

# Assurance Cases and Hazard Analysis

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Charles B. Weinstock  
April 29, 2008



# The System Assurance Problem

---

Systems are getting more complex and more dependent on software

- Reaching sound conclusions about reliability is getting harder

Traditional methods for evaluating dependable behavior are increasingly inadequate

- Too costly (in time and money) to test complex systems well
- Testing is not the best way of showing impact of subtle, but critical errors
- Constraining interactions among system components can make it easier to increase reliability but may be hard to find constraints consistent with required functionality

We need better ways of integrating a variety of analyses (evidence) into assurance cases, i.e.,

- We need better means of showing how certain evidence supports conclusions about system properties



# Overview

---

## **Introduction to Assurance Cases**

Hazard Analyses and Assurance Cases

The SEI Assurance Cases for Medical Devices Project

Conclusions



# Assurance Cases

---

An assurance case presents an argument that a system is acceptably safe, secure, reliable, etc. in a given context

- A system could be physical, a combination of hardware and software, or procedural (e.g., operating rules)

Experience with assurance cases has mainly been for safety cases



# Argument and Evidence

---

An assurance case requires claims, evidence, and an argument linking evidence to claims:

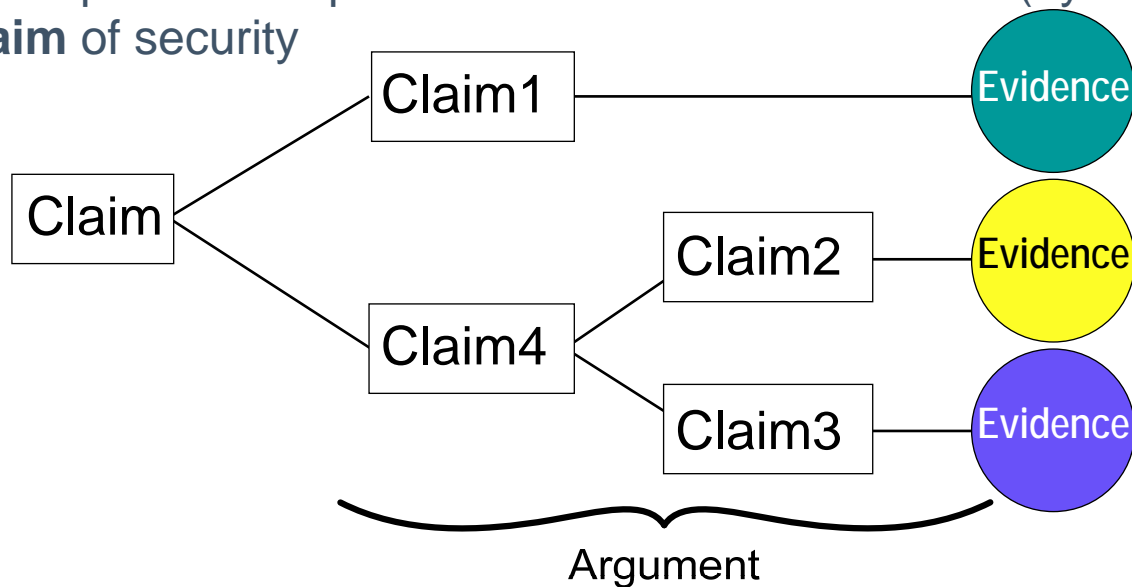
- Evidence
  - Results of observing, analyzing, testing, simulating, and estimating the properties of a system that provide fundamental information from which the presence of some system property can be inferred
- High Level Argument
  - Explanation of how the available evidence can be reasonably interpreted as indicating acceptable operation, usually by demonstrating compliance with requirements, sufficient mitigation of hazards, avoidance of hazards, etc.



# What is an Assurance Case?

A structured demonstration that a system is acceptably safe, secure, reliable, etc.

