Towards Formal Assurance Case Framework in Agda

2014-10-15 @ IOC, Tallinn

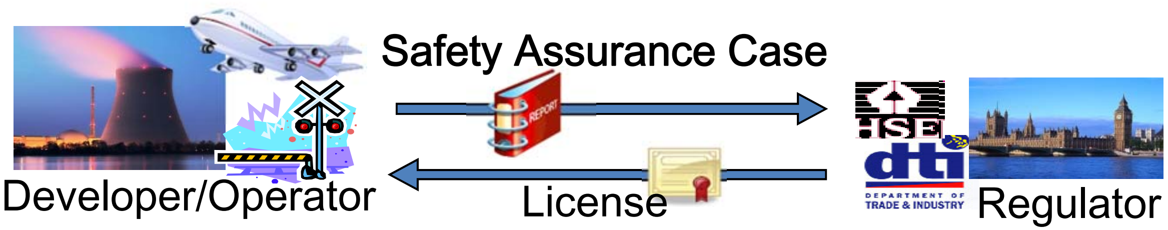
Makoto Takeyama

Kanagawa University

(research funded by IPA RISE initiative)

Assurance case

- Assurance case:
  a documented, explicit argumentation to demonstrate a specified system is “okay” in a specified sense.

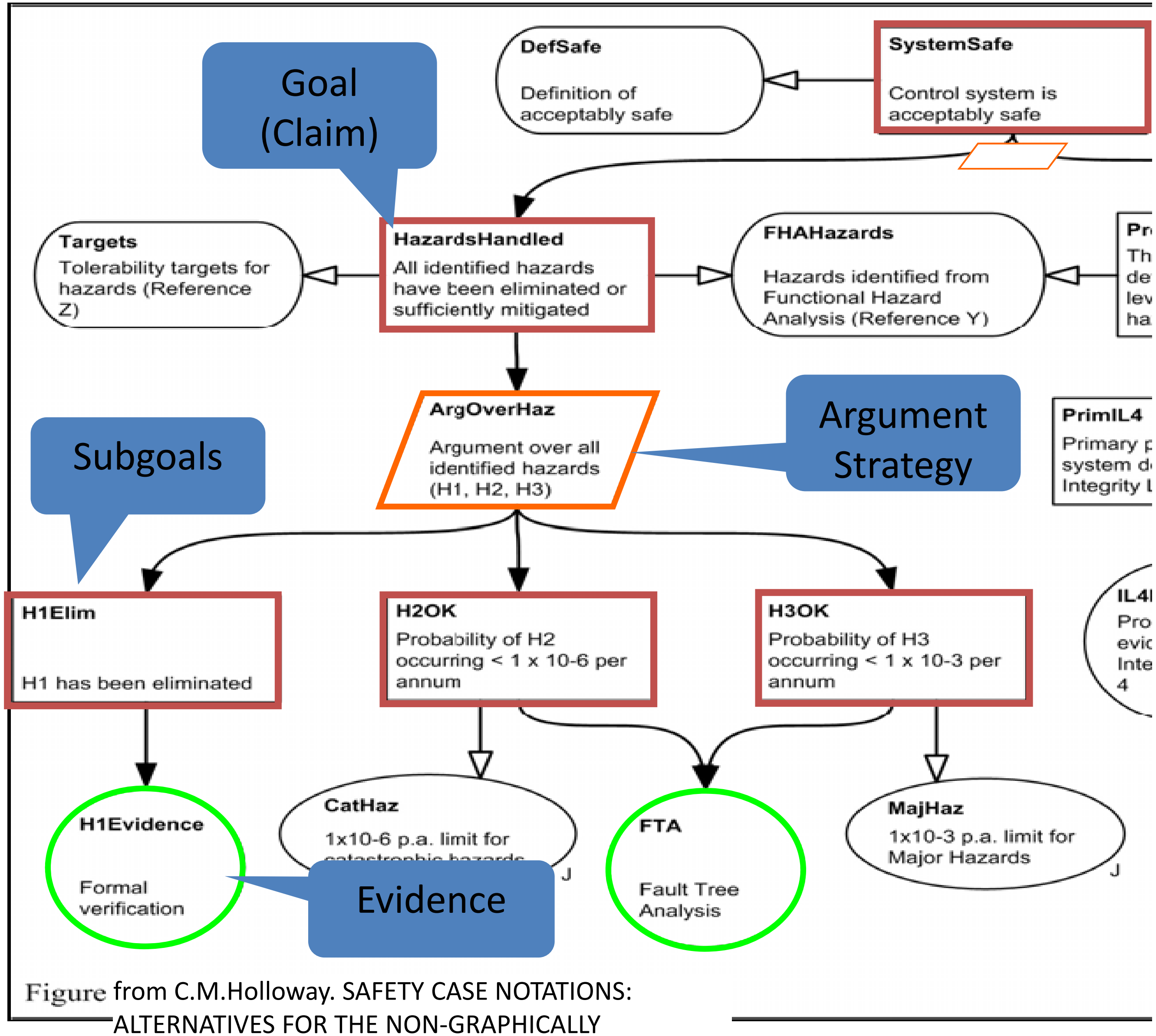


- Origin in Safety regulation regime:
  nuclear power plant, offshore oil platform, aviation, railway, …

- Spreading in other context (acquisition, certification, ···)
  to assure other qualities like
  reliability and maintainability, security, *dependability*, ...

