Tutorial Overview

• Part One: 1300-1515
  • Introduction to Safety Cases
  • The Importance of Safety Arguments
  • The Goal Structuring Notation (GSN)
• Part Two: 1530-1745
  • Software Safety Cases
  • Problems with Current Approaches
  • The Structure of a Typical Software Safety Argument
  • Establishing Hazard-Directed Software Safety Arguments
  • Linking Process and Product Arguments
Part 1: Overview

- Safety Case concept and purpose
- Requirements from standards
- Safety Case contents
- Safety arguments
  - presenting clear arguments
  - Goal Structuring Notation (GSN)
- Creating Arguments in GSN
- Where, When and How to Create Assurance Arguments

Motivation

- Many (UK) standards establish the need for production of a safety case, e.g.
  
  “Safety Cases are required for all new ships and equipment as a means of formally documenting the adequate control of Risk and demonstrating that levels of risk achieved are As Low As Reasonably Practicable (ALARP).” (JSP430)

  A person in control of any railway infrastructure shall not use or permit it to be used for the operation of trains unless
  (a) he has prepared a safety case …
  (b) the Executive has accepted that safety case …”
  (HSE Railway Safety Case Regulations)

  “The Software Design Authority shall provide a Software Safety Case …”
  (U.K. Defence Standard 00-55)
The Purpose of a Safety Case

**Principal Objective:**
- Safety case presents the argument that a system will be acceptably safe in a given context
- ‘System’ could be ...
  - physical (e.g. aero-engines, reactor protection systems)
  - procedural (e.g. railway operations, off-shore)
- Safety Cases can be prepared for ..
  - commissioning
  - maintenance
  - decommissioning ...

Some Definitions

- "A safety case is a comprehensive and structured set of safety documentation which is aimed to ensure that the safety of a specific vessel or equipment can be demonstrated by reference to:
  - safety arrangements and organisation
  - safety analyses
  - compliance with the standards and best practice
  - acceptance tests
  - audits
  - inspections
  - feedback
  - provision made for safe use including emergency arrangements” (JSP 430)
- "The software safety case shall present a well-organised and reasoned justification based on objective evidence, that the software does or will satisfy the safety aspects of the Statement of Technical Requirements and the Software Requirements specification.” (DS 00-55)
Argument & Evidence

A safety case requires two elements:

- **Supporting Evidence**
  Results of observing, analysing, testing, simulating and estimating the properties of a system that provide the fundamental information from which safety can be inferred.

- **High Level Argument**
  Explanation of how the available evidence can be reasonably interpreted as indicating acceptable safety – usually by demonstrating compliance with requirements, sufficient mitigation / avoidance of hazards etc.

- Argument without Evidence is unfounded
- Evidence without Argument is unexplained
Safety Cases vs. Safety Case Reports

- **Safety Case** is the totality of the safety justification + all the supporting material: testing reports, validation reports, relevant design information etc.

- **Safety Case Report** is the document that summarises all the key components of the Safety Case and references all supporting documentation in a clear and concise format.

Safety Case Reports

- Exact contents depends on regulatory environment
- The following are key elements of most standards:
  - Scope
  - System Description
  - System Hazards
  - Safety Requirements
  - Risk Assessment
  - Hazard Control / Risk Reduction Measures
  - Safety Analysis / Test
  - Safety Management System
  - Development Process Justification
  - Conclusions
The Safety Case is not just a collection of disparate pieces of information.

The Safety Argument should form the ‘spine’ of the Safety Case showing how these elements are related and combined to provide assurance of safety.

- within the limits defined [Scope], the system [System Description] is SAFE because all identified hazards [System Hazards] and requirements [Safety Requirements] have been addressed. Hazards have been sufficiently controlled and mitigated [Hazard Control / Risk Reduction Measures] according to the safety risk posed [Risk Assessment]. Evidence [Safety Analysis / Test] is provided that demonstrates the effectiveness and sufficiency of these measures. Appropriate roles, responsibilities and methods were defined throughout the development of this system [Development Process Justification] [Safety Management System] and defined future operation.

Safety Arguments – Text Example

The Defence in Depth principle (P65) has been addressed in this system through the provision of the following:

- Multiple physical barriers between hazard source and the environment (see Section X)
- A protection system to prevent breach of these barriers and to mitigate the effects of a barrier being breached (see Section Y)

Safety Arguments should clearly describe how a safety objective / requirement / claim has been achieved in the system as proposed.

- how it has been interpreted
- ultimately, what evidence supports the requirements
Safety Arguments – Text Problems

For hazards associated with warnings, the assumptions of [7] Section 3.4 associated with the requirement to present a warning when no equipment failure has occurred are carried forward. In particular, with respect to hazard 17 in section 5.7 [4] that for test operation, operating limits will need to be introduced to protect against the hazard, whilst further data is gathered to determine the extent of the problem.

- not everyone can write clear English
- can take many readings to decipher meaning
- multiple cross-references in text can be awkward
- is there a clear shared understanding of the argument?