- A comprehensive presentation of **evidence** linked (by **argument**) to a **claim** of security



**IF ● THEN Claim1; IF ● THEN Claim2; IF ● THEN Claim3;**  
**IF Claim2 and Claim3 THEN Claim4; IF Claim1 and Claim4 THEN Claim**



# Goal Structuring Notation

---

Goal Structuring Notation (GSN) was developed to help organize and structure Safety Cases in a readily reviewable form

GSN has been used in Safety Case development for over a decade. A brief overview of its history is in [Kelly 04]

GSN has been successfully used to document Safety Cases for systems such as aircraft avionics, rail signaling, air traffic control, and nuclear reactor shutdown


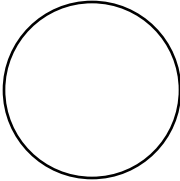

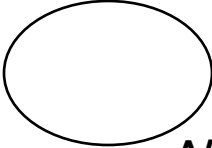

We have used it to build cases showing that other attributes of interest (e.g., security) have been met.

[Kelly 04] Tim Kelly, and Rob Weaver. The Goal Structuring Notation — A Safety Argument Notation. <http://www-users.cs.york.ac.uk/~rob/papers/DSN04.pdf>



# Goal Structuring Notation

---

To show how **claims**  are broken down into sub-claims,  
and eventually supported by **evidence**   
while making clear the argumentation **strategies**  adopted,  
the rationale for the approach (**assumptions, justifications**)   
and the **context**  in which claims are stated A/J



### **C: Timing Constraints Satisfied**

Timing constraints for all mission (or safety-) critical functions are satisfied under worst case execution timing

## functions

Functions are those whose failure is likely to cause failure of the function. We here refer to critical functions whose failure is caused by a failure of timing constraints.

### **C: Timing Constraints Satisfied**

Timing constraints for all mission (or safety-) critical functions are satisfied under worst case execution timing.

## functions

Functions are those whose failure is likely to cause failure of the function. We here define critical functions as those whose failure is caused by a failure to meet timing constraints.

### C: Timing Constraints Satisfied

Timing constraints for all mission (or safety-) critical functions are satisfied under worst case execution timing.

### Ctxt: Functions and Tim

Critical functions are defined in Document XXX, version N, Section Y. Timing constraints and their relation to critical functions are defined in document Z, version N.

**functions**

Functions are those whose failure is likely to cause failure of the function. We here define critical functions as those whose failure is caused by a failure of timing constraints.

**C: Timing Constraints Satisfied**

Timing constraints for all mission (or safety-) critical functions are satisfied under worst case execution timing.

**Ctxt: Functions and Tim**

Critical functions are defined in Document XXX, version N, Section Y. Timing constraints and their relation to critical functions are defined in document Z, version N.

**S: Analysis and Timing**

Argue by appealing to analytical and testing results.

**functions**

Functions are those whose failure is likely to cause failure of the function. We here define critical functions as those whose failure is caused by a failure of timing constraints.

**C: Timing Constraints Satisfied**

Timing constraints for all mission (or safety-) critical functions are satisfied under worst case execution timing.

**Ctxt: Functions and Tim**

Critical functions are defined in Document XXX, version N, Section Y. Timing constraints and their relation to critical functions are defined in document Z, version N.

**S: Analysis and Timing**

Argue by appealing to analytical and testing results.

# Presenting Clear Cases

---

## Basic structure

- Claim: what we want to show
  - A proposition: either true or false
- Argument: why we believe the claim is met, based on
- Evidence: test results, analysis results, etc.