- Assurance Case is a set of auditable claims, arguments and evidence created to support the claim that a defined system/service will satisfy the particular requirements. (OMG SACM 1.0)

Current practice: Structured Argument in Graphical Notation



Goal (Claim)

DefSafe

Definition of acceptably safe

SystemSafe

Control system is acceptably safe

Targets

Tolerability targets for hazards (Reference Z)

HazardsHandled

All identified hazards have been eliminated or sufficiently mitigated

FHAHazards

Hazards identified from Functional Hazard Analysis (Reference Y)

Pr

Th de lev ha

ArgOverHaz

Argument over all identified hazards (H1, H2, H3)

Argument Strategy

PrimIL4

Primary p system d Integrity l

Subgoals

H1Elim

H1 has been eliminated

H2OK

Probability of H2 occurring < 1 x 10-6 per annum

H3OK

Probability of H3 occurring < 1 x 10-3 per annum

IL4I

Pro evic Inte 4

H1Evidence

Formal verification

CatHaz

1x10-6 p.a. limit for catastrophic hazards

J

FTA

Fault Tree Analysis

MajHaz

1x10-3 p.a. limit for Major Hazards

J

Evidence

Figure from C.M.Holloway. SAFETY CASE NOTATIONS: ALTERNATIVES FOR THE NON-GRAPHICALLY INCLINED?, 2008

- GSN (Kelly, 1998), CAE (Adelard)
- Argument elements explicitly identified and linked.
- Goals decomposed by strategies into subgoals until direct evidence become available.