Presenting Clear Arguments

- It is possible in text – at least sometimes
  - use simple language and short sentences
  - use bullet points for key statements
  - break down the argument one step at a time
    - and refer to following sub-sections
  - structure document sub-sections around separate concepts
    - e.g. Section 6.2 – Control of Hazard – ‘Inadvertent Chaff Release’
- But it is easier with pictures!
  - use a graphical notation to summarise argument
    - Goal Structuring Notation (GSN)
    - Claims Argument Evidence (CAE)
The Goal Structuring Notation

Purpose of a Goal Structure

To show how goals are broken down into sub-goals, and eventually supported by evidence (solutions), whilst making clear the strategies adopted, the rationale for the approach (assumptions, justifications) and the context in which goals are stated.

A Simple Goal Structure

- Control System is Safe
- Hazards identified from FHA (Ref Y)
- Tolerability targets (Ref Z)
- All identified hazards eliminated / sufficiently mitigated
- Software developed to I.L. appropriate to hazards involved
- I.L. Process Guidelines defined by Ref X
- Probability of H2 occurring < 1 x 10^-6 per annum
- Primary Protection System developed to I.L. 4
- Secondary Protection System developed to I.L. 2
- 1x10^-6 p.a. limit for Catastrophic Hazards
- Formal Verification
- Fault Tree Analysis
- Process Evidence of I.L. 4
- Process Evidence of I.L. 2
A Simple Goal Structure

Safety Requirements & Objectives

- Hazards identified from FHA (Ref Y)
- Tolerability targets (Ref Z)
- Software developed to I.L. appropriate to hazards involved

Safety Argument

- All identified hazards eliminated/sufficiently mitigated
- H1 has been eliminated
- Probability of occurrence < 1 x 10^-6 per annum
- Primary Protection System developed to I.L. 4

Safety Evidence

- Probability of H2 occurring < 1 x 10^-3 per annum
- Secondary Protection System developed to I.L. 2

Group Exercise

- Create a goal structure to support the claim:

  **Top**
  This is a good pint of beer

- Process Guidelines defined by Ref X.
- J: 1 x 10^-6 p.a. limit for Catastrophic Hazards
Step 1 - Identify Goals: Phrasing

- Goals should be phrased as propositions
- Statements that can be said to be TRUE / FALSE (e.g. “The sky is blue” or “York is a beautiful city”)
- NB: not limited to statements that can be objectively proven
- Statement should be expressed as a single statement (1 sentence) of in the form:
  \(<\text{NOUN-PHRASE}>\text{<VERB-PHRASE>}\)
- Noun-Phrase identifies the subject of the goal
- Verb-Phrase defines a predicate over the subject
Step 1 - Identify Goals: Phrasing

- The following are examples of correctly stated goals:
  
<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component X</td>
<td>has no 'critical' failure modes</td>
</tr>
<tr>
<td>All identified hazards for System Y</td>
<td>have been sufficiently mitigated</td>
</tr>
<tr>
<td>Non-destructive examination of weld-site Z</td>
<td>has been performed</td>
</tr>
<tr>
<td>Design A</td>
<td>employs triple modular redundancy</td>
</tr>
</tbody>
</table>

- The following are examples of incorrectly stated goals:
  
  - "Hazard Log for System Y"  
    Reason: NP – describes an entity - not a statement
  - "Fault Tree for Hazard H1"  
    Reason: As above
  - "Perform Fault Tree Analysis of Hazard H1"  
    Reason: VP - an action - not a statement
  - "How many failure modes does component X have?"  
    Reason: Question - not a statement

- Test: can we say goal is TRUE / FALSE?
Step 1 – Example

G1

Press is acceptably safe to operate within CCC Whatford Plant

As with conventional safety case report – we wish to clearly set out the objective and scope of the safety argument being presented.

Step 2 - Define basis for goals: Context

Examine goal statement for terms / concepts not clearly defined within scope (context inherited as with models)

It is justified to leave a term ‘free’ if it will be defined through the course of the argument.
Step 2 – Example

- Terms – **Press, Operate** and **CCC Whatford Plant** drawn out explicitly as contextual information
- **Acceptably Safe** left for expansion through the supporting argument

Step 3 - Identify strategy

- Next step is to work out how to substantiate the stated goal
  - “What reasons are there for saying the goal is TRUE?”
  - “What statements would convince the reader that the goal is TRUE?”
- Aiming for statements that are easier to support than the larger goal
  - Breaking into a number of smaller goals - i.e. Divide-and-Conquer
  - Relating goal more closely to specific application in question (e.g. for a generic requirement)
Step 3 - Identify strategy

Examples:

- Requirement 6.3 (Defence in Depth) has been met
- 3 independent protection systems in place to shutdown system in case of detected abnormal operation

Divide-and-Conquer

Interpretation / Particularisation

Step 3 - Identify strategy: Phrasing

- The role of a strategy node is to clearly explain the relationship between a goal and a set of sub-goals
- An analogy:
  
  \[ 3xy^3 + 2x^2y^2 + 5xy = 17y \]  
  \( \text{(Divide both sides by } y) \)  
  \[ 3xy^2 + 2x^2y + 5x = 17 \]

- Strategy statement should succinctly describe the argument approach adopted, ideally in the form:
  - “Argument by … <approach>”

Example statements:

- “Argument by appeal to test results”
- “Argument by consideration of historical data”
- “Quantitative argument using simulated run data”
Step 3 - Identify strategy

Example

- **G5**
  - System is SAFE

- **G7**
  - Sub-system A is SAFE

- **G8**
  - Sub-system B is SAFE

- **G9**
  - Sub-system C is SAFE

These terms have been introduced

Is it obvious? Any assumptions?