## In general, arguments are structured hierarchically

- Claim, argument, sub-claims, sub-arguments, evidence
- Easy to show graphically, although can be done in a document or tabular structure



# Potential Assurance Case Benefits

---

Improves comprehension of existing arguments

Improves discussion and reduces time-to-agreement on what evidence is needed and what the evidence means

(Having identified argument structure up front) focuses activities towards the specific end-objectives

Recognition and exploitation of successful (convincing) arguments becomes possible (assurance case patterns)

Supports monitoring of project progress towards successful certification



# Overview

---

Introduction to Assurance Cases

**Hazard Analyses and Assurance Cases**

The SEI Assurance Cases for Medical Devices Project

Conclusions



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Faulty data exchanged among redundant computers causes all computers to fail. This could occur because of Improper requirements, incorrect coding of logic, incorrect data definitions (e.g., initialized data), and/or inability to test all possible modes in the SW</p>	<p>Effect: Loss of operation of system during critical phase, leading to loss of life. Severity: Catastrophic Likelihood: Improbable Class: Controlled</p>	<p>a) Software safeguards reduce, to the maximum extent feasible, the possibility that faulty data sent among redundant computers causes them to fail b) Program Development Specifications and Functional SW Requirements c) Subsystem design and functional interface requirements are used in the design and development of the relevant SW d) ...</p>	<p>Extensive validation and testing are in place to minimize generic SW problems. The contractors must perform rigorous reviews throughout the SW definition, implementation, and verification cycles. These review processes cover requirements, design, code, test procedures and results, and are designed to eliminate errors early in the SW life cycle. After completing system level verification, critical SW undergoes extensive integrated HW/SW verification at facility XXX Extensive verification is independently performed at facility XXX, using hardware and software maintained to duplicate the configuration of the fielded system</p>

**Portion of a typical, traditional, software-related hazard report**



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Faulty data sent among redundant computers</p> <p>This could be caused by improper initialization of data, and/or inability to test all possible modes in the SW</p>	<p>Effect: Loss of operation of system during critical phase leading to loss of data</p> <p>Severity: Catastrophic</p> <p>Likelihood: Improbable</p> <p>Class: Controlled</p>	<p>a) Software safeguards are used to reduce, to the maximum extent feasible, the possibility that faulty data sent among redundant computers causes them to fail</p> <p>b) ...</p> <p>c) ...</p> <p>d) ...</p>	<p>Validation and testing are in progress through static analysis, code reviews, and results, and are designed to detect errors early in the SW life cycle</p> <p>Upon completing system level verification, the SW undergoes extensive integrated HW/SW verification at facility XXX</p> <p>Extensive verification is independently performed at facility XXX, using hardware and software maintained to duplicate the configuration of the fielded system</p>

What software safeguards?



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	
<p>Faulty data exchanged among redundant computers causes all computers to fail.</p> <p>This could occur because of improper requirements, incorrect coding of logic, incorrect data definitions (e.g., initialized data), and/or inability to test all possible modes in the SW</p>		<p>b) Program Development Specifications and Functional SW Requirements</p> <p>c) Subsystem design and functional interface requirements ...</p> <p>d) ...</p>	<p>How are requirements a mitigation for this cause? How so? Which ones?</p> <p>... processes cover design, code, test and results, and are designed to catch errors early in the SW life</p> <p>... including system level verification, and undergoes extensive hardware/SW verification at facility</p> <p>... Extensive verification is independently performed at facility XXX, using hardware and software maintained to duplicate the configuration of the fielded system</p>



# From Hazard Report to Assurance Case

What is “extensive”?

What are “rigorous reviews”? How do they eliminate errors?

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	
<p>Faulty data exchanged among redundant computers causes all computers to fail.</p> <p>This could occur because of improper requirements, incorrect coding of logic, incorrect data definitions (e.g., initialized data), and/or inability to test all possible modes in the SW</p>		<p>a) Software safety reduce, to the maximum extent feasible, the possibility that faults sent among redundant computers cause to fail</p> <p>b) Program Development Specifications and Functional SW Requirements</p> <p>c) Subsystem design functional interface requirements are the design and development of the relevant SW</p> <p>d) ...</p>	<p><b>Extensive</b> validation and testing are in place to minimize generic SW problems. The contractors must perform <b>rigorous reviews</b> throughout the SW definition, implementation, and verification cycles. These review processes cover requirements, design, code, test procedures and results, and are designed to eliminate errors early in the SW life cycle.</p>