Current practice: Structured Argument in Graphical Notation

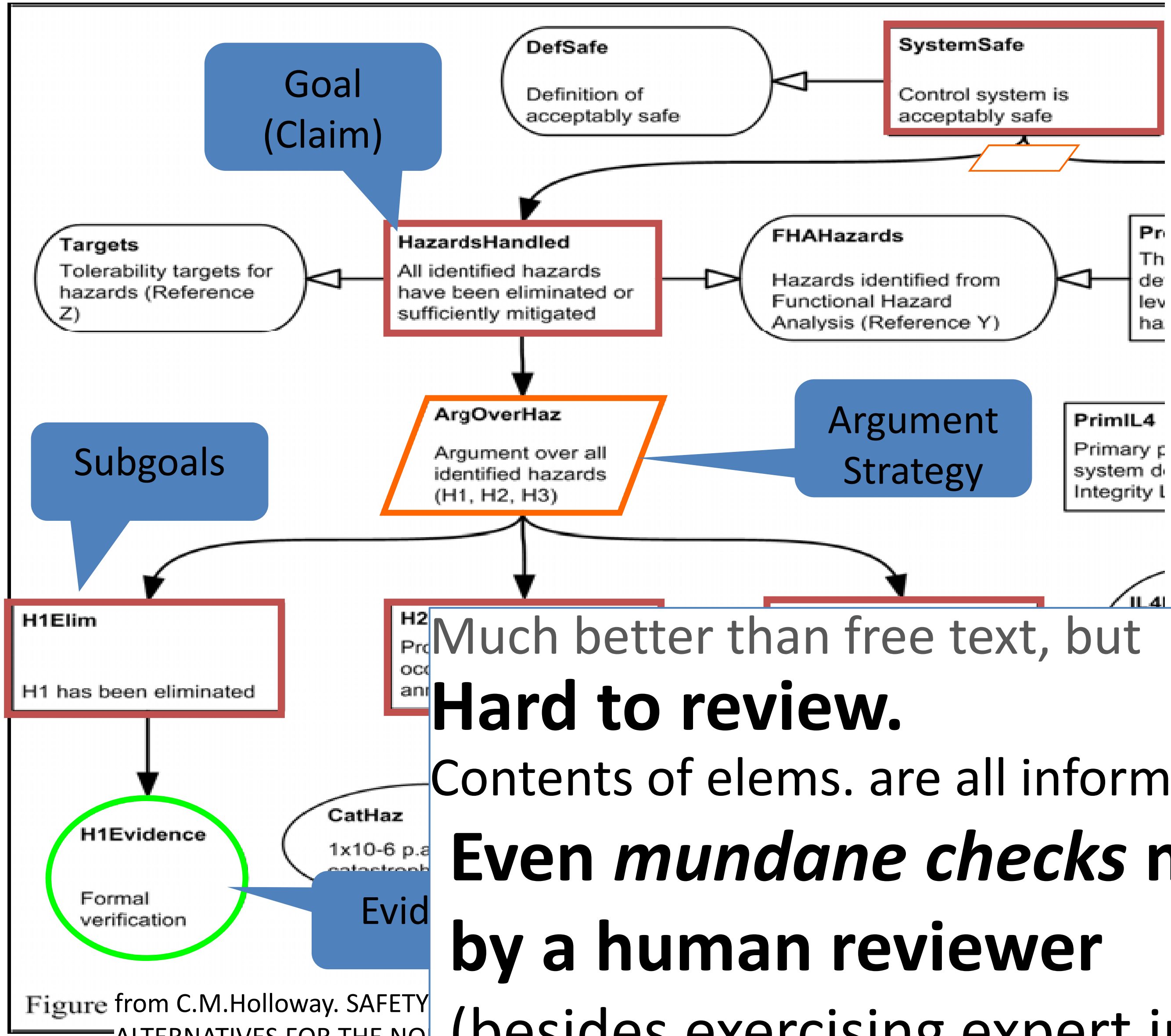

Goal (Claim)
DefSafe
Definition of acceptably safe
SystemSafe
Control system is acceptably safe
Targets
Tolerability targets for hazards (Reference Z)
HazardsHandled
All identified hazards have been eliminated or sufficiently mitigated
FHAHazards
Hazards identified from Functional Hazard Analysis (Reference Y)
Pr
Th
de
lev
ha
ArgOverHaz
Argument over all identified hazards (H1, H2, H3)
Argument Strategy
PrimIL4
Primary p
system d
Integrity L
Subgoals
IL 4
H1Elim
H1 has been eliminated
H2
Pr
oc
an
H1Evidence
Formal verification
CatHaz
1x10-6 p.a
catastroph
Evid

Figure from C.M.Holloway. SAFETY ALTERNATIVES FOR THE NO INCLINED?, 2008

- GSN (Kelly, 1998) CAE (Adelard)
- Argument elements explicitly identified and linked.
- Goals decomposed by strategies into subgoals until direct evidence become available.

Much better than free text, but

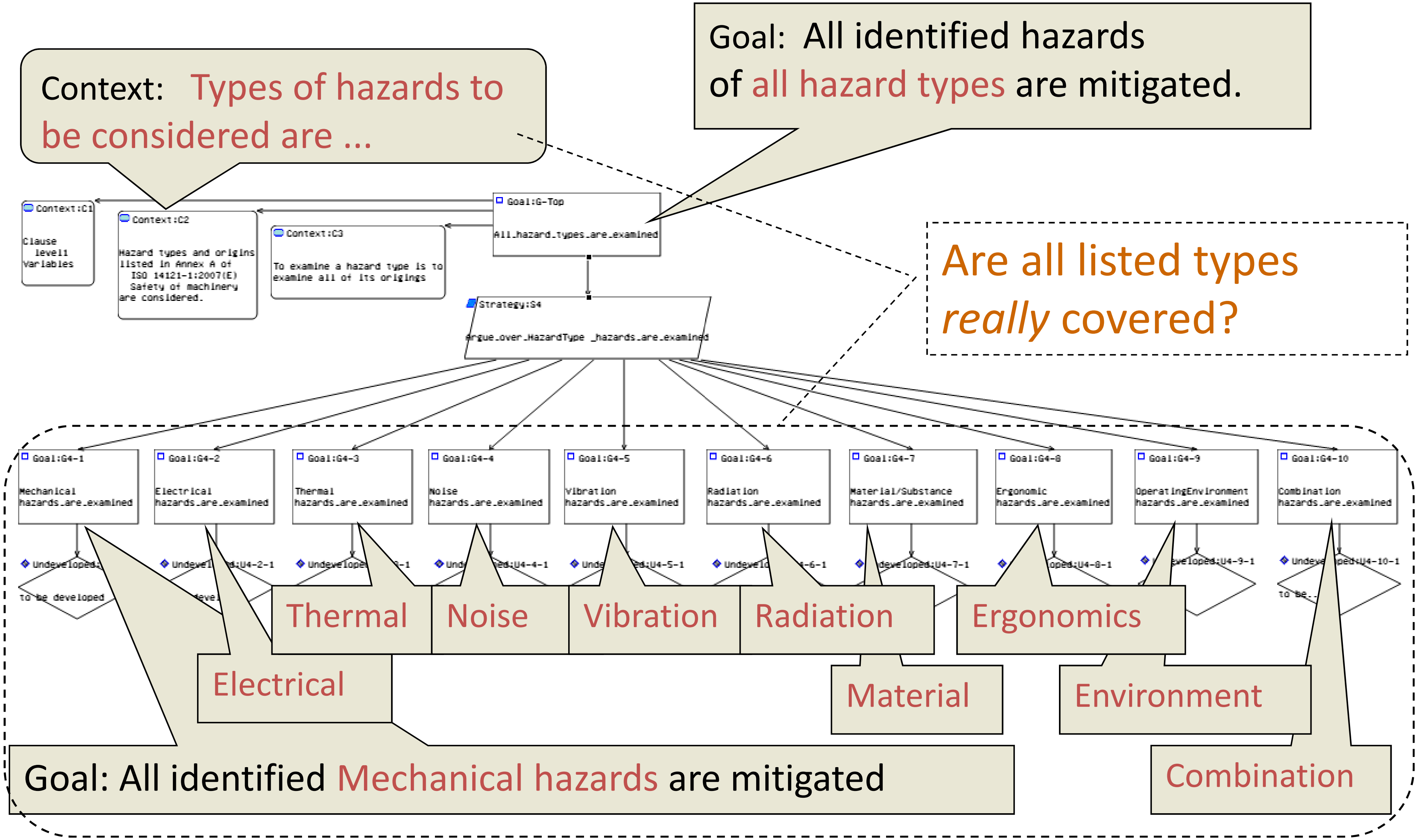
Hard to review.