Goal Structure Build-Up 3

- **C1**
  - Press Design

- **C2**
  - Press Operation

- **G1**
  - Press is acceptably safe to operate within CCC Whatford Plant

- **C3**
  - CCC Whatford Plant

- **S1**
  - Argument by addressing all identified operating hazards

- **S2**
  - Argument of compliance with all applicable safety standards & regulations

Two separate strategies – for reader’s benefit
Step 4 - Define basis for strategy

- In the same way as is necessary for goals, must examine what contextual information (including models) is required
- Same process - examine strategy for terms / concepts introduced but not ‘bound’
  - e.g. for sub-system breakdown strategy the term ‘All Identified sub-systems’ is used
- Ask what information is required in order to expand / fulfill strategy outlined

Step 4 - Example

- Needs Justification?
- Any Assumptions?
Step 5 - Elaborate strategy

- Having identified an approach, it is necessary to lay out the goals that fulfill that approach, e.g.
  - e.g., for strategy ranging over all sub-systems - expand for goals over each individual sub-system
  - e.g. for quantitative results strategy - provide quantitative goal statements
- In elaborating the strategy, again defining goals (back to Step 1)
- If strategy, and basis of strategy, are clear - this step can be straightforward
  - E.g. see next slide

Step 5 - Example

```
C4 All identified operating hazards
S1 Argument by addressing all identified operating hazards
S2 Argument of compliance with all applicable safety standards & regulations
C5 All applicable safety standards & regulations

G2 Hazard of 'Operator Hands Trapped by Press Plunger' sufficiently mitigated
G4 Hazard of 'Operator Upper Body trapped by Press Plunger' sufficiently

G3 Hazard of 'Operator Hands Caught in Press Drive Machinery' sufficiently mitigated
G5 Press compliant with U.K. HSE Provision and Use of Work Equipment Regulations
G6 Press compliant with U.K. enactment of EU Machinery Directive
G7 PES element of press design compliant with IEC61508
```
Step 6 - Identify Solutions

- Eventually, faced with a goal that doesn’t need further expansion, refinement, explanation ...
- In such cases, simply have to reference out to information that supports claim by means of solution
- As references, solutions should be NOUN-PHRASEs

Testing of Module XYZ123 showed no anomalies

Sn1
Software Test Results for Module XYZ123

Step 6 – Example

- Reference to source of information that would substantiate claim

G3
Hazard of ‘Operator Hands Caught in Press Drive Machinery’ sufficiently mitigated

G8
Motor / Clutch / Drive Belts surrounded with safety cage

G9
Press operation will (safely) halt if safety cage tampered with

Sn10
Press Design (Safety Cage)

More explanation required here
How to Create Goal Structures

- Two main approaches:
  - As an individual
  - As a group
    - Where aim is to reach common understanding and agreement of structure of the safety argument
    - Attendance: Safety Argument Owner (Principal Stakeholder), Experts, GSN Facilitator, Secretary (Optional)
    - Requires Key Documents to be circulated beforehand

Where to Brainstorm the Argument?

- Q: Where is it worth brainstorming an argument?
- Answers:
  - Wherever there is most uncertainty about the argument (key claims, evidence)
  - Wherever the argument is currently confused or is over-complex
  - Wherever there is disagreement about the argument
  - Wherever the consequences of having a wrong argument are high (in terms of rework, delays etc.)
When to Visualise the Argument?

Q: At what stage in a project is it worth visualising the argument?

Answers:
- Early on (high level) to get a clear picture (and gain agreement) of argument structure
  - Useful as a scoping exercise and effort allocation
- As project is progressing, in order to monitor status towards completion of an acceptable argument
- At end of project in order to present the final argument and evidence that exists

How to Present Goal Structures

- Customers are keen to see goal structures within safety documents
- Possible approaches to inclusion of GSN:
  - In full as Appendix / Annex to document
  - Integrated within body of document
    - Goal structure (1 level), Text, Goal structure, Text ...
    - See ‘Nuclear Trip System Safety Case Example’
  - As ‘Executive Summary’ at beginning of document
    - Maximum 2 pages of structure, 2-3 levels of decomposition
  - As separate, stand-alone, Index Document
    - e.g. to explain argument distributed across many safety case documents
Potential GSN Benefits

- Improving comprehension of existing arguments
- Improving discussion and reducing time-to-agreement on argument approaches being adopted
- (Having identified argument structure up front) focusing activities towards the specific end-objectives
- Recognition and exploitation of successful (convincing) arguments becomes possible
- Supports ‘light-weight’ evolution of an argument
- Supports monitoring of project progress towards a successful safety case