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Fault exchange redundancy components all components. This because requirements incorrect logic, incorrect data definitions (e.g., uninitialized data), and/or inability to test all possible modes in the SW</p>	<p><b>These ambiguities in the hazard report make certification assessment more difficult.</b></p>	<p>a) re exchange possible scenarios to b) Sp Functional SW</p> <p><b>The answers may be available elsewhere, but we are left to find them ourselves.</b></p>	<p>Validation and testing are in nize generic SW problems. rs must perform rigorous houghout the SW definition, n, and verification cycles. processes cover design, code, test nd results, and are designed rrors early in the SW life cycle.</p> <p>After crit inte XX Ext per and configuration of the fielded system</p> <p><b>The mitigations and verification actions are implicitly related to the cause/hazard.</b></p>
<p>Make the relationship more explicit by constructing claims and evidence in an assurance case.</p>			



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Faulty data exchanged among redundant computers causes all computers to fail.</p> <p>This could occur because of improper requirements, incorrect coding of logic, incorrect data definitions (e.g., initialized data), and/or inability to test all possible modes in the SW</p>		<p>a) Software safeguards reduce, to the maximum extent feasible, the possibility that sent among redundant computers cause to fail</p> <p>b) Program Design Specifications Functional SW Requirements</p> <p>c) Subsystem functional interface requirements at the design and development of relevant SW</p> <p>d) ...</p>	<p>Extensive validation and testing are in place to minimize generic SW problems.</p> <p><b>Claim: The possibility that 'Faulty data exchanged among redundant computers causes all such computers to fail (during critical mission phases)' has been reduced ALARP</b></p>



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Faulty data exchanged among redundant computers causes all computers to fail.</p> <p>This could occur because of improper requirements, incorrect coding of logic, incorrect data definitions (e.g., initialized data), and/or inability to test all possible modes in the SW</p>		<p>a) Software safeguards reduce, to the maximum extent feasible, the possibility that faulty data sent among redundant computers causes them to fail</p> <p>b) Program Specifications Functional Requirements</p> <p>c) Subsystem functional requirements are used in the design and development of the relevant SW</p> <p>d) ...</p>	<p>Extensive validation and testing are in place to minimize generic SW problems. The contractors must perform rigorous reviews throughout the SW definition, implementation, and verification cycles. These review processes cover code, test, and are designed in the SW life</p> <p>in level verification, extensive verification at facility</p> <p>XXX</p> <p>Extensive verification is independently performed at facility XXX, using hardware and software maintained to duplicate the configuration of the fielded system</p>

**Better claims: Each possible cause has been mitigated (differently)**



# From Hazard Report to Assurance Case

Cause/Fault Tree Ref	Effect/Severity/Likelihood	Mitigation	Verification
<p>Faulty data exchanged among redundant computers caused all computers to fail. This could occur because of incorrect requirements, incorrect coding logic, incorrect definitions (e.g. uninitialized data) and/or inability to test all possible modes in the SW</p>	<p><b>Effect:</b> Loss of operation of system during critical phase, leading to loss of life.</p> <p><b>Severity:</b> Catastrophic</p> <p><b>Likelihood:</b> Improbable</p> <p><b>Class:</b> Controlled</p>	<p>are safeguards to the maximum extent possible, the probability that faulty data will be long reduced. Errors caused by program Development and initial SW elements system design and al interface elements are used in the design and development of the relevant SW</p> <p>d) ...</p>	<p>Extensive validation and testing are in place to minimize generic SW problems. The contractors must perform rigorous reviews throughout the SW definition, development cycles. Tests are designed to eliminate errors early in the SW life cycle.</p> <p>After completing system level verification, critical SW undergoes extensive integrated HW/SW verification at facility XXX</p> <p>Extensive verification is independently performed at facility XXX, using hardware and software maintained to duplicate the configuration of the fielded system</p>