Contents of elems. are all informal, so

Even *mundane checks* must be done by a human reviewer

(besides exercising expert judgment)

A very mundane checking


Context:C1
Clause level1 Variables
Context:C2
Hazard types and origins listed in Annex A of ISO 14121-1:2007(E) Safety of machinery are considered.
Context:C3
To examine a hazard type is to examine all of its origings
Goal:G-Top
All_hazard_types_are_examined
Strategy:S4
argue_over_HazardType _hazards_are_examined
Goal:G4-1
Mechanical hazards_are_examined
Goal:G4-2
Electrical hazards_are_examined
Goal:G4-3
Thermal hazards_are_examined
Goal:G4-4
Noise hazards_are_examined
Goal:G4-5
Vibration hazards_are_examined
Goal:G4-6
Radiation hazards_are_examined
Goal:G4-7
Material/Substance hazards_are_examined
Goal:G4-8
Ergonomic hazards_are_examined
Goal:G4-9
OperatingEnvironment hazards_are_examined
Goal:G4-10
Combination hazards_are_examined
Undeveloped:
to be developed
:U4-2-1
:U4-4-1
:U4-5-1
-6-1
developed:U4-7-1
oped:U4-8-1
eveloped:U4-9-1
Undeve
ped:U4-10-1
To be..

Context: Types of hazards to be considered are ...

Goal: All identified hazards of all hazard types are mitigated.

Are all listed types *really* covered?

Thermal

Noise

Vibration

Radiation

Ergonomics

Electrical

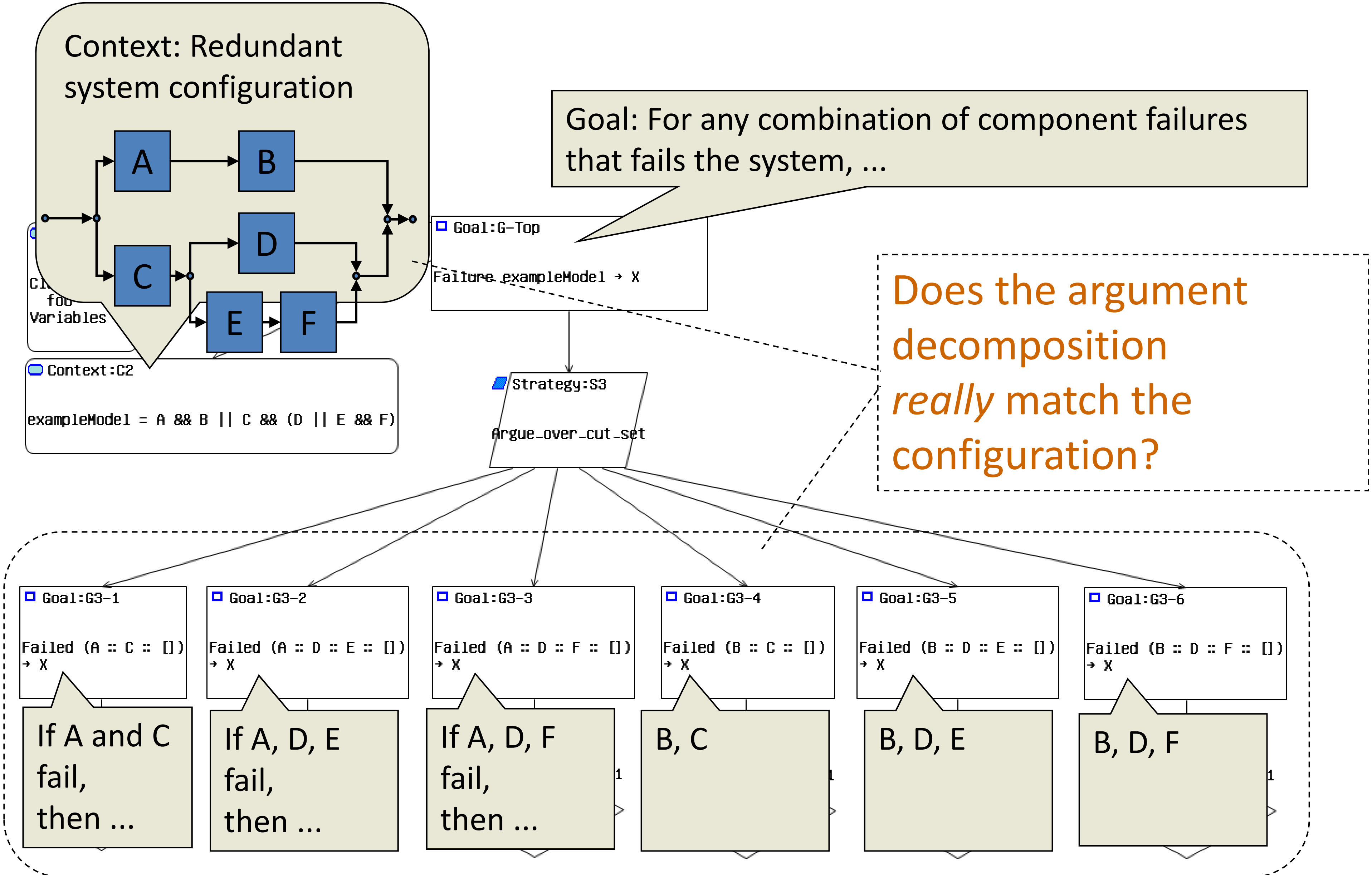
Material

Environment

Goal: All identified Mechanical hazards are mitigated

Combination

a not-so-mundane but mechanical checking



Context: Redundant system configuration

Goal: For any combination of component failures that fails the system, ...