Conclusions

- Within conventional safety case reports the ‘chain of argument’ can often get lost
  - The argument is more important than the document!
  - GSN has been found to be a useful basis for mapping out and evolving the structure of the Safety Arguments
    - Provides a Road-map for a document / set of documents
    - Provides a basis for discussion amongst engineers and between developers and assessors
    - Creating an outline arguments towards beginning of project can be seen as making progress towards a final solution
Software Safety Case Management – Part Two

Dr Tim Kelly
Department of Computer Science
University of York

E-mail: tim.kelly@cs.york.ac.uk

Acknowledgements: Rob Weaver, Ibrahim Habli

Part 2 Overview

- Current approaches (and the problems) for Software Safety Case development
- An Evidence-based Framework for software safety arguments
  - Software Level Safety Requirements
  - Different types of Evidence
  - Different types of Hazardous Failure Modes To provide an overview of a ‘generic’ computer system safety argument structure

We will look at:
- Process Arguments
- Product Arguments
- Hazard-Based vs. Requirements-Based Arguments
- Functional vs. Non-Functional Issues
Problems with Current Practice

- Discontinuity in the safety case
  - System Argument has a Product Focus
  - Software Argument has a Process Focus
- Does Good Process = Good Product?
  - Questionable Assumption
- Application to Legacy and Commercial Off The Shelf (COTS) software
- Modern Practices – e.g. Code Generators
- Standards prescribe Software Development Process
  - Discourages intelligent thought about what evidence is useful or relevant
- Responsibility for safety lies with those that set the standard rather than those that build the software

Overall S/W SC Structure

Software Safety Case Management Tutorial – Tim Kelly

© Copyright Tim Kelly, 2007. Not to be reproduced without permission of author.
Top Level Argument

- Important of Context and Clear System Definition
- Process and Product (Historically more Process Emphasis)
- Both +/ve and -/ve aspects to Product Argument

![Diagram of argument structure]

Software ‘Fault Free’ Pattern

- As seen in U.K. DS 00-55

![Diagram of software 'Fault Free' pattern]
Conformance to Standards Argument

- SILS can govern required level of conformance
- Role for both conventional dev’t and ‘safety’ standards

Issues of Required SIL can govern required level of conformance to standards in this part of the argument

S3: Argument over safety lifecycle phases / activities

C3: Applicable standards

G4: [Computer system] hardware and software developed to [standard(s)]

G4a: [Computer system] hardware and software developed to safety [standard(s)]

G4b: [Computer system] hardware and software developed in accordance with appropriate development [standard(s)]

Covers aspects such as Config. Mgt., Gen Quality Mgt., conformance to coding standards, Development Environment

SIL Tailoring of Process Arguments

SIL (Process) Justifications:

Annex D - Tailoring Guide Across Differing Safety Integrity Levels

<table>
<thead>
<tr>
<th>Clause</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>36 Coding Process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36.1 Coding Standards</td>
<td>J1</td>
<td>J1</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>36.2</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>36.3</td>
<td>J2</td>
<td>J1</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>36.4</td>
<td>J2</td>
<td>J1</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>36.5 Static Analysis and Formal Verification</td>
<td>J1</td>
<td>J2</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- M = Must be Applied
- J1 = Justification of not following clause
  - Inapplicability
  - Cost-benefit (ALARP)
- J2 = Less detailed / rigorous justification
DO-178B Example

Table A-3 Verification of Output of Software Requirements Process

<table>
<thead>
<tr>
<th>Objective</th>
<th>Applicability by SW Level</th>
<th>Output</th>
<th>Control Category by SW Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Software high-level requirements comply with system requirements</td>
<td>6.3.1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 High-level requirements are accurate and consistent</td>
<td>6.3.1b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 High-level requirements are compatible with target computer</td>
<td>6.3.1c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 High-level requirements are verifiable</td>
<td>6.3.1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 High-level requirements conform to standards</td>
<td>6.3.1e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 High-level requirements are traceable to system requirements</td>
<td>6.3.1f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Algorithms are accurate</td>
<td>6.3.1g</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEGEND:

- The objective should be satisfied with independence
- The objective should be satisfied
- Blank
- Satisfaction of objective is at applicant’s discretion
- Data satisfies the objectives of Control Category 2

Software Safety Case Management Tutorial – Tim Kelly

51

© Copyright Tim Kelly, 2007. Not to be reproduced without permission of author

Safety Standards Argument

- Structured here by lifecycle phase
- Argument that safety considered appropriately at various points in the traditional development lifecycle
- Process Evidence, possibly with requirement for independence, needed here

Software Safety Case Management Tutorial – Tim Kelly

52

© Copyright Tim Kelly, 2007. Not to be reproduced without permission of author
‘Traditional’ Development Standards Arg.