**Claim context and assumptions**

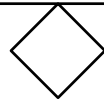


# AC to Resolve These Problems

---


## **C: Data Exchange Haz Mitigated**

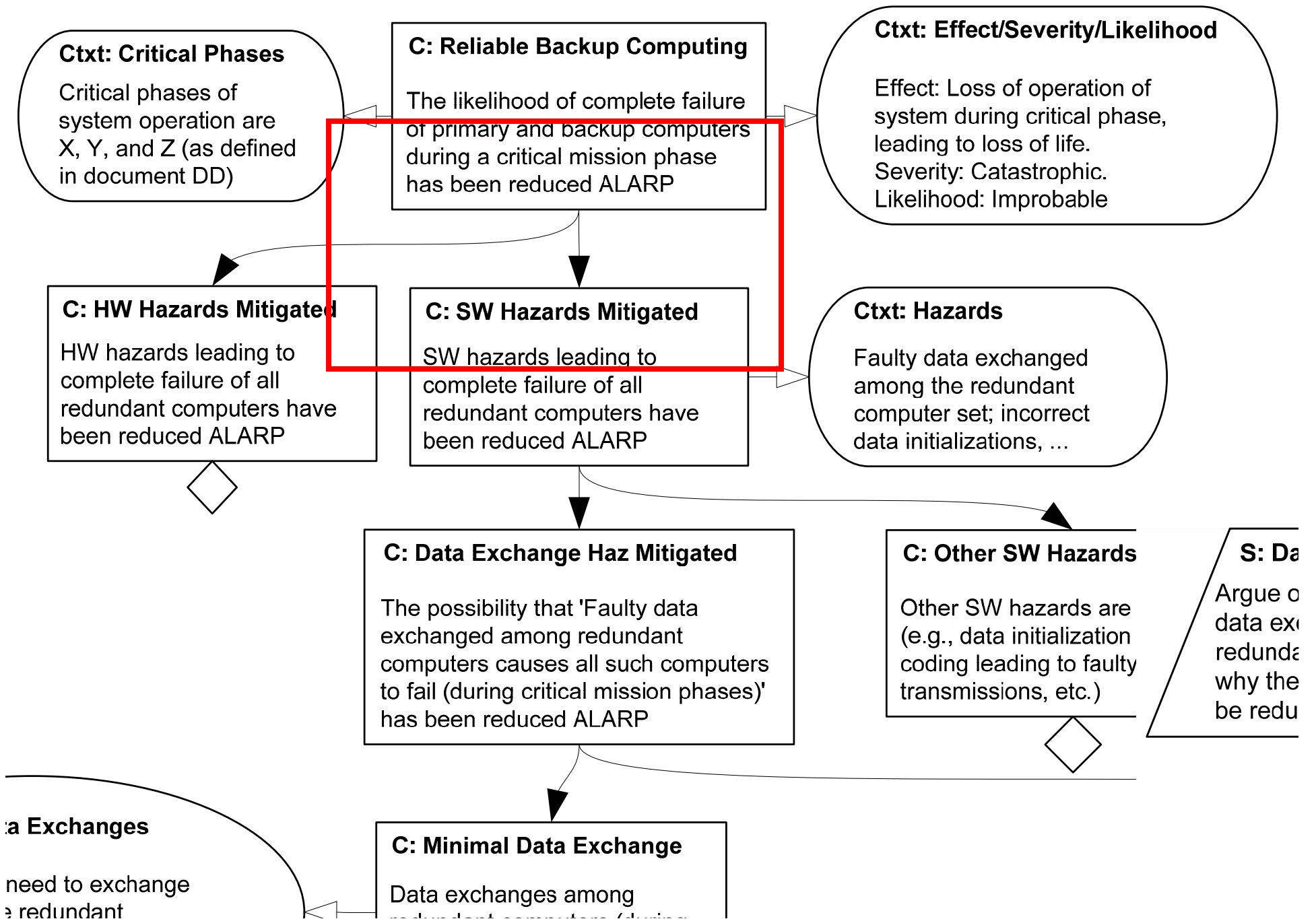
The possibility that 'Faulty data exchanged among redundant computers causes all such computers to fail (during critical mission phases)' has been reduced ALARP'