Goal:G-Top

Failure exampleModel → X

Context:C2

exampleModel = A && B || C && (D || E && F)

Strategy:S3

Argue_over_cut_set

Does the argument decomposition *really* match the configuration?

Goal:G3-1

Failed (A :: C :: []) → X

If A and C fail, then ...

Goal:G3-2

Failed (A :: D :: E :: []) → X

If A, D, E fail, then ...

Goal:G3-3

Failed (A :: D :: F :: []) → X

If A, D, F fail, then ...

Goal:G3-4

Failed (B :: C :: []) → X

B, C

Goal:G3-5

Failed (B :: D :: E :: []) → X

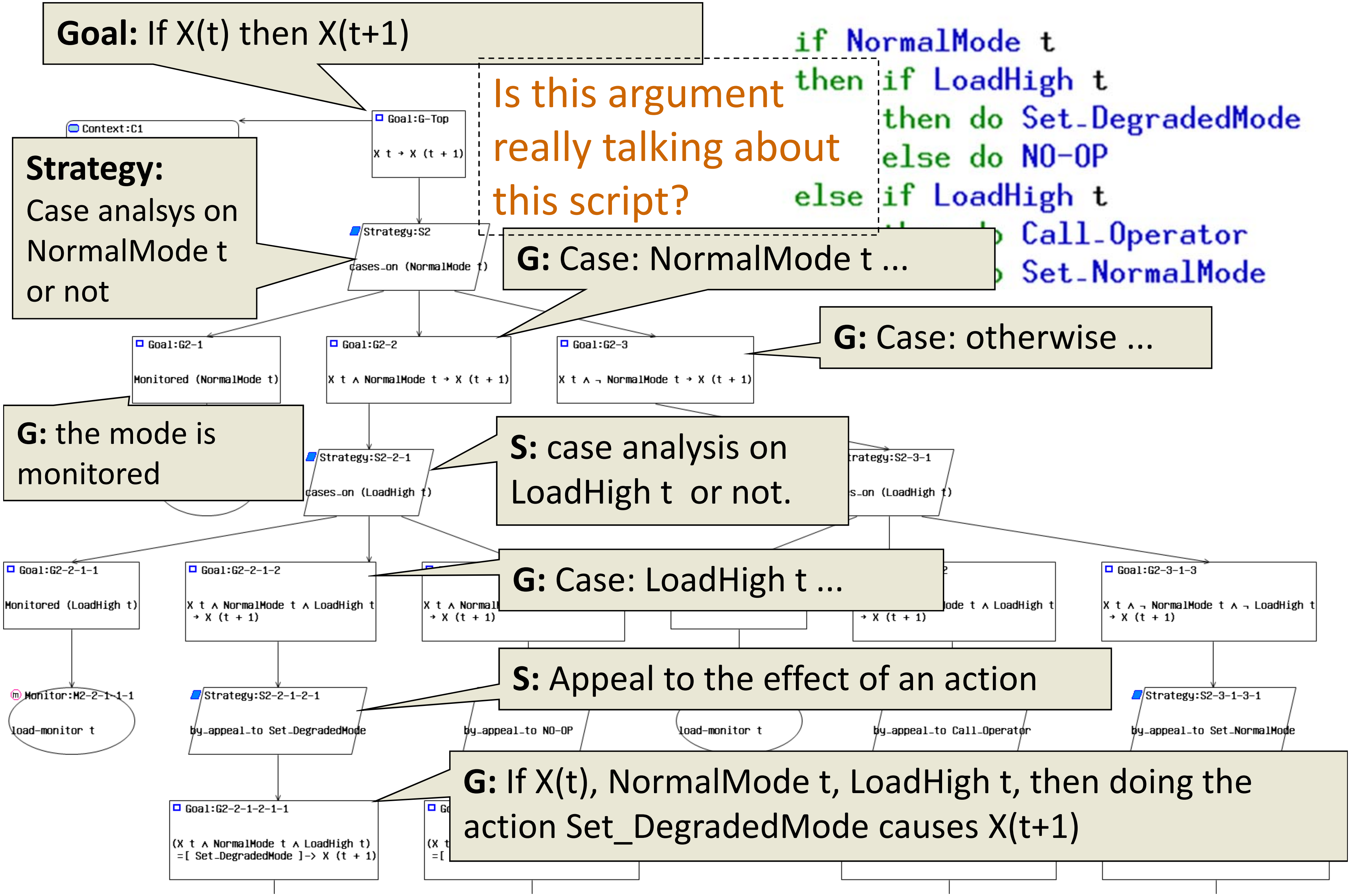
B, D, E

Goal:G3-6

Failed (B :: D :: F :: []) → X

B, D, F

a not-so-mundane but mechanical checking



```
if NormalMode t
then if LoadHigh t
     then do Set_DegradedMode
     else do NO-OP
else if LoadHigh t
            Call_Operator
            Set_NormalMode
```

Goal: If X(t) then X(t+1)

Is this argument really talking about this script?

