphasis added].*

The corrupted challenge, Challenge 8.1, was issued against the top-level fragment of SC_{base} . The fragment failed certification, despite being a higher-assurance argumentation strategy than arguing standards

conformance. Thus, C_6 arrived at a **negative** certification decision for a system that was adequately safe, resulting in an **unwarranted rejection**.

C₇: $F_{\text{cert.chal.omi}}$

C_7 examined $F_{\text{cert-chal.omi}}$, the fault_{cert} exercised when a challenge is omitted. C_7 used evaluation safety case $SC_{CA.Top}$, the safety case with faulty fragment $Frag_{CA}$ inserted at the top level. The challenge chosen was Challenge 5.11, a challenge designed to be issued against the top-level fragment.

Recall from section 8.2.2 that $Frag_{CA}$ argued safety based on conformance to a standard. Further recall from Chapter 5 the inadequacy of such a strategy compared to arguing over hazards. However, C_7 failed to issue Challenge 5.11 due to $F_{\text{cert-chal.omi}}$. Thus, C_7 arrived at a **positive** certification decision for a system whose safety argument had an inadequate strategy, resulting in an **unwarranted acceptance**.

Discussion

Of the 7 certifications, 5 (C_1, C_3, C_4, C_5, C_7) had the expected certification decision of unwarranted acceptance, 1 (C_6) had the expected certification decision of unwarranted rejection, and 1 (C_2) had an *unexpected* decision of warranted rejection. Two certifications deserve special mention, C_2 and C_6 :

- C_2 . The warranted rejection indicates that the qualities and challenges in a GKG instantiation can provide some redundancy for each other, depending on the instantiation. In this work, GKG_{SEPE} was instantiated with Challenge 5.12, which protected against the common assumption, “Assume testing is sufficient to establish verification.” Other redundancies in GKG_{SEPE} are not identified. Moreover, a complete GKG_{SEPE} instantiation could have far more redundancies, decreasing its probability of an unwarranted acceptance. The effect of redundancies on the probability of unwarranted acceptance is an open question. Furthermore, whether a decrease in the probability of unwarranted acceptance necessarily affects the probability of unwarranted rejection is also an open question.

Open Question 8.12. *What is the effect of redundancies in a GKG instantiation on its probability of unwarranted acceptance?*

Open Question 8.13. *What is the effect of redundancies in a GKG instantiation on its probability of unwarranted rejection?*

- C_6 . C_6 raised two issues:
 1. The unwarranted rejection reached by C_6 indicates Faults_{cert} that corrupt a GKG element – a challenge or quality– E , can corrupt E such that the stringency of E is *increased* as well as decreased

(as in the case of the other $\text{faults}_{\text{cert}}$). Increased stringency results in a higher probability of unwarranted rejection, while decreased stringency results in a higher probability of unwarranted acceptance. The extent of this effect an open question.

Open Question 8.14. *What is the effect of corruption of GKG elements on the probability of unwarranted acceptance?*

Open Question 8.15. *What is the effect of corruption of GKG elements on the probability of unwarranted rejection?*

2. Recall from Chapter 4 that this analysis assumes a *single* submission action for simplicity, whereas the original GKG paper advocated an iterative approach, similar to the “early and often” advocated by Kelly [13]. Following the unwarranted rejection of SC_{base} due to the corrupted challenge, Challenge 8.1, the iterative process would dictate that the top-level fragment of SC_{base} would be changed to pass the *corrupted* challenge. Following the iterative approach in such a case would result in an eventual **unwarranted acceptance**.

Overall, this analysis is more limited than the analysis presented in Chapter 7, especially by available resources. Nevertheless, the results point toward the need for the last part of the filter repair cycle: generating $\text{fault}_{\text{cert}}$ mitigation strategies.

8.5.2 Generated $\text{Fault}_{\text{cert}}$ Mitigation Strategies

The results of the sample certifications (presented in section 8.5.1), while limited, suggest that $\text{faults}_{\text{cert}}$ can have serious effects. To mitigate these effects, the filter model analysis framework includes a **filter repair** mechanism designed to occur prior to deployment of a certification mechanism. Table 8.7 shows the repairs that the filter repair made for GKG_{SEPE} . Each repair is designed to mitigate one $\text{fault}_{\text{cert}}$; some $\text{faults}_{\text{cert}}$ are assigned multiple repairs. These repairs are inspired by safety engineering. Table 8.7 also links the repair to the safety engineering technique that inspired the form of the repair. Some repairs, e.g., a multi-inspector phase for each inspection GKG calls for, are analogous to classic hardware redundancy techniques. Others are adapted from recommendations of various standards, e.g., requiring qualification of interoperable devices. The rest are derived from the author’s intuition about GKG.

Note that the adaptation of recommendations of standards is, in effect, a hybridization of the goal-based and prescriptive approaches. Judicious application of such recommendations is a good “middle ground.” Certification mechanisms can reap the benefits of both the goal-based approach and the prescriptive approach while not suffering from the disadvantages. Over-inclusion of standards recommendations would give the

Fault Designation	Solution	Technique	
$F_{cert-qual.est.inc}$	Include a list of commonly-made erroneous assumptions.	Fault-specific	
	Multi-inspector phase	Redundancy	
	Applicant: Create comprehensive glossary. Certifier: Inspect glossary before starting fragment inspections.	Fault-specific	
	Require rigorous inspections of all documents included in fragments.	Fault-specific/	Redundancy
$F_{cert-qual.omi}$	Treat GKG instantiation as a system with inspections of qualities list before deployment. Checklist derived from qualities list in GKG paper.	Fault-specific/	Redundancy
$F_{cert-qual.oth}$	Treat GKG instantiation as a system with inspections of qualities list before deployment. Checklist derived from qualities list in GKG paper.	Fault-specific/	Redundancy
$F_{cert-chal.oth}$	Treat GKG instantiation as a system with inspections of challenges list before deployment. Checklists derived from GKG paper.	Fault-specific/	Redundancy
$F_{cert-chal.omi}$	Treat GKG instantiation as a system with inspections of challenges list before deployment. Checklists derived from challenge categories in GKG paper.	Fault-specific/	Redundancy
$F_{cert-dial.rev}$	Link document safety case patterns to relevant challenges.	Fault-specific	
	Require qualification of interoperable devices. No more grandfathering (abbreviated 510(k)).	Fault-specific	
$F_{cert-dial.le}$	Issue at least 2 challenges per fragment.	Redundancy	

Table 8.7: Fault_{cert} mitigation strategies and the safety engineering techniques from which they are derived.

certification mechanism the same disadvantages that prescriptive standards have, thus defeating the point of the filter repair cycle.

Feasibility and Yield Assessment

This analysis uses several metrics to assess feasibility and yield of generating fault_{cert} mitigation strategies to apply to GKG_{SEPE}. Table 8.8 shows these metrics, the value at which each metric was measured, and what property each metric is designed to assess.

Filter repair took 4 hours to complete and generated at least one mitigation strategy for each fault_{cert}. Thus, filter repair fulfilled both feasibility metrics – **time** and **generation of non-trivial results**. In this analysis, filter repair addressed 100% of the analyzed faults_{cert}, with an average of 1.286 mitigation strategies per fault_{cert}. Thus, filter repair achieved a **non-zero** yield.

Metric	Value	Category
Hours taken to produce mitigation strategies	4	Feasibility
Total fault _{cert} mitigation strategies	9	Feasibility
Average number of fault _{cert} mitigation strategies per fault _{cert}	1.286	Feasibility/Yield
Percentage of faults _{cert} with generated mitigation strategies	100 %	Feasibility/Yield
Mitigation strategies generated per hour	2.25	Feasibility/Yield

Table 8.8: Measurements for fault mitigation strategy generation for the GKG_{SEPE} certification mechanism.

Limitations

In addition to the limitations stated in the Open Questions in this chapter (Open Question 8.1 through 8.15), filter repair is subject to limitations on metric acceptability criteria (cf. section 7.2.6).

Open Question 8.16. *In a filter model analysis, what metrics should analysts measure to determine the feasibility and yield of filter repair?*

Open Question 8.17. *In a filter model analysis, what are the acceptability criteria for the metrics chosen to determine feasibility and yield of filter repair?*

In this analysis, filter repair addressed 100% of the analyzed faults_{cert}. Assuming that the mitigation strategies are 100% effective, this means that filter repair had a 100% success rate. Intuitively, this number seems implausible. Fault elimination is preferable to fault tolerance in *candidate systems*, implying that fault tolerance has a lower success rate than fault elimination.

However, there is no way to determine *a priori* the efficacy of a particular fault_{cert} mitigation strategy for *certification mechanisms*. In general, without a large-scale empirical evaluation, there is no way to empirically determine the efficacy of any particular fault_{cert} mitigation strategy. These strategies are well-studied for candidate systems, so analysts could make educated guesses about strategy efficacy based on previous applications to candidate systems. However, the strategies have never been examined for certification systems; whether the strategies have similar levels of efficacy is an open question. Furthermore, this chapter presents only *one* iteration of the filter repair cycle. The effect of subsequent iterations in reducing the probability of unwarranted acceptance for certification mechanisms is an open question. A related open question is the value of the **termination criterion** of the FRC, i.e., after how many iterations should analysts terminate the FRC and proceed with deployment of the certification mechanism.

Open Question 8.18. *How effective are $\text{fault}_{\text{cert}}$ mitigation strategies?*

Open Question 8.19. *What is the effect of subsequent iterations of the filter repair cycle on the probability of unwarranted acceptance?*

Open Question 8.20. *After how many iterations should analysts terminate the FRC?*

8.6 Summary

This chapter presented:

- The concept of a **fault mapping** and how fault mapping contributes to the examination and understanding of $\text{faults}_{\text{cert}}$.
- Examples of faulty safety case fragments.
- Examination of GKG_{SEPE} $\text{faults}_{\text{cert}}$ through hypothetical certification of several evaluation safety cases.
- The $\text{fault}_{\text{cert}}$ mitigation strategies generated by filter repair.
- Measurements of feasibility and yield for the process of filter repair.

The sections throughout this chapter note 20 open questions about the concepts presented in the chapter. As in Chapter 7, the questions fell into two broad categories:

1. **Variance introduced by analyst expertise.** This analysis was conducted by the author. The author is not an expert in (a) safety case construction, (b) certification process, nor (c) $\text{fault}_{\text{cand}}$ mitigation techniques. The validity and utility of the results of *candidate system* $\text{fault}_{\text{cand}}$ mitigation depend on the expertise of the analyst; by analogy, the results of *certification mechanism* $\text{fault}_{\text{cert}}$ mitigation *also* depend on the expertise of the analyst.
2. **Fundamental unknowns.** Due to the novelty of this analysis, the optimal way examine $\text{faults}_{\text{cert}}$ and generate mitigation strategies for $\text{faults}_{\text{cert}}$ is unclear. Furthermore, the effects of the mitigation strategies and the optimal amount of iterations to execute are unclear. In addition, the metrics used for feasibility and yield might not be the right metrics, and the acceptability criteria for the metrics presented are unknown.

Table 8.9 shows to which category each Open Question belongs.

The contribution of this work lies in the introduction of the filter model and the attendant comprehensive analysis framework. The results presented in Chapter 7 and this chapter offer confidence that the filter model is an effective framework for analysis and repair of certification mechanisms.