- Necessary but not sufficient elements
- Still concerns general ‘integrity’ of finished product

G4:
[Computer system] hardware and software developed to [standard(s)]

G4b:
[Computer system] hardware and software developed in accordance with appropriate development [standard(s)]

Covers aspects such as Config. Mgt., Gen Quality Mgt., conformance to coding standards, Development Environment

Product Argument

- Functional and Non-functional split possible both under hazard based argument and requirements conformance argument
- Requirements based argument can deal with cascade from system level
- Can be overlap between requirements and hazard based arguments

S1: Argument over product and process

S2: Argument over functional and non-functional properties

S3: [Computer system] does not contribute to [system] hazards

C4: [Computer system] requirements

G2: [Computer system] satisfies explicit safety requirements

Non-functional, functional split also possible here (e.g. timing requirements - leading to test/results evidence could be addressed here)
**Functional Behaviour Argument**

- Must demonstrate exhaustive identification of contributions
- Typically requires both top-down and bottom-up approaches
- Useful to pin-point contributions to dev’t items in software structure

**S2:** Argument over functional and non-functional properties

**G5:** Functional behaviour of [computer system] does not contribute to [system] hazards

**G14:** Potential [computer system] software and hardware contributions to system hazards have been identified

**G15:** [Computer system] functions related to system hazards have been identified

**G16:** [Computer system] software and hardware contributions to system hazards have been eliminated or mitigated

**Sn1** System Level Hazard Analysis

**Sn2** Functional Failure Analysis

---

**Functional Contribution Argument 1**

**G18** [Computer system] software and hardware contributions to system hazards have been eliminated or mitigated

**G19** [Computer system] requirements defined s.t. contribution eliminated or mitigated

**G20** (Analysis) [Computer system] code implementation defined s.t. contribution eliminated or mitigated

**G21** (Computer system) testing defined such that elimination or mitigation of contribution demonstrated

**G22** (Computer system) testing demonstrates elimination or mitigation of contribution

**Sn3** Requirements Level Hazard Analysis

**Sn4** Design Level Hazard Analysis

**Sn5** Code Level Hazard Analysis

**Sn6** Test Coverage Analysis

**Sn7** Test Evidence
Functional Contribution Argument 2

- Need to demonstrate systematic consideration of each contribution
- Ideally, wish to see consideration through lifecycle
- Often best solutions to potential contribution are early lifecycle
- Argument can be based on both analysis (e.g. of specification) and test (of implementation)
- For Testing – must demonstrate coverage of contributions
  - May be possible to integrate with conventional acceptance testing

Non-Functional Contribution Argument

S2: Argument over functional and non-functional properties

G6: Non-functional properties and behaviours of [computer system] do not contribute to [system] hazards

- Remember, non-functional requirements dealt with elsewhere
- Dealing here with non-intentional implementation ‘hazards’
  - Sometimes described at ‘Computing’ Hazards
- Problem of Completeness
  - Standard Lists exist (e.g. 882C)

Covers aspects such as Resource Usage / Starvation, Bus Clash, Illegal Instructions, Exception Handling, Control Flow Anomalies, Data Flow Anomalies, Moding Problems, Variable Initialisation, etc.
Evidence-Based Framework

- System Level Requirements
- Requirement for System Level Evidence
  - Validation, Satisfaction, Traceability
- Software Level Safety Requirements
  - Focus on Software Contributions to System Level Hazards
- Requirement for System level Evidence
  - Validation, Satisfaction, (Traceability)
- Classification of Hazardous Failure Modes
  - Omission, Commission, Early, Late, Value
- Generic Arguments
- Targeted Selection of Evidence

Example Case Study – Air to Air Missile

Seeker

Fuse

Warhead (or Telemetry Unit)

Umbilical

Actuator

Inertial Measurement Unit (IMU)

Electronics & Power Unit (EPU)

Data Link
System Level Safety Requirements

- System Level Hazard Analysis
- Handling Hazards
- Hazards to Launcher
  - Premature Detonation (or Break-up, on telemetry rounds)
  - Premature Launch
  - Disintegration near or in front of launcher due to high-g or high-roll manoeuvres
  - Hitting the launcher due to incorrect trajectory
  - Premature fin movement (i.e. prior to launch, perhaps damaging missile or launcher)
  - Hang-fire (excessive delay between ignition and thrust)
  - Hang-up (missile remains on launcher but thrusts)
- Hazards to Friendly-Forces
- Hazards of Mission Failure

System Level Evidence Categorisation

- Validation
  - Demonstration that the set of System Safety Requirements is complete
- Satisfaction
  - Demonstration that all System Safety Requirements have been met
- Traceability
  - Demonstration that all System Safety Requirements have been tracked throughout System Development and Safety Analysis

Next stage is to consider Hardware, Software and Other contributions to System-level Hazards
Pattern – System-level SR

- **SysDefn**
  - System Definition

- **SystemSafe**
  - (System) is acceptably safe to operate from a hazard control perspective

- **DefnAccSafe**
  - Definition of acceptably safe

- **ReqValid**
  - System Safety Requirements are valid

- **HazAccept**
  - All identified system level hazards occur at acceptably low rates

- **SysHaz**
  - Identified System Level Hazards

- **Traceability**
  - Traceability of safety requirements and safety evidence has been shown