Strategy: Case analsys on NormalMode t or not

G: Case: NormalMode t ...

G: Case: otherwise ...

G: the mode is monitored

S: case analysis on LoadHigh t or not.

G: Case: LoadHigh t ...

S: Appeal to the effect of an action

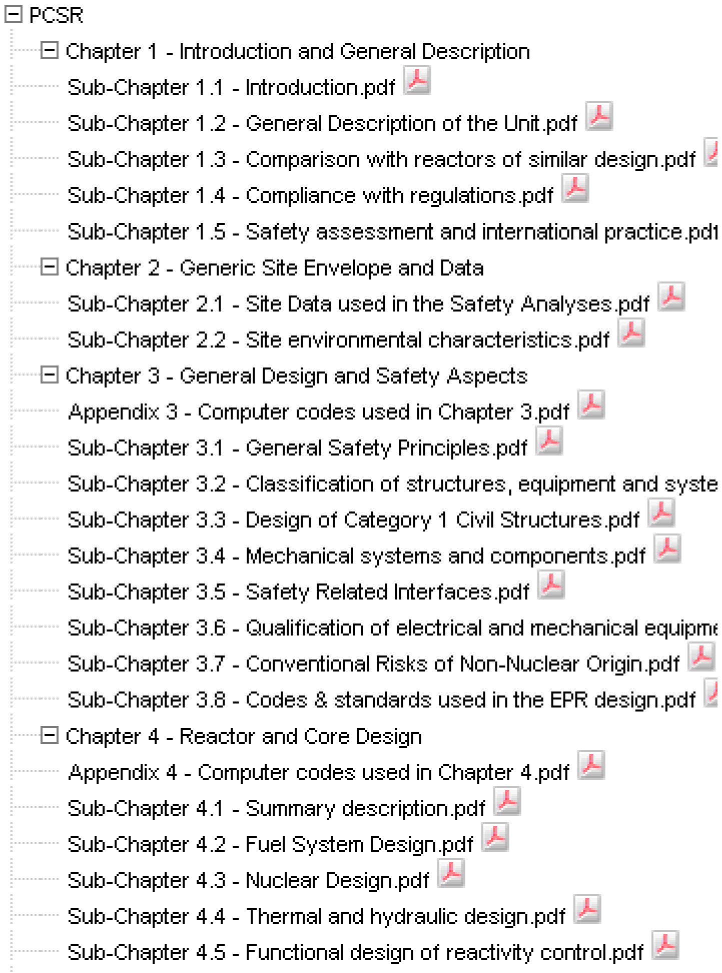
G: If X(t), NormalMode t, LoadHigh t, then doing the action Set_DegradedMode causes X(t+1)

The sheer size is also an issue

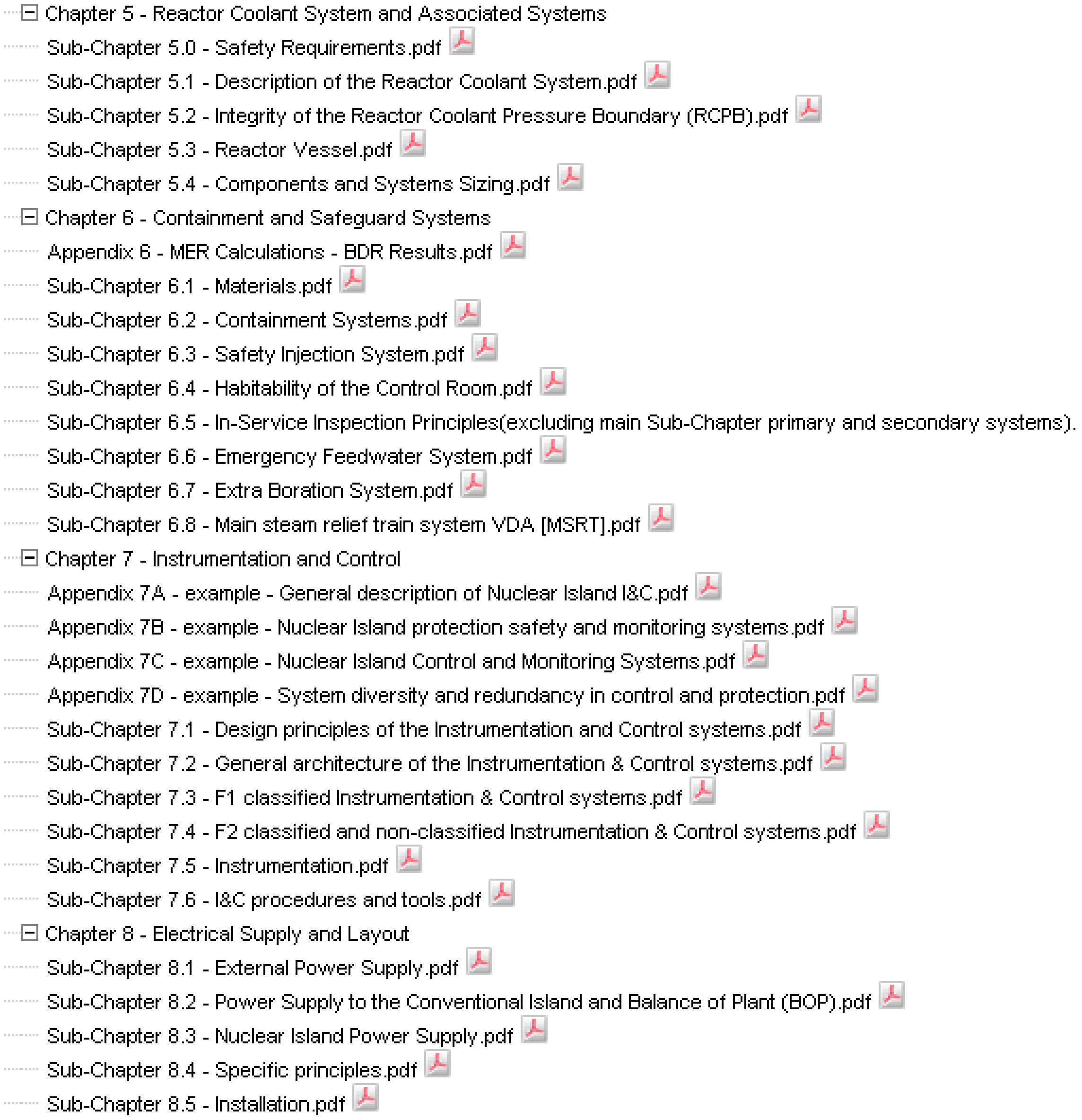
A nuclear reactor design's "Preconstruction Safety Report" in English