Analyst Expertise Variance	Fundamental Unknowns
8.5, 8.8, 8.9, 8.10	8.1, 8.2, 8.3, 8.4, 8.6, 8.7, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20

Table 8.9: Categorization of the open questions in this chapter.

Chapter 9

Related Work

9.1 Prescriptive Standards

Papadopoulos and McDermid [83] summarize, compare, and contrast various standards across safety-critical domains. The authors assert that “[i]t is accepted that only with respect to hardware safety it is possible to quantify and apply reliability prediction in accessing whether safety requirements have been met” and that “[e]ffective defense against [systematic] faults can only be assured by the implementation of the quality and safety management conditions specified within the standards.” These assertions, made in 1999 (although not for the first time), are the reason for continuing difficulty with system dependability assessment and, by extension, certification.

Commonalities between the standards were:

- The notion of certification, based on evidence of both followed processes and evidence about the system integrity itself.
- The notion of safety as freedom from unacceptable risk.
- System safety requirements and how to derive them.
- Development and safety processes at a structural and semantic level.
- The presence of development assurance/integrity levels.
- Requirement of system verification activities driven by the functional safety and integrity requirements.
- Standards present the structure of a safety case, whether explicitly or implicitly.

Areas of divergence were:

- Development and safety process.
- Derivation and allocation of functional safety requirements.

- Verification and validation.
- Certification and structure of the safety case.
- Treatment of integrity levels (the primary area of difference).

German [5] presents the results of practical application of static analysis to various safety-critical systems in the aircraft domain. The systems varied in size from 3,000 to 300,000 lines of code in various languages. German's prime result is that a comparison of static analyses of code adhering to DO-178B Levels A and B found no discernible difference between the two levels. This result indicates that "[e]ven the most extensive testing does not remove the anomalies found by static code analysis." This result heavily implies that adhering to more rigorous prescriptive standards does not necessarily result in a higher-assurance system, especially for certain classes of faults. No other such analyses exist, let alone comprehensive analyses of the faults of prescriptive standards.

Rouvroye and Brombacher [84] present the problem of standards for safety instrumented systems (SIS). SIS are used in the process industry to perform safety functions. Some common standards for these systems, e.g., the German DIN [85], do not require quantitative analysis, relying instead on qualitative analysis and expert judgment. Other standards, e.g., IEC 61508 [53], prescribe quantitative risk analysis but do not prescribe analysis methods. Adherence to these standards results in the safety of a SIS being quantified into a "safety integrity level" (SIL); guidelines on determination of a system's SIL are the only guidelines given.

The authors give an overview of analysis techniques, including:

- Expert analysis.
- FMEA.
- Parts count analysis [86].
- Reliability block diagrams [87].
- Hybrid analysis [53].
- FTA.
- Markov analysis [88].
- Enhanced Markov Analysis [89].

The authors compare these techniques using a 2 out of 3 (2oo3) system with a single sensor and single actuator/final element, finding the following:

- Different techniques account for different aspects of system behavior, meaning that techniques are complementary.
- Only quantitative techniques can calculate probabilities of dangerous failures. However, the techniques are based on models that are based on possibly uncertain data.

- Parts count analysis resulted in the highest (i.e., worst) time-average probability of failure on demand (PFD) and thus the lowest SIL, SIL0. EMA resulted in a range that contained the highest (i.e., best) PFD and thus the highest SIL, SIL2.

The main conclusion that the authors draw is that, even with the same set of data, different analysis techniques can result in different integrity levels. This result is shown by the disparity between the parts count analysis and EMA results.

9.2 Goal-Based Standards

Sokolsky, Lee, and Heimdahl [90] summarize problems in the broad area of software certification, and the specific area of medical devices. Challenges in software certification include:

- Lack of objectivity in software dependability assessment.
- Doubts if there is a correlation between practices required by standards and dependability (adherence assumption).
- Standards discourage adoption of new, more effective techniques because of the regulatory risk associated with deviation from the relevant standard.
- Model-based development is used to decrease cost (financial and temporal) and increase dependability. However, there is no established solution to the model validation problem.

Problems specific to medical devices include:

- Devices that are too safe; physicians may need to override restrictions in order to provide effective, creative care.
- Patients are the largest source of uncertainty; this source of uncertainty is difficult to model.
- Medical devices increasingly require network interoperability. However, different device developers do not necessarily design devices for interoperability.

Sokolsky, Lee, and Heimdahl propose an evidence-based approach, essentially advocating safety cases. The paper summarizes assurance case definitions, approaches, and patterns. The paper also proposes certification of virtual devices, i.e., scenarios based on types of devices that are networked together.

Penny, et al. [7] recognize the magnitude of the paradigm shift from the prescriptive to goal-based approach. They cite past experience, which shows that while the goal-based approach remedies the problems of prescriptive standards, the magnitude of the shift indicates that industry needs significant support to ease the transition. Penny, et al. discuss the motivation behind the goal-based approach, illustrating with an example from CAP 670. The paper states that prescriptive standards:

- Only require developers to carry out the mandated prescriptions in order to fulfill legal responsibility.
- Cannot cope with innovation; as a result, they range from inappropriate to dangerous in technically innovative industries.
- Are a snapshot of best practices at the time they are written; as a result, they become outdated quickly in industries with rapidly changing best practices.
- Create a barrier to open markets, especially when they are overly restrictive.

The goal-based approach suffers none of these disadvantages, but the creation of a convincing safety case is a difficult process. CAP 670 SW 01 provides guidance on creating a safety case that fulfills the safety requirements of CAP 670 without prescribing how the requirements are to be fulfilled. Penny, et al. present an example — air traffic management software — and introduce the concept of the assurance evidence level (AEL). Informally, the AEL represents the rigor of supporting arguments and evidence; the higher the AEL, the greater the rigor.

McDermid [91] addresses evidence regarding variation in safety integrity levels and examines assumptions central to the standards-based approach. The two main assumptions are:

- Processes for higher SILs produce software of higher integrity.
- Processes for SILs are more expensive, and are thus only warranted with severe failure consequences.

McDermid cites work that shows little correlation between “obvious” process factors and failure rate [92], and work that shows no significant cost variations between SILs 0, 2, and 3. Neither assessment is definitive, but both challenge the central standards assumptions. Essentially, standards prescriptions are not safety-focused, but quality- and repeatability-focused.

The paper also discusses potential advantages of product-based (as opposed to process-based) evidence for safety assessment, i.e., the goal-based approach. At a high level, this approach treats software as any other system component.

Lastly, McDermid discusses the ALARP principle, along with how it can be shown, in software-intensive systems, that risks have been reduced ALARP.

9.3 Assurance Cases

Kelly [13] defines the role and purpose of a safety case:

A safety case should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context.

Various standards, including 00-55 [60] provide different pieces of this definition. Following from the definition, Kelly [13] states that the three main pieces of a safety case are:

- Safety requirements and objectives.
- The safety argument.
- Safety evidence.

Safety evidence supports the safety argument, which argues that the system fulfills the safety requirements and objectives. Argument and evidence are closely coupled; Kelly [13] states:

Argument without supporting evidence is unfounded, and therefore unconvincing. Evidence without argument is unexplained it can be unclear that (or how) safety objectives have been satisfied.

Kelly recommends that GSN be used for recording safety arguments because natural language in safety cases often becomes ambiguous and unclear. Natural language can also contain many cross-references to evidence and other parts of the safety argument.

Kelly [13] also presents the problems with producing a safety case at the end of the development lifecycle, including:

- Significant redesign due to the inability of developers to construct a satisfactory safety argument.
- Final safety cases with less robust safety arguments, due to safety case developers having no influence over the design of the system.
- Safety cases with lost safety rationale, due to lack of recording of rationale at the time of system design.

Based on industry experience, Kelly recommends incremental safety case development, which integrates the safety case development into the system development through three safety cases: the preliminary safety case, the interim safety case, and the operational safety case. GSN can be used to record the preliminary safety case; the interim and operational safety cases can be evolved from the preliminary safety case using GSN as well.

Kelly and McDermid [11] introduce the concept of safety case patterns, i.e., pieces of safety arguments that are commonly reused. The paper observes that many arguments reuse parts of arguments informally. Informal reuse has the following problems:

- Opportunities for reuse must be identified manually.
- Informal reuse is ad hoc and thus not predictable nor dependable.
- Informal reuse can be inappropriately applied.
- Informal reuse is not traceable, nor does informal reuse offer consistency or process maturity.
- There exists no mechanism for recording reusable argument parts, resulting in possible loss of knowledge.

Safety case patterns are designed to remedy these problems. These patterns capture successful safety argument approaches, including:

- Solutions that evolved over time.
- Solutions involving company expertise.
- Approaches that proved successful for certification.
- Special skills and knowledge associated with particular domains.

Kelly and McDermid present several patterns, encoding their patterns in GSN for clarity and consistency of the pattern library and using the documentation format proposed by Gamma, et al. [93]. The appendix of the paper presents the ALARP pattern. Kelly and McDermid conclude that patterns will help to improve safety case construction and reuse.

Bishop, Bloomfield, and Guerra [8] summarize the characteristics of the goal-based approach and contrast the goal-based approach with the prescriptive approach and vulnerability assessment. The goal-based approach is top down and entails demonstrating, through argument and evidence, that safety goals are fulfilled by a candidate system. The prescriptive approach is also top down and entails demonstrating compliance to known safety standards. Vulnerability assessment is bottom up and entails demonstrating that the existence of potential vulnerabilities within a system is acceptably safe.

The goal-based approach is based on the idea of supporting sophisticated arguments about characteristics of a candidate system, e.g., safety. Toulmin's earlier work on argumentation [94] identifies the general shape of an argument, regardless of application domain (e.g., scientific, legal, etc.):

- A claim, i.e., an assertion that the arguer makes.
- Grounds, i.e., the facts that are purported to support a claim.
- A warrant, i.e., the reasoning that links the grounds to the claim.
- Backing, i.e., support for the warrant.

Bishop, Bloomfield, and Guerra focus on the claims-argument-evidence notation (CAE), presenting its genesis from Toulmin: claims are the same as Toulmin's claims, evidence is the same as Toulmin's grounds, and argument combines Toulmin's warrant and backing. GSN has a similar genesis.

Bishop, Bloomfield, and Guerra also present the structured approach to safety cases used at Adelard. The paper notes open issues, including:

- The lack of formality and applicable models for assurance argumentation.
- The difficulty in transitioning from the predominant prescriptive approach to the goal-based approach, and the role of standards, both prescriptive and goal-based, in this transition.
- Using assurance case technology for system properties other than safety, e.g., reliability, dependability, security, etc.

Hawkins, et al. [62] introduce *assured safety arguments* to solve the difficulties with a single safety argument. The traditional structure of the safety argument includes the reasons for *confidence* in the claims of the safety argument; Hawkins, et al. decompose the traditional structure into a safety argument and a *confidence argument*. The confidence argument separates out the justification for confidence in the safety argument.

The stated difficulties of the single safety argument are:

- Arguments become large and unwieldy, causing rambling arguments, omission of necessary elements, inclusion of unnecessary material “just in case,” and difficulty reviewing (e.g., for certification).
- Safety and confidence portions of the argument are not differentiated, causing both portions to be poorly prepared, unfocused argumentation, and difficulty building and maintaining the argument.