### **C: Reliable Backup Computing**

The likelihood of complete failure of primary and backup computers during a critical mission phase has been reduced ALARP





# Assurance Case Benefits

---

It's not that we don't already produce this data, but we don't present it effectively

- The motivation for some activities and decisions is not evident to outside reviewers or new personnel

Effective presentation can

- Motivate consideration of different designs
  - Ways of eliminating data exchanges
- Explain why certain evidence is critically important
  - Code reviews showing that data validation checks are done
  - Test results showing invalid data is rejected
- Help prioritize activities that contribute to the argument



# When to Visualize the Argument?

---

Q: At what stage in a project is it worth visualizing 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



# Overview

---

Introduction to Assurance Cases

Hazard Analyses and Assurance Cases

**The SEI Assurance Cases for Medical Devices Project**

Conclusions



# Current Reality

---

## More software in medical devices

- More software-related problems
- Privacy, safety, and regulatory concerns
- Plug-and-play issues

Certification based on process quality vs. product insight



# Purpose and Goals of the Project

---

Lay the basis for the use of assurance cases to ensure the safety of medical devices.

- Relevant arguments and evidence for device approval
- Criteria for reviewing the soundness of a medical device assurance case
- Methods for discovering and constraining emergent, potentially unsafe (or insecure) properties in a PnP environment
- Needed tool support

Benefits to the medical device community and the FDA and ultimately to providers and patients



# Approach

---

Develop an assurance case for a generic infusion pump (GIP)

- A research vehicle created by Prof. Insup Lee at University of Pennsylvania
- Realistic but not tied to a proprietary design

Work with at least one medical device manufacturer and the FDA to ensure that we are addressing their needs

Determine what information the FDA needs to see and how that information can be presented in an assurance case



# Project Team & Collaborators

---

**John B. Goodenough**

**Charles B. Weinstock**

**Paul L. Jones, FDA**

**Sherman Eagles, Medtronic, Inc.**

**Insup Lee, University of Pennsylvania**



# Example Requirements

---

## 2. User Interface

### 2.1. Resistance to tampering and accidents

- 2.1.1. To avoid accidental tampering of the pump's settings such as the flow rate/VTBI, at least two steps should be required to change the settings.
- 2.1.2. Changing settings, such as the patient's weight or infusion duration, while the pump is infusing, should either not be allowed, or at least require confirmation.
- 2.1.3. The administration set should be designed to prevent compromising patient safety or cause an unacceptable flow error.
- 2.1.4. There shall be no multiple-key legal values. That is, there should be no legal inputs that require multiple keys to be pressed simultaneously.
- 2.1.5. If the numeric keypad cover is broken or unlocked during infusion, the pump should issue an alarm to indicate illegal tampering.

### 2.2. User input

- 2.2.1. If the pump is in a state where user input is required, the pump shall issue periodic alerts/indications every *15 minutes* till the required input is provided.
- 2.2.2. The pump shall issue an alert if paused for more than *x* minutes
- 2.2.3. Clearing of the pump settings and resetting of the pump shall require confirmation.

Source: Safety Requirements for the Generic Infusion Pump (University of Pennsylvania)