- **ArgSWHWOther**
  - Argument across software, hardware and other parts of (System) that may cause hazards

- **DependExplicit**
  - System can be decomposed as all dependencies between different parts of the system are explicit

Evidence-Based Framework

- **System Level Requirements**
- **Requirement for System Level Evidence**
  - Validation, Satisfaction, Traceability
- **Software Level Safety Requirements**
  - Focus on Software Contributions to System Level Hazards
- **Requirement for System level Evidence**
  - Validation, Satisfaction, (Traceability)
- **Classification of Hazardous Failure Modes**
  - Omission, Commission, Early, Late, Value
- **Generic Arguments**
- **Targeted Selection of Evidence**
Software Level Safety Requirements

- Derived from System Level Safety Requirements
  - From System Level Perspective can be seen as a “Basic Event”
  - From Software Level Perspective can be seen as the “Top Level”

- Hazard Based
  - Potential Failures within the Software that can lead to System Level Hazards

- Example – Software Safety Requirement
  - Acceptability of Software Failure Mode - Software Fails to block premature launch
Software Level Evidence Categorisation

- **Validation**
  - *Demonstration that the set of Software Safety Requirements is complete*

- **Satisfaction**
  - *Demonstration that all Software Safety Requirements have been met*

- **Traceability**
  - *Demonstration that all Software Safety Requirements have been tracked throughout System Development and Safety Analysis*

Pattern – Software Level SR

- **SWDefn**  
  - Software Definition

- **SWContrib**  
  - All software contributions to system level hazards have been identified

- **ArgOverSWContrib**  
  - Argument over all identified software contributions to system level hazards

- **HSFMAccept**  
  - All causes of Hazardous Software Failure Mode (HSFM) are acceptable

- **SWContribAccept**  
  - Software contributions to System Level Hazards are acceptable

- **SWSRTraceability**  
  - Traceability of software safety requirements and safety evidence has been shown

- **HSFM**  
  - Hazardous Software Failure Mode

- **n**  
  - Number of software hazardous failure modes from SWContrib

Software Safety Case Management Tutorial – Tim Kelly

© Copyright Tim Kelly, 2007. Not to be reproduced without permission of author.
Validation and Traceability

- **Validation**
  - Completeness
    - Functional Failure Analysis/HAZOP

- **Traceability**
  - DO-178B
    - Traceability between the system requirements and software requirements.
    - Traceability between the low-level requirements and high-level requirements
    - Traceability between the Source Code and the low-level requirements
  - Traceability Matrix

Evidence-Based Framework

- **System Level Requirements**
- **Requirement for System Level Evidence**
  - Validation, Satisfaction, Traceability
- **Software Level Safety Requirements**
  - Focus on Software Contributions to System Level Hazards
- **Requirement for System level Evidence**
  - Validation, Satisfaction, (Traceability)
- **Classification of Hazardous Failure Modes**
  - Omission, Commission, Early, Late, Value
- **Generic Arguments**
- **Evidence Characteristics**
  - Relevance, Coverage, Independence
Hazardous Failure Mode Classification

- Omission, Commission, Early, Late, Value
  - Service Provision, Service Timing, Value

- Common arguments for different classes
  - Certain arguments (and evidence types) can be used for different failure types

- It is possible to produce generic safety case arguments that can be reused.

Example

- Software Failure Mode - Software Fails to block premature launch

Example - Software Architecture 1
Software Fails to block premature launch caused by:

- Software interlock handler fails to write to interlock pool – Omission

Failure Type (Service Provision Failure Type)

Example - Software Architecture 2

Pattern – Failure Mode Classification
Evidence-Based Framework

- System Level Requirements
- Requirement for System Level Evidence
  - Validation, Satisfaction, Traceability
- Software Level Safety Requirements
  - Focus on Software Contributions to System Level Hazards
- Requirement for System Level Evidence
  - Validation, Satisfaction, (Traceability)
- Classification of Hazardous Failure Modes
  - Omission, Commission, Early, Late, Value
- Generic Arguments
- Targeted Selection of Evidence

Arguing about Requirements

- Satisfaction of a Typed Hazardous Failure Requirement

- Demonstration that the requirement has been met shown through combination of
  - Absence
  - Handling
  - (Probability – we can't show this for software)

- Example
  - Show Absence of Omission Hazardous Failure Mode - Software output module fails to provide braking value to actuator
Pattern – Argument Approach

**Hazardous Software Failure Mode (HSFM)**

- All causes of Hazardous Software Failure Mode (HSFM) of type {type} are acceptable

**Software Definition (SWDefn)**

- Hazardous Software Failure Mode (HSFM) of type {type} are absent in contributory software functionality (CSF)

**Argument over Absence and/or Handling of Hazardous Software Failure Mode**

- Argument for Absence Omission Failure
  - Primary Argument
    - All feasible paths through software functionality contain a unique output statement
  - Secondary Argument
    - Failure of other software functionality which could lead to a failure of primary software functionality does not occur
    - All necessary resources exist to support correct operation of primary software functionality
  - Control Argument
    - Primary software functionality is scheduled and allowed to run (at least) once
Product Oriented Decomposition