Hawkins, et al. separate the safety argument from the confidence argument, allowing mitigation of these difficulties. The confidence argument is linked back to the safety argument by assurance claim points (ACP). The paper provides several patterns for construction of confidence arguments. An example of how a confidence argument could be structured is presented; the example is from the safety/confidence arguments for a hypothetical drug infusion pump.

9.4 Dependability Assessment

Bloomfield, Littlewood, and Wright [63] argue that quantitative assessment of confidence in dependability case claims is necessary for proper risk assessment. The authors model judgment of safety integrity levels (SIL); the higher the estimated SIL, the higher assurance a system is required to be. From prior experimental work, a log-normal model is suggested as a good distribution for how experts make probability of failure on demand, or *pdf*, judgments. The log-normal distribution has different mean and mode values for variances greater than zero. The difference between the mean and the mode models the confidence in the judgment, e.g., if most experts judge the system to be SIL2 but are less than 67 percent confident in that judgment, then the mean *pdf* is actually SIL1. This means that a system that is judged to be SIL($n+1$) can be considered to be a SIL(n) system with high confidence. This paper assumes a perfectly trustworthy test oracle and a perfectly representative operational testing profile.

Littlewood and Wright [64] examine the use of multiple diverse argument legs to support dependability claims. By analogy to fault tolerance through diversity, some make the assertion that multiple diverse arguments supporting a claim increase the confidence in the claim. The problem is that there is no theoretical basis to this assertion. In this paper, the authors model the structure of a two-legged argument example using

a six-variable Bayesian belief network (BBN). The example's legs are (1) a testing leg and (2) a verification leg, i.e., formal verification against the system's specification. The results of the model analysis are:

- A diverse argument leg can increase confidence, e.g., reduce doubt by almost two-thirds in one particular example.
- An entirely supportive second leg can reduce the overall confidence when the correlation between specification and oracle correctness is sufficiently high.
- Nonsupportive legs can also substantially decrease confidence.
- For perfection claims (estimation of pfd equal to zero), failure-free testing always constitutes a supportive argument leg, and ideal testing evidence added to a positive verification outcome always improves upon the confidence of the verification leg alone, even if the verification leg is not supportive.

The authors do mention ways that the systems/arguments analogy breaks down: (a) lack of composability and (b) lack of proven efficacy of the approach. They also mention the fact that they do not know the extent to which their counter-intuitive results are simply model artifacts. This work also assumes a perfectly trustworthy test oracle and a perfectly representative operational testing profile.

Bishop, et al. [65] continue this work on confidence by placing their conservative pfd values on a formal footing. The approach is extremely conservative when only a priori beliefs are considered. With evidence of failure-free working, the approach is much less conservative, allowing the expert to treat as true stronger claims of pfd. If the expert has a strong enough prior belief that the system is perfect (completely fault-free), he can treat as true even stronger claims after the same evidence of failure-free working. However, this particular paper only treats testing as evidence in its formalism; formal verification and standards adherence are mentioned but do not make an appearance in the formalism, other than as a possibility for defining prior belief that the system is perfect. As before, this work assumes that the test oracle is totally trustworthy, and the operational testing profile accurately represents real use.

Chapter 10

Conclusion

This work defines the **filter model**, a new framework for researching safety-critical systems *certification*, and evaluates the filter model for feasibility and yield. The fundamental assertion of the filter model is that *the mechanisms that regulators use for critical systems certification are, themselves, critical systems*. Thus, certification mechanisms are amenable to systematic safety engineering.

The work in this thesis supports the following conclusions:

1. The filter model is a feasible model for representing certification mechanisms as systems.
2. Adapting and applying common hazard analysis techniques to certification mechanisms is feasible and yields non-trivial results.
3. Adapting and applying common safety engineering practices to certification mechanisms is feasible.

This thesis makes no claims beyond these conclusions. The analysis presented in this work is the first of its kind, so the conclusions are necessarily presented with significant caveats. Section 10.1 summarizes the results supporting these conclusions, section 10.2 summarizes the limitations of these conclusions, and 10.3 summarizes the avenues of future work.

10.1 Results

The general categories of results that this thesis recognizes are:

- Model feasibility.
- Technique feasibility.
- Technique yield.
- Problem scope.

- Other contributions.

Each of the following subsections briefly summarizes these results.

10.1.1 Model Feasibility

The techniques applied and examined in Chapters 7 and 8 were examined in part to assess the feasibility of the central assumption of the filter model, stated in Chapter 1.

Any given certification mechanism can be viewed as a safety-critical system.

The techniques were all found to be feasible and have non-zero yield. The hazard analysis techniques in particular depended on an *information flow model*; another important result of this thesis is the following, reproduced from Chapter 7:

Analysts can derive an information flow model for any certification mechanism.

Thus, the filter model was found to be a feasible model for representing certification mechanisms as safety-critical systems.

10.1.2 Technique Feasibility

The work in this thesis applied *safety engineering techniques* to an instantiation of the GKG certification mechanism, GKG_{SEPE}. Chapter 7 presented the results of applying hazard analysis techniques – HazOp, FTA, and FMECA – to GKG_{SEPE}, and Chapter 8 presented the results of applying fault treatment – fault elimination and fault tolerance – to GKG_{SEPE}. Each hazard analysis technique was found to be feasible. Generating fault elimination and fault tolerance strategies to apply to GKG_{SEPE} was also found to be feasible. The feasibility of other hazard analysis techniques was not examined; neither was the feasibility of generating fault treatment strategies from fault avoidance or fault forecasting.

10.1.3 Technique Yield

The safety engineering techniques examined in Chapters 7 and 8 were examined for *yield* as well as feasibility. Each hazard analysis technique was found to have a non-zero yield. HazOp and FTA yield were measured in the amount of faults_{cert} discovered. FMECA yield was measured in the organization of faults_{cert} and failure mode discovery.

The generation of fault treatment strategies was also found to have a non-zero yield. At least one strategy was generated for each fault_{cert} that was analyzed.

Analyst Expertise Variance	Fundamental Unknowns
7.5, 7.6, 7.7, 7.8, 7.14, 7.15, 7.18, 7.19, 7.21, 8.5, 8.8, 8.9, 8.10	7.1, 7.2, 7.3, 7.4, 7.9, 7.10, 7.11, 7.12, 7.13, 7.16, 7.17, 7.20, 8.1, 8.2, 8.3, 8.4, 8.6, 8.7, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20

Table 10.1: Categorization of the open questions in this thesis.

10.1.4 Problem Scope

The analysis presented in Chapters 7 and 8 was partially an *exploratory* analysis. Thus, the *open questions* noted in the analysis *are a category of results in themselves*. This thesis produced 41 open questions. Each question was categorized according to whether the question was about (a) variance introduced by analyst expertise or (b) fundamental unknowns. Thirteen questions were about variance and 28 questions were about fundamental unknowns. Table 10.1 summarizes this categorization. These results can be interpreted in three different ways:

- First, the open questions provide a foundation for placing the other results in context. While incomplete, the foundation did not exist before the analysis in this thesis.
- Second, the open questions define the limitations to the results presented in this thesis. Section 10.2 expands on these limitations.
- Third, the open questions present avenues for future work. Answering these questions would expand understanding of the certification process. Section 10.3 expands on avenues for future work.

10.1.5 Other Contributions

Two other results can be gleaned from viewing this thesis holistically:

- **Additional insight into GKG.** GKG was originally intended to provide a definition for the term, “compelling.” The process of (a) creating a partial instantiation of GKG_{SEPE} and (b) analyzing GKG_{SEPE} provided insight into the GKG mechanism itself, culminating in several recommended repairs to the mechanism.
- **Certification analysis framework.** To the author’s knowledge, the filter model is the first framework capable of analyzing and evaluating *any* certification mechanism. The introduction of the framework in this thesis paves the way for comprehensive analysis and comparison of *all* certification mechanisms. Section 10.3.3 elaborates on future filter model analyses, and section 10.3.5 elaborates on future comparison of standards, which is enabled by the existence of the filter model.

10.2 Limitations

The general categories of caveats that this thesis recognizes are:

- Scope.
- Specimen certification mechanism.
- Candidate system.
- Technique application.
- Results metrics.

Each of the following subsections briefly summarizes these caveats.

10.2.1 Scope

Ideally, the claim of this thesis would be that the filter model enables feasible and comprehensive analysis and repair for certification mechanisms in general. Assessing this claim requires controlled experiments, culminating in statistical assessment of the effect of filter analysis and repair on certification mechanisms.

However, such experiments must include comparison to control groups. Every element in a filter model analysis introduces some variance. A comprehensive study requires control groups for every source of variance:

- **Specimen certification mechanism(s).** Certification mechanisms certainly vary across domains, and may vary widely in their ability to filter unfit systems.
- **Certifiers.** The group(s) of certifiers that instantiate and use the specimen certification mechanism(s) may vary widely in their abilities.
- **Candidate system(s).** The characteristics of candidate systems, e.g., complexity, vary across domains, and within domains.
- **Developers.** The group(s) of developers that engineer the candidate system(s) may vary widely in their capabilities.

Additionally, the novelty of this analysis means that the limits of variance-introducing elements is not known.

10.2.2 Specimen Certification Mechanism

This work analyzes GKG as the specimen certification mechanism. GKG was chosen in part because it has never been evaluated, but this means that whether GKG is better or worse than any other certification

mechanism is unknown. In addition, the instantiation of GKG is incomplete; a comprehensive analysis would include a complete instantiation.

10.2.3 Candidate System

This work analyzes DAIS as a candidate system for certification. As DAIS is both a novel type of system and one that was developed in a research context, standard assumptions about development process are not applicable.

10.2.4 Application of Techniques

The hazard analysis techniques used in this work – HazOp, FTA, and FMECA – are well-established in the literature and in the safety-critical systems industry. The same goes for the safety engineering techniques used in this work. However, application of these techniques to a new domain raises questions about the exact methods that analysts should use to apply the techniques. In particular:

- This work makes no claim that the inputs to the techniques are the correct or optimal set of inputs.
- The techniques themselves may not be optimal for filter analysis; there may already exist other techniques that are *more* amenable to adaptation for use in filter analysis. Also, future filter analyses may identify specialized techniques for filter analysis.

10.2.5 Results Metrics

The metrics used to measure how well the filter model analysis and repair worked are only one set of possible metrics. The novelty of the work leaves questions about what the measurements mean, including:

- What are acceptable values/ranges for the metrics?
- What metrics apply to which techniques?
- Are the metrics used in this work necessary or sufficient?
- Are there other metrics that could be used in such an analysis?

10.2.6 Other Limitations

In addition to these limitations, there may be others that are not recognized in this work. The caveats presented in this section place this work in a greater context – examining the characteristics of certification mechanisms. Much of this context is unexplored. As a first foray into this greater context, this work presents the above caveats as areas that warrant future work.

10.3 Future Work

All of the open questions presented throughout the thesis imply future directions for this research. This section summarizes and focuses those questions into research questions in their own right.

Areas of future work include:

- Refining the filter model.
- Refining the process of filter model analysis.
- Filter model analysis of other certification mechanisms.
- Measuring benefits and drawbacks of safety cases.
- Comparison of standards-based certification to goal-based certification.

Each of the following subsections briefly summarizes these areas of future work.

10.3.1 Refining the Filter Model

The filter model makes a number of assumptions. Chief among these assumptions is that certification is a one-time event, occurring at the end of system development and resulting in a binary decision: accept or reject. This assumption, while helpful in this analysis, may be unwarranted; Kelly and others recommend that safety cases be developed during system development, and checked with regulators “early and often” [13]. One obvious avenue for future work is refining the structure of the filter model to include iterative filtration, i.e., multiple attempts to certify a candidate system.