http://www.epr-reactor.co.uk/scripts/ssmod/publigen/content/templates/show.asp?P=290&L=EN&ID_CAT=1.2

Each 📄 is 10's~100's pages



- ⊟ PCSR
  - ⊟ Chapter 1 - Introduction and General Description
    - Sub-Chapter 1.1 - Introduction.pdf
    - Sub-Chapter 1.2 - General Description of the Unit.pdf
    - Sub-Chapter 1.3 - Comparison with reactors of similar design.pdf
    - Sub-Chapter 1.4 - Compliance with regulations.pdf
    - Sub-Chapter 1.5 - Safety assessment and international practice.pdf
  - ⊟ Chapter 2 - Generic Site Envelope and Data
    - Sub-Chapter 2.1 - Site Data used in the Safety Analyses.pdf
    - Sub-Chapter 2.2 - Site environmental characteristics.pdf
  - ⊟ Chapter 3 - General Design and Safety Aspects
    - Appendix 3 - Computer codes used in Chapter 3.pdf
    - Sub-Chapter 3.1 - General Safety Principles.pdf
    - Sub-Chapter 3.2 - Classification of structures, equipment and syste
    - Sub-Chapter 3.3 - Design of Category 1 Civil Structures.pdf
    - Sub-Chapter 3.4 - Mechanical systems and components.pdf
    - Sub-Chapter 3.5 - Safety Related Interfaces.pdf
    - Sub-Chapter 3.6 - Qualification of electrical and mechanical equipme
    - Sub-Chapter 3.7 - Conventional Risks of Non-Nuclear Origin.pdf
    - Sub-Chapter 3.8 - Codes & standards used in the EPR design.pdf
  - ⊟ Chapter 4 - Reactor and Core Design
    - Appendix 4 - Computer codes used in Chapter 4.pdf
    - Sub-Chapter 4.1 - Summary description.pdf
    - Sub-Chapter 4.2 - Fuel System Design.pdf
    - Sub-Chapter 4.3 - Nuclear Design.pdf
    - Sub-Chapter 4.4 - Thermal and hydraulic design.pdf
    - Sub-Chapter 4.5 - Functional design of reactivity control.pdf