ContribSWFunc
- Identified Software Functionality which contributes to software hazardous failure mode (HSFM)

SafeRespCSF
- Safety Requirements of contributory software functionality

HSFM
- Hazardous Software Failure Mode

DefnOPS
- Definition of output statement

AllOnPrimary
- All feasible control paths through CSF include a unique output statement

AllOnSecondary
- Failures of other components which could lead to CSF Omission Hazardous Failure Mode are acceptable

AllOnControl
- CSF is scheduled and allowed to run once

ContextCSF
- Within the context of contributory software functionality (CSF)

CauseOmHaz
- Known causes of Omission Hazardous Failure Mode

AllCauses
- Identified failure mechanisms describe all known causes of Omission Hazardous Failure Mode

ArgFailureMech
- Argument over failure mechanisms

ArgHSFM
- Hazardous Software Failure Mode (HSFM) of type Omission

A

J

Software Architecture

Secondary Argument

Primary Argument
Secondary Causes Decomposition

- **HSFM**: Hazardous Software Failure Mode
- **ContribSWFunc**: Identified Software Functionality which contributes to hazardous software failure mode (HSFM)
- **Ab{type}Secondary**: Failure of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **ArgOtherSWFuncCause**: Argument over other Software Functionality, Hardware Components or Other Components identified as cause of hazardous software failure mode
- **Ab{type}OtherFn**: Failures of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **AbResources**: Necessary Resources exist to support correct operation of CSF
- **HSFM**: Hazardous Software Failure Mode
- **ContribSWFunc**: Identified Software Functionality which contributes to hazardous software failure mode (HSFM)
- **Ab{type}Secondary**: Failure of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **ArgOtherSWFuncCause**: Argument over other Software Functionality, Hardware Components or Other Components identified as cause of hazardous software failure mode
- **Ab{type}OtherFn**: Failures of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **AbResources**: Necessary Resources exist to support correct operation of CSF
- **HSFM**: Hazardous Software Failure Mode
- **ContribSWFunc**: Identified Software Functionality which contributes to hazardous software failure mode (HSFM)
- **Ab{type}Secondary**: Failure of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **ArgOtherSWFuncCause**: Argument over other Software Functionality, Hardware Components or Other Components identified as cause of hazardous software failure mode
- **Ab{type}OtherFn**: Failures of other functionality which could lead to CSF (type) Hazardous Failure Mode are acceptable
- **AbResources**: Necessary Resources exist to support correct operation of CSF

---

**Secondary Causes Decomposition 2**

- **HWDefn**: Hardware Definition
- **HWFM**: Hazardous Hardware Failure Mode
- **HWFAbsent_Hardware Safety Argument**: Hazardous Hardware Failure Mode absent in contributory hardware
- **HWFProb_Hardware Safety Argument**: Probability of Hazardous Hardware Failure Mode in contributory hardware acceptably low
- **HWHandl**: Occurrence of Hazardous Hardware Failure Mode {HHFM} of type {type} acceptably detected and handled
- **HHFM**: Hazardous Hardware Failure Mode
- **HHFMAccept**: All causes of Hazardous Hardware Failure Mode {HHFM} are acceptable
- **HWAbsent_Hardware Safety Argument**: Hazardous Hardware Failure Mode absent in contributory hardware
- **HWFProb_Hardware Safety Argument**: Probability of Hazardous Hardware Failure Mode in contributory hardware acceptably low
- **HWHandl**: Occurrence of Hazardous Hardware Failure Mode {HHFM} of type {type} acceptably detected and handled
- **HHFM**: Hazardous Hardware Failure Mode
- **HHFMAccept**: All causes of Hazardous Hardware Failure Mode {HHFM} are acceptable
- **OFAbsent_Other Safety Argument**: Hazardous Other Component Failure Mode absent in other contributory component
- **OFProb_Other Safety Argument**: Probability of Hazardous Other Component Failure Mode in other contributory component acceptably low
- **OFHandl**: Occurrence of Hazardous Other Component Failure Mode {HOFM} of type {type} acceptably detected and handled
- **HOFM**: Hazardous Other Component Failure Mode
- **OFAbsent_Other Safety Argument**: Hazardous Other Component Failure Mode absent in other contributory component
- **OFProb_Other Safety Argument**: Probability of Hazardous Other Component Failure Mode in other contributory component acceptably low
- **OFHandl**: Occurrence of Hazardous Other Component Failure Mode {HOFM} of type {type} acceptably detected and handled
- **HOFM**: Hazardous Other Component Failure Mode
- **OFAbsent_Other Safety Argument**: Hazardous Other Component Failure Mode absent in other contributory component
- **OFProb_Other Safety Argument**: Probability of Hazardous Other Component Failure Mode in other contributory component acceptably low
- **OFHandl**: Occurrence of Hazardous Other Component Failure Mode {HOFM} of type {type} acceptably detected and handled
- **HOFM**: Hazardous Other Component Failure Mode