Refining the filter model itself could also prove to be a source for improvement of the *design* of certification mechanisms. Recall from Chapter 7:

***There exist no generally accepted design methods for
certification mechanisms.***

The existence of the filter model enables systematic design of certification mechanisms. As the filter model is novel, such design would be rudimentary. Refining the filter model would improve the certification mechanism design process.

10.3.2 Refining Filter Model Analysis

This work is the first filter model analysis of certification mechanisms; thus, the results are of an exploratory nature. The hazard analyses, safety engineering techniques, and metrics used were largely used because of necessity. This analysis has enabled future work that:

- Studies the effect of variance in how the presented techniques and metrics are used.
- Identifies and studies the use of different techniques and metrics.

10.3.3 Filter Model Analysis of Other Certification Mechanisms

The filter model is designed to be a general model that is applicable to all certification mechanisms. Of particular interest are *extant* certification mechanisms; filter model analysis could prove fruitful in understanding *which* of today's certification mechanisms work, *why* they work, and *how* to fix the mechanisms that do *not* work. Filter model analysis of extant standards, both prescriptive and goal-based, is an area of future work, as is filter model analysis of any new certification mechanisms.

A sub-area of analyzing a wide variety of certification mechanisms is analyzing certification mechanisms *for a particular domain*; standards such as DO-178B [23] were designed to be applicable to the specific domain of "airborne systems." Establishing conclusions about a particular domain's certification mechanisms requires a representative sample of *candidate systems* from that domain. Such a study would examine the certification decisions of multiple certification mechanisms that are supplied the representative sample.

10.3.4 Benefits and Drawbacks of Safety Cases

One common thread in the safety case literature is the assumption that safety cases are generally beneficial to system development. Intuitively, this seems to be true; the very act of creating a safety case structures the thought processes of system developers to think about how a particular system can convincingly be shown to fulfill a safety case. However:

There are no studies that confirm or refute the assumption of the superiority of a goal-based approach in general.

The idea of goal-based certification is, intuitively, a good one, but the benefits of goal-based certification depend in part on the benefits of safety cases. If safety case construction turns out to be extraordinarily expensive for comparatively little benefit, the benefit of goal-based certification will be called into question.

10.3.5 Standards-Based vs. Goal-Based Certification

Lastly, this work enables the possibility of a comparison of prescriptive standards to goal-based standards. The critical systems community is moving to goal-based standards largely because of intuition [6]. However:

There are no studies that show a significant difference, either positive or negative, between standards-based certification and goal-based certification.

A scientific comparison of standards-based certification to goal-based certification would be beneficial in settling the debate about which approach is better. Metrics for this comparison include (but are not limited to) cost to the certifier, cost to the developer, residual risk in certified systems, and rates of unwarranted decisions (both positive and negative).

Bibliography

- [1] Gu Lin. Increasing patient safety by enhancing the environment. Master's thesis, University of Virginia, 2011.
- [2] B.W. Boehm. Software risk management: principles and practices. *Software, IEEE*, 8(1):32–41, 1991.
- [3] Lawrence B Sperry. Automatic pilot for airplanes, April 2 1929. US Patent 1,707,690.
- [4] N.R. Storey. *Safety critical computer systems*. Addison-Wesley Longman Publishing Co., Inc., 1996.
- [5] A. German. Software static code analysis lessons learned. *Crosstalk: Real-Time Software Development Requires Rigid Constraints*, 801:13, 2003.
- [6] Heike Hoppe. Goal-based standards. *WMU Journal of Maritime Affairs*, 4:169–180, 2005.
- [7] J. Penny, A. Eaton, P.G. Bishop, and R.E. Bloomfield. The practicalities of goal-based safety regulation. In Felix Redmill and Tom Anderson, editors, *Aspects of Safety Management*, pages 35–48. Springer London, 2001.
- [8] P. Bishop, R. Bloomfield, and S. Guerra. The future of goal-based assurance cases. In *Proceedings of Workshop on Assurance Cases. Supplemental Volume of the 2004 International Conference on Dependable Systems and Networks*, pages 390–395, 2004.
- [9] UK Ministry of Defence. Safety management requirements for defence systems, 2007.
- [10] Civil Aviation Authority (CAA). Ats safety requirements, 2011.
- [11] T. Kelly and John McDermid. Safety case patterns-reusing successful arguments. In *Understanding Patterns and Their Application to Systems Engineering (Digest No. 1998/308)*, IEE Colloquium on, pages 3/1–3/9, 1998.
- [12] T. Kelly and R. Weaver. The goal structuring notation—a safety argument notation. In *Proc. DSN 2004 Workshop on Assurance Cases*. Citeseer, 2004.
- [13] T. Kelly. A systematic approach to safety case management. In *Proc. of SAE 2004 World Congress, Detroit, MI*. Citeseer, 2004.
- [14] T.P. Kelly et al. Arguing safety—a systematic approach to managing safety cases. *UNIVERSITY OF YORK DEPARTMENT OF COMPUTER SCIENCE-PUBLICATIONS-YCST*, 1999.
- [15] Food and Drug Administration. Total product life cycle: Infusion pump - premarket notification [510(k)] submissions: Draft guidance, 2010.
- [16] B. Damascelli, A. Marchiano, C. Spreafico, R. Lutman, M. Salvetti, MG Bonalumi, M. Mauri, F. Garbagnati, A. Del Nero, G. Comeri, et al. Circadian continuous chemotherapy of renal cell carcinoma with an implantable, programmable infusion pump. *Cancer*, 66(2):237–241, 2006.
- [17] S. Sankaranarayanan, H. Homaei, and C. Lewis. Model-based dependability analysis of programmable drug infusion pumps. *Formal Modeling and Analysis of Timed Systems*, pages 317–334, 2011.

- [18] J. Knight and B. Randell. *Fundamentals of dependable computing for software engineers*. Chapman & Hall, 2012.
- [19] Robert E Melchers. On the alarp approach to risk management. *Reliability Engineering & System Safety*, 71(2):201–208, 2001.
- [20] Title 21 Code of Federal Regulations. Determination of safety and effectiveness, 2010.
- [21] CSR Bev Littlewood and L. Strigini. Validation of ultra-high dependability for software-based systems. *Communications of the ACM (CACM)*, 36(11):69–80, 1993.
- [22] J.C. Knight. An introduction to computing system dependability. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 730–731, 2004.
- [23] RTCA. Do-178b, software considerations in airborne systems and equipment certification.
- [24] NASA. Standard for software assurance, 2005.
- [25] UL. Software in programmable components, 1999.
- [26] SAE. Certification considerations for highly-integrated or complex aircraft systems, 2010.
- [27] R. Chapman, A. Burns, and A. Wellings. Integrated program proof and worst-case timing analysis of spark ada. In *ACM Workshop on language, compiler and tool support for real-time systems*. ACM Press. Citeseer, 1994.
- [28] International Organization for Standards (ISO). Medical devices — application of risk management to medical devices, 2007.
- [29] W.H.H. Chapman Iii, R.J. Albrecht, V.B. Kim, J.A. Young, and W.R. Chitwood Jr. Computer-assisted laparoscopic splenectomy with the da vinci surgical robot. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 12(3):155–159, 2002.
- [30] O.I. Iweala. Hiv diagnostic tests: an overview. *Contraception*, 70(2):141–147, 2004.
- [31] R.D. MacDonald, J.M. Swanson, J.L. Mottley, and C. Weinstein. Performance and error analysis of automated external defibrillator use in the out-of-hospital setting. *Annals of emergency medicine*, 38(3):262–267, 2001.
- [32] Patrick John Graydon. *Assurance Based Development*. PhD thesis, University of Virginia, 2010.
- [33] J. Reason. The contribution of latent human failures to the breakdown of complex systems. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 327(1241):475–484, 1990.
- [34] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. Fault tree analysis, methods, and applications no. 2013; a review. *Reliability, IEEE Transactions on*, R-34(3):194–203, aug. 1985.
- [35] D.H. Stamatis. *Failure mode and effect analysis: FMEA from theory to execution*. Asq Press, 2003.
- [36] F. Redmill, M. Chudleigh, and J. Catmur. *System safety: HAZOP and Software HAZOP*. Wiley, 1999.
- [37] John A McDermid, Mark Nicholson, David J Pumfrey, and P Fenelon. Experience with the application of hazop to computer-based systems. In *Computer Assurance, 1995. COMPASS'95. Systems Integrity, Software Safety and Process Security. Proceedings of the Tenth Annual Conference on*, pages 37–48. IEEE, 1995.
- [38] Trevor A Kletz. Hazop - past and future. *Reliability Engineering & System Safety*, 55(3):263–266, 1997.
- [39] Ada Resource Association. Ada information clearinghouse. <http://www.adaic.org/>, 2013. Accessed: 31/03/2013.
- [40] International Organization for Standards (ISO). Programming languages - c, 2011.

- [41] International Organization for Standards (ISO). Medical devices – quality management systems – requirements for regulatory purposes, 2003.
- [42] PJ Graydon, JC Knight, and M. Green. Certification and safety cases. In *Proceedings of the 28th International System Safety Conference*, Minneapolis, MN, 2010.
- [43] Charles Haddon-Cave et al. *The Nimrod Review: an independent review into the broader issues surrounding the loss of the RAF Nimrod MR2 aircraft XV230 in Afghanistan in 2006*, report, volume 1025. Stationery Office, 2009.
- [44] Tim Kelly. Reviewing assurance arguments-a step-by-step approach. In *Proc. of Workshop on Assurance Cases for Security-The*, 2007.
- [45] Rob Weaver, Georgios Despotou, Tim Kelly, and John McDermid. Combining software evidence: arguments and assurance. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, May 2005.
- [46] J.C. Knight and E. Myers. Phased inspections and their implementation. *ACM SIGSOFT Software Engineering Notes*, 16(3):29–35, 1991.
- [47] W.S Greenwell, J.C. Knight, C.M. Holloway, and J.J Pease. A taxonomy of fallacies in system safety arguments william s. greenwell. *Red Herring*, 1:1, 2006.
- [48] Merriam-Webster. *Merriam-Webster's Collegiate Dictionary, 11th Edition thumb-notched with Win/Mac CD-ROM and Online Subscription*. Merriam-Webster, July 2003.
- [49] International Electrotechnical Commission (IEC). A standard for software reuse in embedded, distributed control systems, 2005.
- [50] National Institute of Standards and Technology. Guidelines for managing and securing mobile devices in the enterprise (draft), 2012.
- [51] World Wide Web Consortium. Soap version 1.2 part 0: Primer, 2003.
- [52] Food and Drug Administration. Off-the-shelf software use in medical devices, 1999.
- [53] International Electrotechnical Commission (IEC). Functional safety of electrical/electronic/programmable electronic safety-related systems, 1997.
- [54] Australian Transport Safety Bureau. In-flight upset event 240km north-west of perth, wa boeing company 777-200, 9m-mrg 1 august 2005., 2007.
- [55] Jacques-Louis Lions et al. Ariane 5 flight 501 failure, 1996.
- [56] U.S. Food and Drug Administration. Manufacturer and user facility device experience. <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/search.cfm>, 2013. Accessed: 17/06/2013.
- [57] Bertrand Meyer. On formalism in specifications. *IEEE software*, 2(1):6–26, 1985.
- [58] J.M. Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–22, 1990.
- [59] R.W. Butler and G.B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *Software Engineering, IEEE Transactions on*, 19(1):3–12, 1993.
- [60] UK Ministry of Defence. Requirements for safety related software in defence equipment, 1997.
- [61] Robert Andrew Weaver. *The safety of software: Constructing and assuring arguments*. PhD thesis, University of York, Department of Computer Science, 2003.
- [62] R. Hawkins, T. Kelly, J. Knight, and P. Graydon. A new approach to creating clear safety arguments. *Advances in Systems Safety*, pages 3–23, 2011.