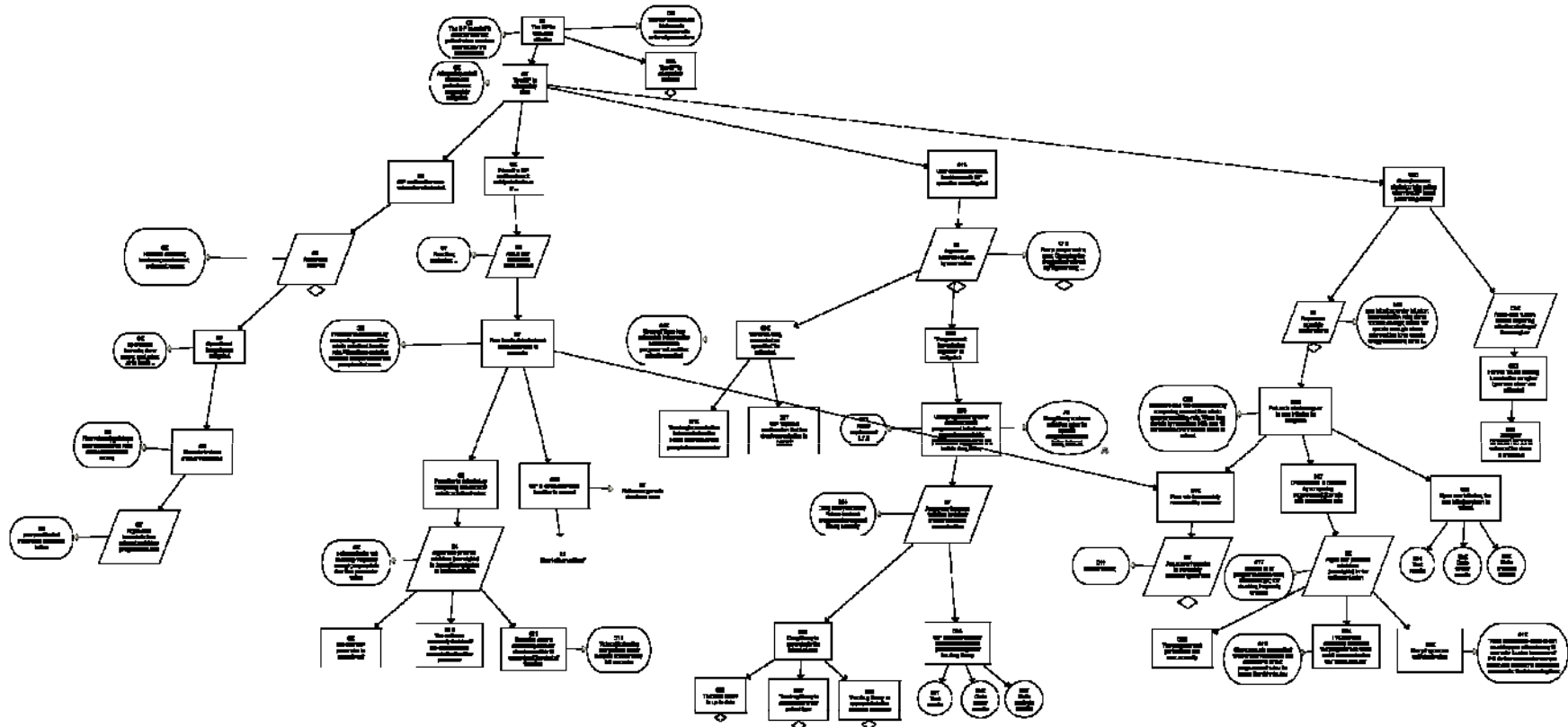
# Example Hazard Analysis

HID	Hazard	Pump Type	Cause	Action	Mitigated by	Safety Requirement
2. Environmental Hazards						
2.7	Tampering	FRN	Patient tampers with pump settings without authorization			2.1
2.8	Tampering	FRN	Panel lock broken or opened during infusion	Alert(); Stop()		2.1, 3.3
2.9	Tampering	FRN	Panel/door opened during infusion; Infusion started when door open	Alert(); Log()		2.1, 3.3
8. Use Hazards						
8.6	Rule-based failure	FRN	Incorrect prescription given to patient; Incorrect drug library loaded		Barcode scanner	2.2, 5.1.8
8.7	Overinfusion	FRN	User / Patient change infusion settings inadvertently		Drug library	2.2, 5.1
8.8	Underinfusion	FRN	User / Patient change infusion settings inadvertently		Drug library	2.2, 5.1