---

Software Safety Case Management Tutorial – Tim Kelly

© Copyright Tim Kelly, 2007. Not to be reproduced without permission of author.
Targeted Selection of Evidence

- **IEC 61508**

<table>
<thead>
<tr>
<th>Technique/Measure</th>
<th>Ref</th>
<th>SIL.1</th>
<th>SIL.2</th>
<th>SIL.3</th>
<th>SIL.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Formal proof</td>
<td>C.5.13</td>
<td></td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>2. Probabilistic testing</td>
<td>C.5.1</td>
<td></td>
<td></td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>3. Static analysis</td>
<td>8.6.4</td>
<td>Table B.6</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>4. Dynamic analysis and testing</td>
<td>8.6.5</td>
<td>Table B.2</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>5. Software complexity metrics</td>
<td>C.5.14</td>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

- Software module testing and integration: See table A.6
- Programmable electronics integration testing: See table A.6
- Software system testing (validation): See table A.7

Selection of techniques is difficult except with respect to the Safety Case Objectives

- A problem with the current standards
- Approach drives the argument down to low-level objectives first

Process and Product Argument 1

![Diagram showing Process and Product Argument 1](image-url)
Process and Product Argument 2

Product Argument

- **SysDefn**: System Definition
- **OpCtxt**: System operating context
- **HazLog**: Hazard Log
- **HazMit**: Mitigation argument for all identified hazards
- **HazXMit**: Hazard X is sufficiently mitigated
- **HazYMit**: Hazard Y is sufficiently mitigated
- **HazZMit**: Hazard Z is sufficiently mitigated
- **Top**: System is acceptably safe to operate
- **AllHazIdent**: All credible hazards have been exhaustively identified

Process and Product Argument 3

Process Argument

- **AllHazIdent**: All credible hazards have been exhaustively identified
- **SysHA**: Hazard Analysis conducted using an established, systematic techniques
- **SQEPHA**: Suitably Qualified and Experienced Personnel undertook the Hazard Analysis
- **DataInHA**: Hazard Analysis was based on accurate and representative system data
Elements of Process Argument

- Traceability of artefacts
  - E.g. traceability of requirements to design, implementation and verification
- Competency of personnel
  - E.g. experts, practitioners or supervised
- Suitability and reliability of Methods
  - E.g. testing or analysis, formal or informal
- Qualification of tools
  - E.g. Development or verification tools, qualified or unqualified
- Suitability and clarity of notations
  - E.g. formal, informal or structured
- Independence
  - E.g. independence at the personnel and organisation level

Product Argument Example

- Black box testing and state machine analysis provide explicit and independent evidence (solutions) directly related to system artefact rather than appealing to quality of development process
Product Argument Example: Process Uncertainty

- Black box testing process
  - Is testing team independent from design team?
  - Is process of generating, executing and analysing test cases carried out systematically and thoroughly?
  - Is traceability between safety requirements and test cases well established and documented?

- State machine analysis process
  - How accurate is correspondence between the mathematical model and software behaviour at run-time?
  - Is analysis carried out by mathematically qualified engineers?

Integrating Process Arguments into Product Arguments

- Process uncertainty should be addressed by linking process arguments to items of evidence used in product argument

<table>
<thead>
<tr>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Argument</td>
</tr>
<tr>
<td>Process Argument</td>
</tr>
<tr>
<td>Safety Requirements</td>
</tr>
<tr>
<td>Process Criteria</td>
</tr>
</tbody>
</table>

Useful in absence of prescribed process
Evidence-based ≠ Product-Based
Product Argument Example Revisited

Software Safety Case Management Tutorial – Tim Kelly

Product & Process Argument in Modular GSN

Software Safety Case Management Tutorial – Tim Kelly
**Process Argument: Black Box Testing is Trustworthy**

- **BXTestingTrustworthy**
  - Black box testing is trustworthy

- **ArgBXTestProcess**
  - Argument by considering black box testing process

- **BXTestTeam**
  - Testing team is competent

- **BXTestCV**
  - Experience, authority & training

- **BXTestInd**
  - Independence from design team

- **BXTestRqSys**
  - Testing method addresses safety requirements systematically

- **ArgBXTestCaseGen**
  - Argument by considering test case generation & execution

- **BXTestCaseGen**
  - Test case generation was thorough

- **BXTestAnalysis**
  - Black box test result was analysed

- **BXTestRqGen**
  - Test case generation was thorough

- **BXTestPlatFrm**
  - Target platform

- **BXTestCodeV**
  - Final version of object code

- **BXTestCaseCov**
  - Safety requirements coverage assessment

- **BXTestCaseExec**
  - Test cases were executed on stable platform & object code version

**Summary**

- Generic Computer System Safety Argument shown
- Both process and product elements required
  - However, historically perhaps over-emphasis on process
- Evidence-based S/W Safety Argument Framework
  - Traceable to system-level hazards
  - Based upon taxonomy of software failure mode types
  - Validation, Satisfaction and Traceability
  - Patterns of required arguments
  - Leading to targeted evidence selection
- Linking Process and Product Arguments
- Importance of a Structured Approach to Assurance Case Construction and Presentation!