- [63] R.E. Bloomfield, B. Littlewood, and D. Wright. Confidence: Its role in dependability cases for risk assessment. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 338–346, june 2007.
- [64] W. Littlewood and D. Wright. The use of multilegged arguments to increase confidence in safety claims for software-based systems: a study based on a bbn analysis of an idealized example. *Software Engineering, IEEE Transactions on*, 33(5):347–365, 2007.
- [65] P. Bishop, R. Bloomfield, B. Littlewood, A. Povyakalo, and D. Wright. Toward a formalism for conservative claims about the dependability of software-based systems. *Software Engineering, IEEE Transactions on*, 37(5):708–717, 2011.
- [66] LB Borghouts and HA Keizer. Exercise and insulin sensitivity: a review. *International journal of sports medicine*, 21(01):1–12, 2000.
- [67] Anthony S Fauci et al. *Harrison's principles of internal medicine*, volume 2. McGraw-Hill Medical New York, 2008.
- [68] Richard S Surwit, Mark S Schneider, and Mark N Feinglos. Stress and diabetes mellitus. *Diabetes care*, 15(10):1413–1422, 1992.
- [69] Naomi Weintrob, Hadassa Benzaquen, Avinoam Galatzer, Shlomit Shalitin, Liora Lazar, Gila Fayman, Pearl Lilos, Zvi Dickerman, and Moshe Phillip. Comparison of continuous subcutaneous insulin infusion and multiple daily injection regimens in children with type 1 diabetes: a randomized open crossover trial. *Pediatrics*, 112(3):559–564, 2003.
- [70] U.S. Food and Drug Administration. Recalls, market withdrawals, & safety alerts. <http://www.fda.gov/Safety/Recalls/default.htm>, 2013. Accessed: 17/06/2013.
- [71] Inc. Cerner Multum. Drug interactions checker. http://www.drugs.com/drug_interactions.html, 2013. Accessed: 17/06/2013.
- [72] SELFNutritionData. Selfnutritiondata. <http://nutritiondata.self.com/>, 2013. Accessed: 17/06/2013.
- [73] CyberSoft. Nutribase 11 professional nutrition & fitness software. <http://www.nutribase.com/>, 2013. Accessed: 17/06/2013.
- [74] N. Paul, T. Kohn, and D.C. Klonoff. A review of the security of insulin pump infusion systems. *J Diabetes Sci Technol*, 5(6):1557–62, 2011.
- [75] R.E. Bloomfield, S. Guerra, A. Miller, M. Masera, and C.B. Weinstock. International working group on assurance cases (for security). *Security Privacy, IEEE*, 4(3):66–68, may-june 2006.
- [76] A CISHEC. Guide to hazard and operability studies. *The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd*, 1977.
- [77] Felix Redmill, Morris Chudleigh, and James Catmur. *System safety: HAZOP and Software HAZOP*. Wiley Chichester, UK, 1999.
- [78] J.A. McDermid and D. J. Pumfrey. A development of hazard analysis to aid software design. In *Computer Assurance, 1994. COMPASS '94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on*, pages 17–25, 1994.
- [79] Cem Kaner. Software negligence and testing coverage. *Proceedings of STAR*, 96:313, 1996.
- [80] P. Asterio de C Guerra, C.M.F. Rubira, A. Romanovsky, and Rogerio De Lemos. Integrating cots software components into dependable software architectures. In *Object-Oriented Real-Time Distributed Computing, 2003. Sixth IEEE International Symposium on*, pages 139–142, 2003.

- [81] B. Littlewood and L. Strigini. Software reliability and dependability: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 175–188. ACM, 2000.
- [82] William H Maisel, Michael O Sweeney, William G Stevenson, Kristin E Ellison, and Laurence M Epstein. Recalls and safety alerts involving pacemakers and implantable cardioverter-defibrillator generators. *JAMA: the journal of the American Medical Association*, 286(7):793–799, 2001.
- [83] Y Papadopoulos and J A. McDermid. The potential for a generic approach to certification of safety critical systems in the transportation sector. *Reliability Engineering & System Safety*, 63(1):47 – 66, 1999.
- [84] JL Rouvroye and AC Brombacher. New quantitative safety standards: different techniques, different results? *Reliability engineering & system safety*, 66(2):121–125, 1999.
- [85] DIN. Grundlegende sicherheitsbetrachtungen fur msr-schutzeinrichtungen, 1994.
- [86] U.S. Department of Defense (DoD). Military handbook — reliability prediction of electronic equipment, 1991.
- [87] International Electrotechnical Commission (IEC). Analysis techniques for dependability — reliability block diagram method, 1991.
- [88] William M Goble. *Evaluating Control Systems Reliability-Techniques and Applications*. ISA, 1992.
- [89] J.L. Rouvroye, W.M. Goble, A.C. Brombacher, and R.Th.E. Spiker. A comparison study of qualitative and quantitative analysis techniques for the assessment of safety in industry. In P.Carlo Cacciabue and IoannisA. Papazoglou, editors, *Probabilistic Safety Assessment and Management 96*, pages 559–566. Springer London, 1996.
- [90] O. Sokolsky, I. Lee, and M. Heimdahl. Challenges in the regulatory approval of medical cyber-physical systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 227–232. IEEE, 2011.
- [91] J.A. McDermid. Software safety: where’s the evidence? In *Proceedings of the Sixth Australian workshop on Safety critical systems and software-Volume 3*, pages 1–6. Australian Computer Society, Inc., 2001.
- [92] M.L. Shooman. Avionics software problem occurrence rates. In *Software Reliability Engineering, 1996. Proceedings., Seventh International Symposium on*, pages 55–64, 1996.
- [93] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: Abstraction and reuse of object-oriented design*. Springer, 2001.
- [94] Stephen E Toulmin. *The uses of argument*. Cambridge University Press, 2003.

A GKG HazOp

This appendix lists the questions generated by applying HazOp to GKG_{SEPE}. The parameters used for the HazOp analysis were introduced in Chapter 7 and are reproduced here:

1. Safety case.
2. Fragment.
3. Fragment selector.
4. Phased inspection.
5. Quality.
6. Dialectic.
7. Challenge.
8. Conclusion combiner.

The guide words used are the standard guide words introduced in Chapter 2; the guide words are reproduced here:

1. No or not.
2. More.
3. Less.
4. As well as.
5. Part of.
6. Reverse.
7. Other than.
8. Early.
9. Late.
10. Before.
11. After.

Each entry in the table gives each question a unique question designation (q_1 through q_{88}). Each question is categorized as either irrelevant or relevant. If a question is irrelevant, the entry contains **N/A**. Otherwise, the entry contains **Regular** and is answered: either **Yes** if the question implies a $\text{fault}_{\text{cert}}$ or **No** if the question does not imply a $\text{fault}_{\text{cert}}$.

Q#	Parameter	Guide Word	Meaning	Question	Category	Fault?
q ₁	Safety Case	NO OR NOT	Complete negation of the design intent	What if the submitted safety case does not address risk?	N/A	N/A
q ₂		MORE	Quantitative increase	What if the submitted safety case over-mitigates risk?	Regular	No
q ₃		LESS	Quantitative decrease	What if the submitted safety case under-mitigates risk?	Regular	No
q ₄		AS WELL AS	Qualitative modification/ increase	What if the submitted safety case needs more relevant parts?	Regular	No
q ₅		PART OF	Qualitative modification/ decrease	What if the submitted safety case has only some relevant parts?	Regular	No
q ₆		REVERSE	Logical opposite of the design intent	What if the submitted safety case argues for risk?	N/A	N/A
q ₇		OTHER THAN	Complete substitution	What if the submitted safety case is corrupted?	Regular	Yes
q ₈		EARLY	Relative to the clock time	What if the submitted safety case is filtered too early?	N/A	N/A
q ₉		LATE	Relative to the clock time	What if the submitted safety case is filtered too late?	N/A	N/A
q ₁₀		BEFORE	Relating to order or sequence	What if the submitted safety case is filtered before it is complete?	N/A	N/A
q ₁₁		AFTER	Relating to order or sequence	What if the submitted safety case is filtered after it is complete?	Regular	No
q ₁₂	Fragment	NO OR NOT	Complete negation of the design intent	What if the current fragment does not support any claims?	Regular	No
q ₁₃		MORE	Quantitative increase	What if the current fragment supports more than a single sub-claim?	Regular	No
q ₁₄		LESS	Quantitative decrease	What if the current fragment supports less than a single sub-claim?	N/A	N/A

Q ₁₅		AS WELL AS	Qualitative modification/ increase	What if the current fragment supports part of another sub-claim?	N/A	N/A
Q ₁₆		PART OF	Qualitative modification/ decrease	What if the current fragment is only part of the support for a single sub-claim?	N/A	N/A
Q ₁₇		REVERSE	Logical opposite of the design intent	What if the current fragment does not support a single sub-claim (!= 1)?	N/A	N/A
Q ₁₈		OTHER THAN	Complete substitution	What if the current fragment is corrupted?	Regular	Yes
Q ₁₉		EARLY	Relative to the clock time	What if the current fragment is examined too early?	N/A	N/A
Q ₂₀		LATE	Relative to the clock time	What if the current fragment is examined too late?	N/A	N/A
Q ₂₁		BEFORE	Relating to order or sequence	What if the current fragment is examined before one that it depends on?	Regular	Yes
Q ₂₂		AFTER	Relating to order or sequence	What if the current fragment is examined after one it depends on?	Regular	Yes
Q ₂₃	Quality	NO OR NOT	Complete negation of the design intent	What if the current quality is unnecessary for risk reduction?	Regular	No
Q ₂₄		MORE	Quantitative increase	What if the current quality is established as greater than needed?	Regular	No
Q ₂₅		LESS	Quantitative decrease	What if the current quality is established as less than needed?	Regular	No
Q ₂₆		AS WELL AS	Qualitative modification/ increase	What if the current quality is too stringent?	Regular	No
Q ₂₇		PART OF	Qualitative modification/ decrease	What if the current quality is not stringent enough?	Regular	Yes
Q ₂₈		REVERSE	Logical opposite of the design intent	What if the current quality is antithetical to reduction of risk?	N/A	N/A