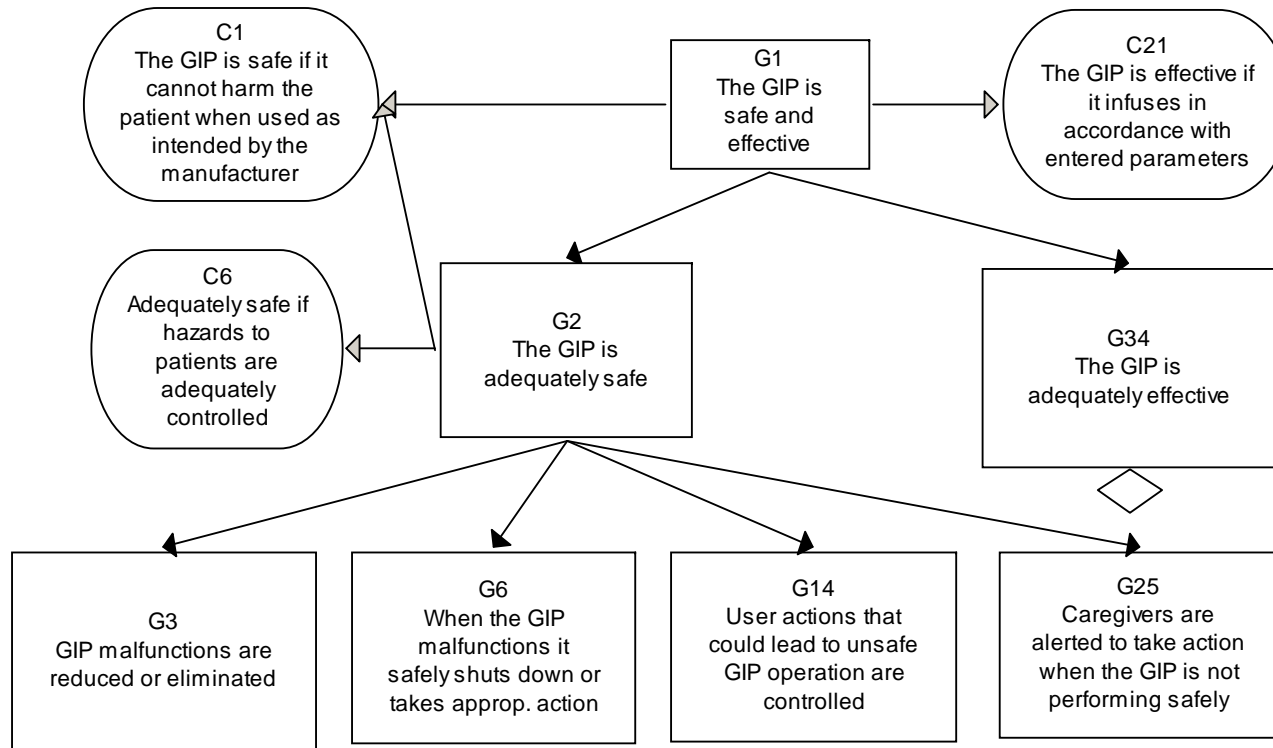
Source: The Generic Infusion Pump (University of Pennsylvania)



# Birds Eye View



# High Level Overview



# Overview

---

Introduction to Assurance Cases

Hazard Analyses and Assurance Cases

The SEI Assurance Cases for Medical Devices Project

**Conclusions**



# Final Thoughts

---

Testing by itself is inadequate for reaching valid conclusions about SW reliability in complex systems

## Assurance case

- Integrates design analyses focused on SW hazards and FMEA
- Is reviewable

Assurance case patterns hold promise of capturing valid arguments and guiding reliability improvement efforts



# Contact Information

---

## **Charles B. Weinstock**

Senior MTS

Performance Critical Systems

Telephone: +1 412-268-7719

Email: [weinstock@sei.cmu.edu](mailto:weinstock@sei.cmu.edu)

## **U.S. mail:**

Software Engineering Institute

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