Q ₂₉		OTHER THAN	Complete substitution	What if the current quality is corrupted?	Regular	Yes
Q ₃₀		EARLY	Relative to the clock time	What if the current quality is examined for too early?	N/A	N/A
Q ₃₁		LATE	Relative to the clock time	What if the current quality is examined for too late?	N/A	N/A
Q ₃₂		BEFORE	Relating to order or sequence	What if the current quality is examined for before one that it depends on?	N/A	N/A
Q ₃₃		AFTER	Relating to order or sequence	What if the current quality is examined for after one that it depends on?	N/A	N/A
Q ₃₄	Challenge	NO OR NOT	Complete negation of the design intent	What if the current challenge does not accurately represent the certifier's definition of reasonable?	N/A	N/A
Q ₃₅		MORE	Quantitative increase	What if the current challenge exceeds the certifier's definition of reasonable?	N/A	N/A
Q ₃₆		LESS	Quantitative decrease	What if the current challenge does not meet the certifier's definition of reasonable?	N/A	N/A
Q ₃₇		AS WELL AS	Qualitative modification/ increase	What if the current challenge is composed of other challenges?	N/A	N/A
Q ₃₈		PART OF	Qualitative modification/ decrease	What if the current challenge is part of another challenge?	N/A	N/A
Q ₃₉		REVERSE	Logical opposite of the design intent	What if the current challenge is totally unreasonable?	N/A	N/A
Q ₄₀		OTHER THAN	Complete substitution	What if the current challenge is corrupted?	Regular	Yes.
Q ₄₁		EARLY	Relative to the clock time	What if the current challenge is issued too early?	N/A	N/A

Q ₄₂		LATE	Relative to the clock time	What if the current challenge is issued too late?	N/A	N/A
Q ₄₃		BEFORE	Relating to order or sequence	What if the current challenge is issued before one that it depends on?	N/A	N/A
Q ₄₄		AFTER	Relating to order or sequence	What if the current challenge is issued after one that it depends on?	N/A	N/A
Q ₄₅	Fragment Selector	NO OR NOT	Complete negation of the design intent	What if the fragment selector does not select related argument elements?	N/A	N/A
Q ₄₆		MORE	Quantitative increase	What if the fragment selector selects too many argument elements?	N/A	N/A
Q ₄₇		LESS	Quantitative decrease	What if the fragment selector selects too few argument elements?	N/A	N/A
Q ₄₈		AS WELL AS	Qualitative modification/ increase	What if the fragment selector selects more argument elements than it should?	N/A	N/A
Q ₄₉		PART OF	Qualitative modification/ decrease	What if the fragment selector selects fewer argument elements than it should?	N/A	N/A
Q ₅₀		REVERSE	Logical opposite of the design intent	What if the fragment selector selects totally unrelated argument elements?	N/A	N/A
Q ₅₁		OTHER THAN	Complete substitution	What if the fragment selector algorithm is corrupted?	N/A	N/A
Q ₅₂		EARLY	Relative to the clock time	What if the fragment selector selects a fragment too early?	N/A	N/A
Q ₅₃		LATE	Relative to the clock time	What if the fragment selector selects a fragment too late?	N/A	N/A

q ₅₄		BEFORE	Relating to order or sequence	What if the fragment selector selects a fragment before one that it depends on?	N/A	N/A
q ₅₅		AFTER	Relating to order or sequence	What if the fragment selector selects a fragment after one that it depends on?	Regular	No
q ₅₆	Phased Inspection	NO OR NOT	Complete negation of the design intent	What if the phased inspection is not able to establish any qualities?	Regular	No
q ₅₇		MORE	Quantitative increase	What if the phased inspection establishes more qualities than intended?	Regular	No
q ₅₈		LESS	Quantitative decrease	What if the phased inspection establishes fewer qualities than intended?	Regular	No
q ₅₉		AS WELL AS	Qualitative modification/ increase	What if the phased inspection establishes qualities more strongly than required?	Regular	No
q ₆₀		PART OF	Qualitative modification/ decrease	What if the phased inspection establishes qualities less strongly than required?	Regular	No
q ₆₁		REVERSE	Logical opposite of the design intent	What if the phased inspection establishes the opposite of all the qualities?	Regular	No
q ₆₂		OTHER THAN	Complete substitution	What if the phased inspection is not conducted correctly?	Regular	Yes
q ₆₃		EARLY	Relative to the clock time	What if the phased inspection is conducted too early?	N/A	N/A
q ₆₄		LATE	Relative to the clock time	What if the phased inspection is conducted too late?	N/A	N/A
q ₆₅		BEFORE	Relating to order or sequence	What if the phased inspection is conducted before the dialectic?	Regular	No

Q ₆₆		AFTER	Relating to order or sequence	What if the phased inspection is conducted after the dialectic?	N/A	N/A
Q ₆₇	Dialectic	NO OR NOT	Complete negation of the design intent	What if the dialectic does not end?	N/A	N/A
Q ₆₈		MORE	Quantitative increase	What if the dialectic issues more challenges than necessary?	Regular	No
Q ₆₉		LESS	Quantitative decrease	What if the dialectic issues fewer challenges than necessary?	Regular	Yes
Q ₇₀		AS WELL AS	Qualitative modification/increase	What if the dialectic results in more agreement than necessary?	N/A	N/A
Q ₇₁		PART OF	Qualitative modification/decrease	What if the dialectic results in less agreement than necessary?	Regular	Yes
Q ₇₂		REVERSE	Logical opposite of the design intent	What if the dialectic does not challenge the fragment?	Regular	Yes
Q ₇₃		OTHER THAN	Complete substitution	What if the dialectic is corrupted?	Regular	Yes
Q ₇₄		EARLY	Relative to the clock time	What if the dialectic is conducted too early?	N/A	N/A
Q ₇₅		LATE	Relative to the clock time	What if the dialectic is conducted too late?	N/A	N/A
Q ₇₆		BEFORE	Relating to order or sequence	What if the dialectic is conducted before the phased inspection?	N/A	N/A
Q ₇₇		AFTER	Relating to order or sequence	What if the dialectic is conducted after the phased inspection?	N/A	N/A
Q ₇₈	Conclusion Combiner	NO OR NOT	Complete negation of the design intent	What if the conclusion combiner cannot combine the conclusions?	N/A	N/A
Q ₇₉		MORE	Quantitative increase	What if the conclusion combiner combines too many conclusions?	N/A	N/A
Q ₈₀		LESS	Quantitative decrease	What if the conclusion combiner combines too few conclusions?	N/A	N/A

Q ₈₁		AS WELL AS	Qualitative modification/increase	What if the conclusion combiner overstates the conclusions?	Regular	Yes
Q ₈₂		PART OF	Qualitative modification/decrease	What if the conclusion combiner understates the conclusions?	Regular	No
Q ₈₃		REVERSE	Logical opposite of the design intent	What if the conclusion combiner cannot combine the conclusions correctly?	Regular	Yes
Q ₈₄		OTHER THAN	Complete substitution	What if the conclusion combiner corrupts conclusions?	Regular	Yes
Q ₈₅		EARLY	Relative to the clock time	What if the conclusion combiner is used too early?	N/A	N/A
Q ₈₆		LATE	Relative to the clock time	What if the conclusion combiner is used too late?	N/A	N/A
Q ₈₇		BEFORE	Relating to order or sequence	What if the conclusion combiner is used before the input fragments are inspected/dialected?	N/A	N/A
Q ₈₈		AFTER	Relating to order or sequence	What if the conclusion combiner is used after the input fragments are inspected/dialected?	N/A	N/A

B GKG Fault Tree Analysis

This appendix presents the fault tree generated by applying FTA to GKG_{SEPE} . The fault tree is presented in textual form; the general format is below:

- [BOOLEAN_OPERATOR] [EVENT]
 - [CONTRIBUTING_EVENT_1]
 - [CONTRIBUTING_EVENT_2]
 - ...
 - [CONTRIBUTING_EVENT_N]

[BOOLEAN_OPERATOR] represents the boolean operator that operates on [CONTRIBUTING_EVENT_1] through [CONTRIBUTING_EVENT_N], which are contributing events to [EVENT]. The most-indented events are basic; the rest are compound.

FAULT TREE

Where C = Certifier, A = Application, **OR**, **AND**, etc. = FTA gates. (Text) = placeholder

- GKG allows certification of a product with unacceptable residual risk

OR

- **OR** (Phased inspection fails to identify “flaw”)
 - Phase order allows fault to slip by
 - **AND**
 - Quality X established incorrectly (i.e., it is established, but is not actually present)
 - Quality X is a critical quality
 - **OR** (Phased inspection tool malfunction)
 - **AND**
 - Tool corrupts inspection data
 - Inspector fails to recognize corruption
 - Tool omits inspection phase
 - Other tool malfunction occurs
- **OR** (Quality omitted from qualities list)
 - **AND**
 - Quality X considered not critical
 - Lack of quality X contributes to unacceptable residual risk
 - Quality X accidentally omitted in transcription from source (e.g., from GKG paper)
- **OR** (Quality stated incorrectly in qualities list)
 - Quality transcribed incorrectly from source (e.g., from paper)
 - Quality derived incorrectly (e.g., misunderstanding of literature)
- **OR** (Challenge omitted from challenge list)
 - Analyzed failure data incomplete
 - Analyzed failure data incorrect (e.g., numbers too high/low to merit inclusion of a challenge)
 - Failure data analyzed incorrectly
 - (Note: both predominantly human and mechanical (tool-based) analysis have the same types of events)

- **OR** (Challenge in challenge list stated incorrectly)
 - **AND** (Tool-related)
 - Challenge derived using tool
 - **OR**
 - Tool provided with incorrect information
 - **AND**
 - Tool provided with correct information
 - Tool derived challenge incorrectly
 - **AND** (User-related)
 - Challenge derived from inspection of failure data
 - **OR**
 - Analyzed data irrelevant to current system domain
 - Analyst derives challenge incorrectly
- **OR** (Dialectic leads to unwarranted agreement)
 - **AND** (Parties agree on incorrect expected practice)
 - C and A agree that expected practice is X
 - Expected practice is actually $Y \neq X$
 - **AND** (Expected practice omitted by agreement)
 - C and A agree that all expected practices adhered to
 - An expected practice was omitted
 - **AND** (Subarguments have common dependence fault)
 - Independent subarguments used
 - C and A agree that subarguments are independent
 - Subarguments have a common dependence
 - **AND** (Negative experience dealt with, but not sufficiently/incorrectly)
 - C and A agree that the argument has dealt with the accumulation of negative experience
 - **OR**
 - Negative experience X was not dealt with.
 - Negative experience dealt with incorrectly
 - (Note what are the negative experiences)
 - **AND** (*Unrealistic assumptions*)
 - C & A agree that all assumptions made are realistic

- Assumption X is unrealistic
- **AND** (*Inapplicability*)
 - C & A agree that all circumstances that the safety argument depends on are applicable
 - Circumstance X is not applicable
 - (Note What are the circumstances)
- **AND** (*Improper use of patterns*)
 - C & A agree that all patterns are used correctly
 - Pattern X is used incorrectly
- **AND** (*Inadequate argument strength*)
 - C & A agree that the argument fragments all have adequate strength
 - **OR** (Fragment X is inadequate)
 - Fragment X is on the inadequate fragments list
 - Fragment X is from a system that was inadequate in given circumstances
 - **AND** (Fragment passes through the review process but is inadequate)
 - Fragment X passes local review process
 - Likely consequences of failure underestimated
- **AND** (Break down to fragments results in some failure)
 - **OR** (Break down changes meaning of argument)
 - Break down removes necessary context element from a particular fragment
 - Break down changes logical flow of argument
 - Meaning is changed such that it affects safety

C GKG Failure Modes, Effects, and Criticality Analysis

This appendix presents the results of the FMECA applied to GKG_{SEPE}. The results are presented in tabular form; each entry contains:

- The fault designation (cf. Chapter 7).
- A failure mode for that fault designation.
- Possible effects of that failure mode.
- A short statement explaining how the effects can become critical.

The entries are grouped by fault designation, i.e., if a fault designation has multiple failure modes, they are presented in succession.

Fault Designation	Failure Mode	Possible Effects	Failure is Critical When:
$F_{cert-qual.est.inc}$	Quality established incorrectly (i.e., it is established, but is not actually present).	Arguments exhibiting the fallacy associated with that quality will all pass through.	The argument exhibits the fallacy associated with the quality.
$F_{cert-qual.omi}$	Quality omitted from qualities list.	Argument not inspected for that quality.	The argument exhibits the fallacy associated with the quality.
$F_{cert-qual.oth}$	Quality transcribed incorrectly from source (e.g., from paper)	Quality morphs into another quality, which may or may not be necessary to inspect for. Either way, the argument is not inspected for the original quality.	The argument exhibits the fallacy associated with the quality.
	Quality derived incorrectly (e.g., misunderstanding of literature).	(same as above)	(same as above)
	Quality corrupted by some other means.	(same as above)	(same as above)

Fault Designation	Failure Mode	Possible Effects	Failure is Critical When:
$F_{cert-chal.oth}$	Failure data analysis for a challenge used incomplete failure data.	Challenge does not accurately reflect accumulation of negative experience.	The dialectic using the challenge results in an unwarranted agreement that negative experience has been properly mitigated.
	Failure data analysis for a challenge used incorrect failure data.	(same as above)	(same as above)
	Failure data analysis for a challenge conducted incorrectly.	(same as above)	(same as above)
$F_{cert-chal.omi}$	Challenge in challenge list omitted.	The dialectic fails to issue a critical challenge.	The dialectic using the challenge results in an unwarranted agreement that negative experience has been properly mitigated.
$F_{cert-dial.le}$	The dialectic issues fewer challenges than necessary.	The dialectic fails to issue critical challenges.	The dialectic results in unwarranted agreement because the certifier issues too few challenges.
$F_{cert-dial.oth}$	The dialectic is corrupted in some other way.	The dialectic fails to issue challenges that, if issued, would uncover faults _{cand} .	The dialectic results in unwarranted agreement.

Fault Designation	Failure Mode	Possible Effects	Failure is Critical When:
$F_{cert-dial.rev}$	Dialectic leads to unwarranted agreement regarding an expected practice, X.	Whatever assurance EPX should have provided is gone. The developer actually applied EPX'. Any further assurance detriment is dependent on what EPX' is. At the very least, any part of the argument that depends on EPX is invalid.	EPX' does not mitigate the fault that EPX is designed for, and if the candidate system exhibits the fault that EPX is supposed to mitigate, then the candidate system is an accident _{cert} if other parts of the filter do not stop it from getting through.
	Unwarranted agreement - Omitted expected practice X	Whatever assurance that expected practice should have provided is gone. A actually did EPX' = null.	(same as above)
	Unwarranted agreement - Sub-arguments have common dependence fault	Actual assurance of fragment that depends on those sub-arguments is lower than estimated.	The actual assurance is too low: below ALARP region.
	Unwarranted agreement - Unrealistic assumptions	Fragments dependent on those assumptions are not sufficiently strong, despite being believed to be.	The actual assurance is below ALARP region.

Fault Designation	Failure Mode	Possible Effects	Failure is Critical When:
$F_{cert-dial.rev}$	Unwarranted agreement - Inapplicable circumstances	Fragments dependent on those circumstances are not sufficiently strong, despite being believed to be.	The actual assurance is below ALARP region.
	Unwarranted agreement - Improper use of patterns	Actual assurance of argument is probably lower than what it is believed to be.	The actual assurance is below ALARP region.
	Unwarranted agreement - Inadequate argument strength	Actual assurance of argument is probably lower than what it is believed to be.	The actual assurance is below ALARP region.
	Unwarranted agreement in some other way.	Actual assurance of argument is probably lower than what it is believed to be.	The actual assurance is below ALARP region.

D DAIS HazOp

This appendix presents the results of the HazOp analysis on DAIS. The parameters used were the three major system modules, as defined in Lin [1]:

1. Display Module, abbreviated as DM.
2. Data Analysis Module, abbreviated as DAnM.
3. Data Archive Module, abbreviated as DArM.

The guide words used were the standard guide words (cf. Chapter 3 and Appendix A):

1. No or not.
2. More.
3. Less.
4. As well as.
5. Part of.
6. Reverse.
7. Other than.
8. Early.
9. Late.
10. Before.
11. After.

Each entry in the table contains one question and its answer. If a question is irrelevant, the entry contains N/A. Otherwise, the entry contains the answer to the question. The answers to the question were used for the FTA for DAIS, presented in Appendix E.

Parameter	Guide Word	Meaning	Question	Answer
Display Module	NO OR NOT	Complete negation of the design intent	What if the Display Module displays nothing?	DM failure.
	MORE	Quantitative increase	What if the Display Module displays more information than necessary?	N/A.
	LESS	Quantitative decrease	What if the Display Module displays less information than necessary?	DM failure.
	AS WELL AS	Qualitative modification/increase	What if the Display Module displays more information than necessary?	Could lead to information overload.
	PART OF	Qualitative modification/decrease	What if the Display Module displays less information than necessary?	DM failure.
	REVERSE	Logical opposite of the design intent	What if the Display Module displays incorrect information?	DM failure.
	OTHER THAN	Complete substitution	What if the Display Module displays something other than the requested information?	DM failure.
	EARLY	Relative to the clock time	What if the Display Module displays requested information early?	That's OK.
	LATE	Relative to the clock time	What if the Display Module displays requested information late?	Could lead to information overload.

	BEFORE	Relating to order or sequence	What if the Display Module displays requested information before it should?	N/A. Information return should be as fast as possible.
	AFTER	Relating to order or sequence	What if the Display Module displays requested information after it should?	DM failure.
Data Analysis Module	NO OR NOT	Complete negation of the design intent	What if the Data Analysis Module does not analyze data correctly?	DAnM failure.
	MORE	Quantitative increase	What if the Data Analysis Module does more analysis than necessary?	N/A.
	LESS	Quantitative decrease	What if the Data Analysis Module does less analysis than necessary?	DAnM failure.
	AS WELL AS	Qualitative modification/increase	What if the Data Analysis Module does more analysis than necessary?	N/A.
	PART OF	Qualitative modification/decrease	What if the Data Analysis Module does less analysis than necessary?	DAnM failure.
	REVERSE	Logical opposite of the design intent	What if the Data Analysis Module analyzes data incorrectly?	DAnM failure.
	OTHER THAN	Complete substitution	What if the Data Analysis Module analyzes other data?	DAnM failure.
	EARLY	Relative to the clock time	What if the Data Analysis Module analyzes data too early?	N/A.

	LATE	Relative to the clock time	What if the Data Analysis Module analyzes data too late?	DAnM failure.
	BEFORE	Relating to order or sequence	What if the Data Analysis Module analyzes data before it should?	N/A.
	AFTER	Relating to order or sequence	What if the Data Analysis Module analyzes data after it should?	DAnM failure.
Data Archive Module	NO OR NOT	Complete negation of the design intent	What if the Data Archive Module does not archive data?	DArM failure.
	MORE	Quantitative increase	What if the Data Archive Module archives more data than necessary?	Could lead to storage being full. Highly unlikely.
	LESS	Quantitative decrease	What if the Data Archive Module archives less data than necessary?	DArM failure.
	AS WELL AS	Qualitative modification/increase	What if the Data Archive Module archives more data than necessary?	Could lead to storage being full. Highly unlikely.
	PART OF	Qualitative modification/decrease	What if the Data Archive Module archives less data than necessary?	DArM failure.
	REVERSE	Logical opposite of the design intent	What if the Data Archive Module does not archive the necessary data?	DArM failure.
	OTHER THAN	Complete substitution	What if the Data Archive Module archives other data than it should?	DArM failure.

	EARLY	Relative to the clock time	What if the Data Archive Module archives data too early?	Could lead to synchronization problem. DArM failure.
	LATE	Relative to the clock time	What if the Data Archive Module archives data too late?	Could lead to synchronization problem. DArM failure.
	BEFORE	Relating to order or sequence	What if the Data Archive Module archives data before it should?	Could lead to synchronization problem. DArM failure.
	AFTER	Relating to order or sequence	What if the Data Archive Module archives data after it should?	Could lead to synchronization problem. DArM failure.

E DAIS Fault Tree Analysis

This appendix presents the fault tree generated by applying FTA to DAIS. The fault tree is presented in textual form; the general format is the same as in Appendix B and is reproduced below:

- [BOOLEAN_OPERATOR] [EVENT]
 - [CONTRIBUTING_EVENT_1]
 - [CONTRIBUTING_EVENT_2]
 - ...
 - [CONTRIBUTING_EVENT_N]

[BOOLEAN_OPERATOR] represents the boolean operator that operates on [CONTRIBUTING_EVENT_1] through [CONTRIBUTING_EVENT_N], which are contributing events to [EVENT]. The most-indented events are basic; the rest are compound.

FAULT TREE

Where **OR**, **AND**, etc. = FTA gates. Text = placeholder

- **OR** DAIS causes patient mistreatment
 - **OR** DAIS provides misinformation to the patient
 - Data Analysis Module sends incorrect remote alarm signal
 - **OR** Interface Module displays incorrect information
 - Interface Module fails
 - **OR** Data Analysis Module generates incorrect information
 - **OR** External sources present incorrect information
 - Device presents incorrect data
 - FDA Safety Alert presents incorrect information
 - Sensing data is wrong
 - **OR** Data Archive Module presents incorrect information
 - Data Archive Module transports data incorrectly
 - Database gets corrupted
 - **AND** Database has incorrect information
 1. Data Analysis Module verification fails
 2. Patient enters data incorrectly
 - **OR** Data Analysis Module calculates incorrectly
 - Software Failure
 - **AND**
 1. Database information used in calculation
 2. **AND** Database has incorrect information
 1. Data Analysis Module verification fails
 2. Patient enters data incorrectly

- **OR** DAIS does not provide necessary information to the patient (lack of information)
 - **OR** Interface Module fails to display relevant information
 - **OR** Interface Module fails to receive relevant information
 - **OR** Link between Data Analysis and Interface Modules fails
 - **OR** Data Analysis Module fails to send relevant information
 - Data Archive Module does not supply relevant information
 - Data Analysis Module fails to create relevant information
 - **OR** Data Analysis Module cannot procure relevant information
 - **OR** Relevant remote alarm signal (from Data Analysis Module) not received
 - Data Analysis module fails to send relevant remote alarm signal
 - Link between patient device and Data Analysis Module fails

F DAIS Safety Case

This appendix presents the safety case constructed for the high assurance version of DAIS (cf. Chapter 6). The contexts, `C.FormalSpec` and `C.TestPlan` apply to all of the goals directly above leaf (solution) level, e.g., `G_LackOfInformation.1..`

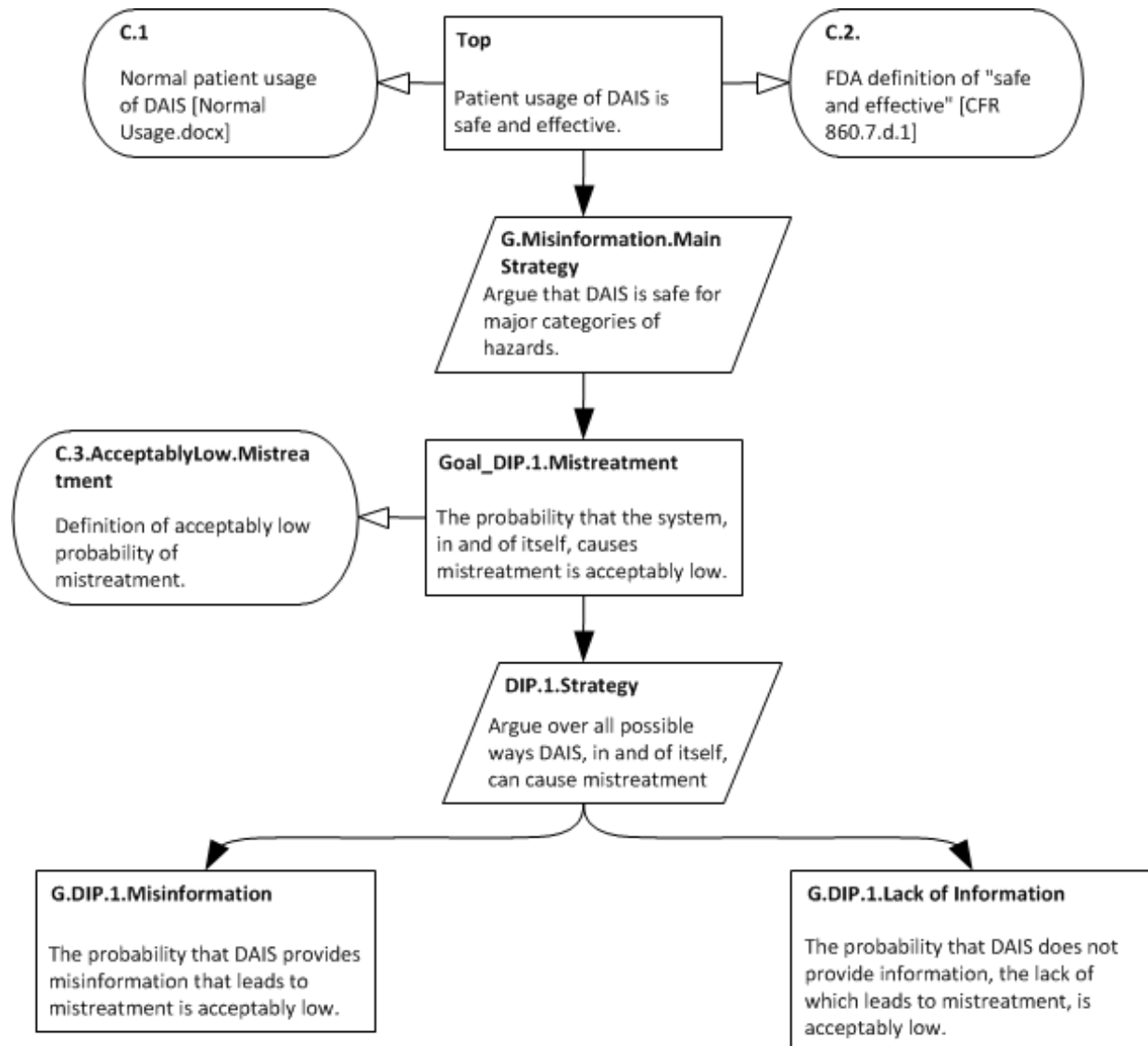


Figure 1: The top of the DAIS safety case.

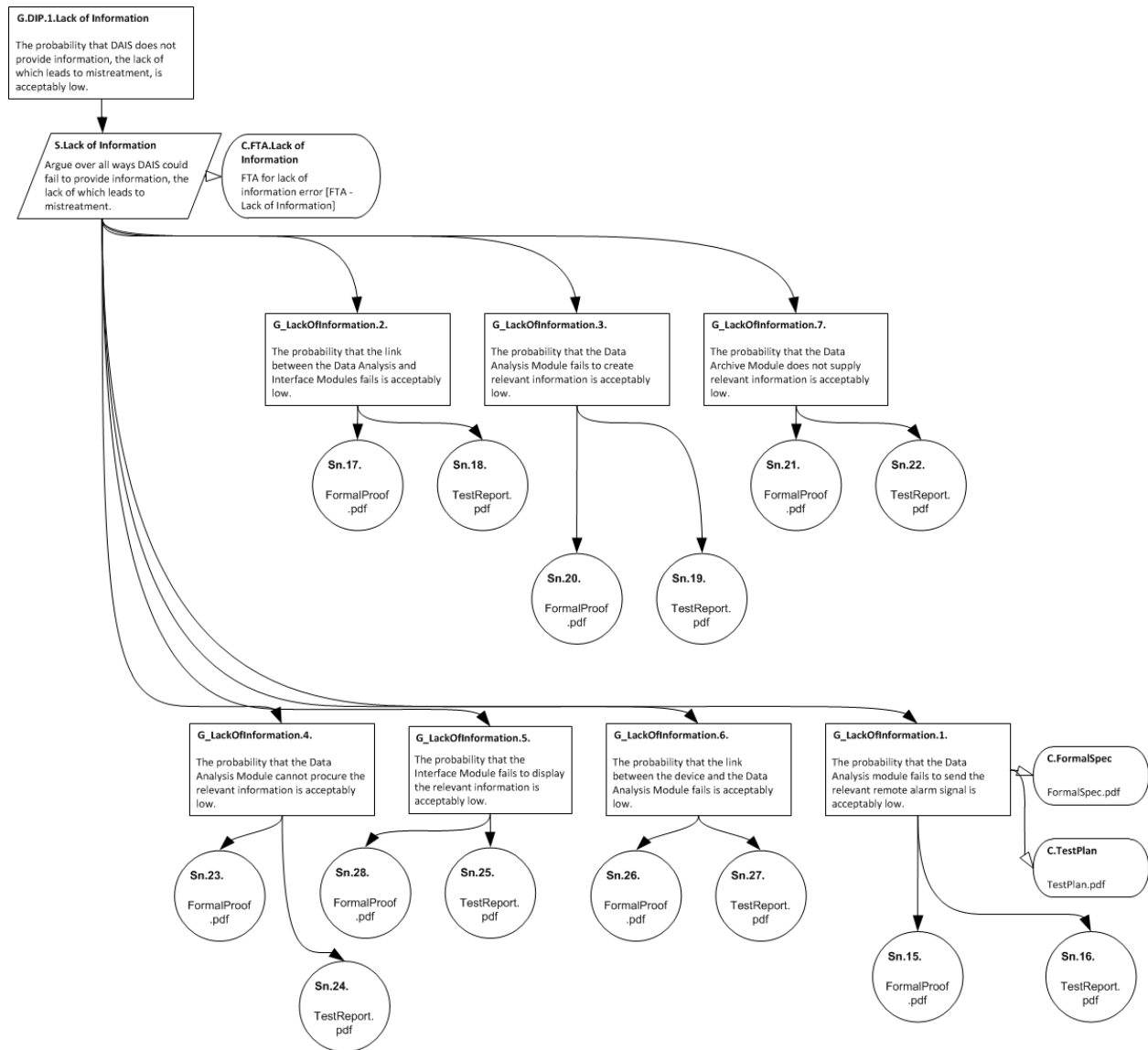


Figure 2: DAIS safety case portion starting with G.DIP.1.Misinformation.

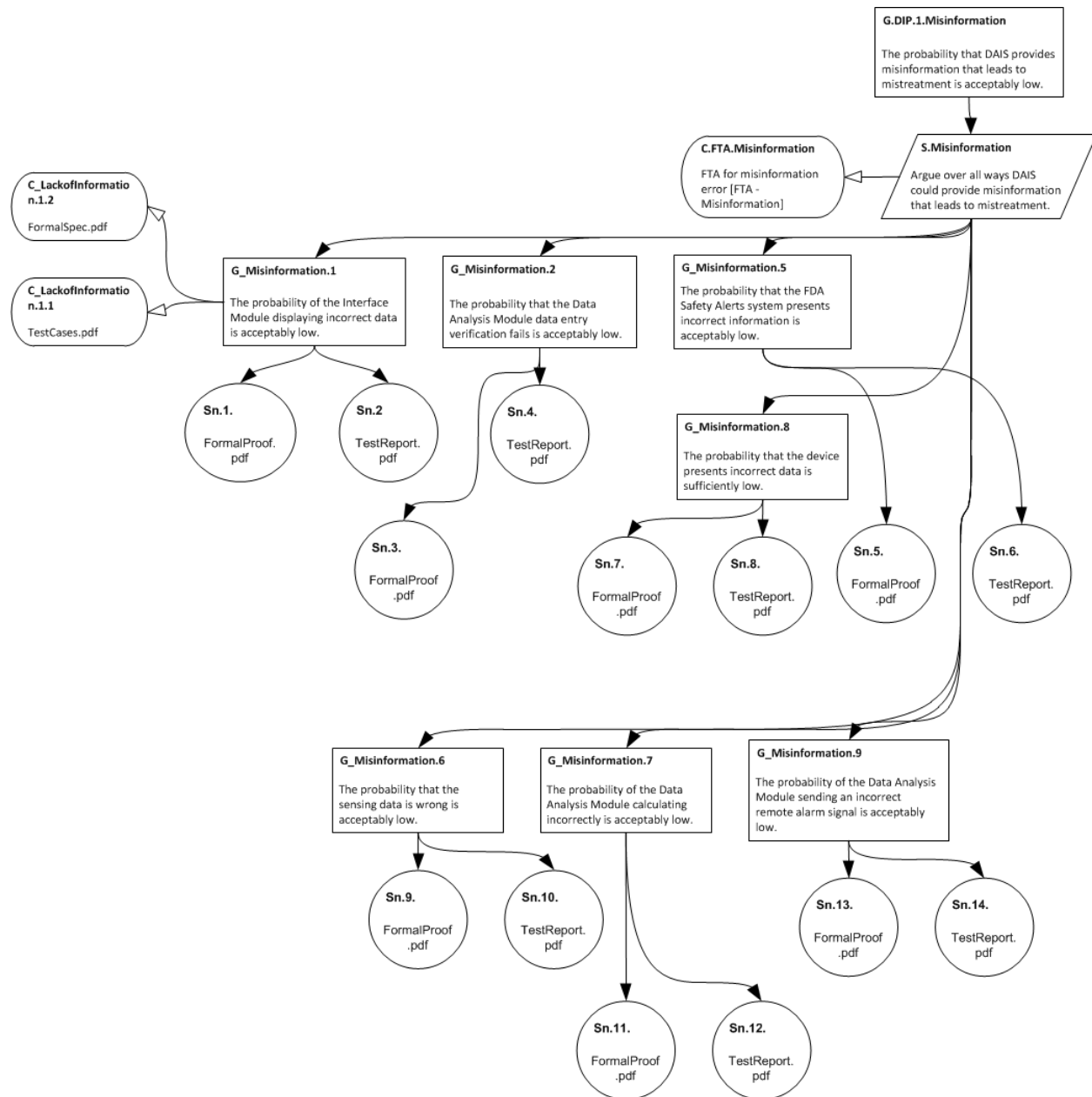


Figure 3: DAIS safety case portion starting with G.DIP.1.Lack of Information.