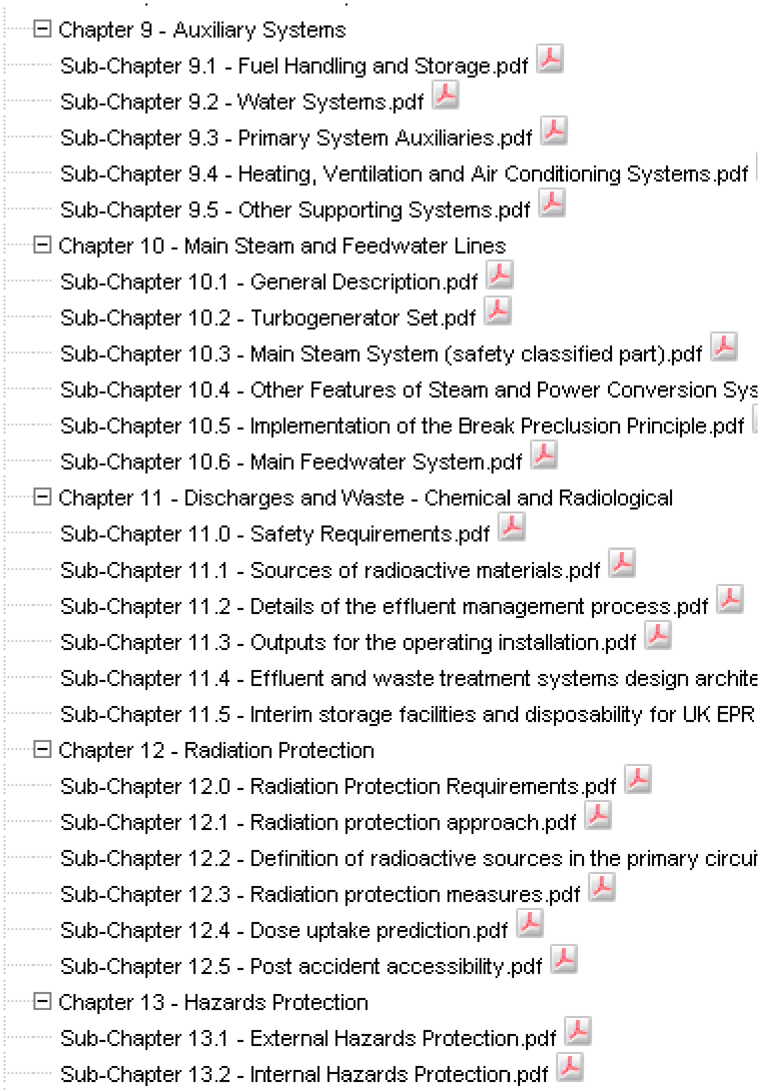


- ⊟ Chapter 5 - Reactor Coolant System and Associated Systems
  - Sub-Chapter 5.0 - Safety Requirements.pdf
  - Sub-Chapter 5.1 - Description of the Reactor Coolant System.pdf
  - Sub-Chapter 5.2 - Integrity of the Reactor Coolant Pressure Boundary (RCPB).pdf
  - Sub-Chapter 5.3 - Reactor Vessel.pdf
  - Sub-Chapter 5.4 - Components and Systems Sizing.pdf
- ⊟ Chapter 6 - Containment and Safeguard Systems
  - Appendix 6 - MER Calculations - BDR Results.pdf
  - Sub-Chapter 6.1 - Materials.pdf
  - Sub-Chapter 6.2 - Containment Systems.pdf
  - Sub-Chapter 6.3 - Safety Injection System.pdf
  - Sub-Chapter 6.4 - Habitability of the Control Room.pdf
  - Sub-Chapter 6.5 - In-Service Inspection Principles(excluding main Sub-Chapter primary and secondary systems).
  - Sub-Chapter 6.6 - Emergency Feedwater System.pdf
  - Sub-Chapter 6.7 - Extra Boration System.pdf
  - Sub-Chapter 6.8 - Main steam relief train system VDA [MSRT].pdf
- ⊟ Chapter 7 - Instrumentation and Control
  - Appendix 7A - example - General description of Nuclear Island I&C.pdf
  - Appendix 7B - example - Nuclear Island protection safety and monitoring systems.pdf
  - Appendix 7C - example - Nuclear Island Control and Monitoring Systems.pdf
  - Appendix 7D - example - System diversity and redundancy in control and protection.pdf
  - Sub-Chapter 7.1 - Design principles of the Instrumentation and Control systems.pdf
  - Sub-Chapter 7.2 - General architecture of the Instrumentation & Control systems.pdf
  - Sub-Chapter 7.3 - F1 classified Instrumentation & Control systems.pdf
  - Sub-Chapter 7.4 - F2 classified and non-classified Instrumentation & Control systems.pdf
  - Sub-Chapter 7.5 - Instrumentation.pdf
  - Sub-Chapter 7.6 - I&C procedures and tools.pdf
- ⊟ Chapter 8 - Electrical Supply and Layout
  - Sub-Chapter 8.1 - External Power Supply.pdf
  - Sub-Chapter 8.2 - Power Supply to the Conventional Island and Balance of Plant (BOP).pdf
  - Sub-Chapter 8.3 - Nuclear Island Power Supply.pdf
  - Sub-Chapter 8.4 - Specific principles.pdf
  - Sub-Chapter 8.5 - Installation.pdf

A nuclear reactor design's "Preconstruction Safety Report" in English

http://www.epr-reactor.co.uk/scripts/ssmod/publigen/content/templates/show.asp?P=290&L=EN&ID_CAT=1.2

Each 📄 is 10's~100's pages



⊟ Chapter 9 - Auxiliary Systems

- Sub-Chapter 9.1 - Fuel Handling and Storage.pdf
- Sub-Chapter 9.2 - Water Systems.pdf
- Sub-Chapter 9.3 - Primary System Auxiliaries.pdf
- Sub-Chapter 9.4 - Heating, Ventilation and Air Conditioning Systems.pdf
- Sub-Chapter 9.5 - Other Supporting Systems.pdf

⊟ Chapter 10 - Main Steam and Feedwater Lines

- Sub-Chapter 10.1 - General Description.pdf
- Sub-Chapter 10.2 - Turbogenerator Set.pdf
- Sub-Chapter 10.3 - Main Steam System (safety classified part).pdf
- Sub-Chapter 10.4 - Other Features of Steam and Power Conversion Sys
- Sub-Chapter 10.5 - Implementation of the Break Preclusion Principle.pdf
- Sub-Chapter 10.6 - Main Feedwater System.pdf

⊟ Chapter 11 - Discharges and Waste - Chemical and Radiological

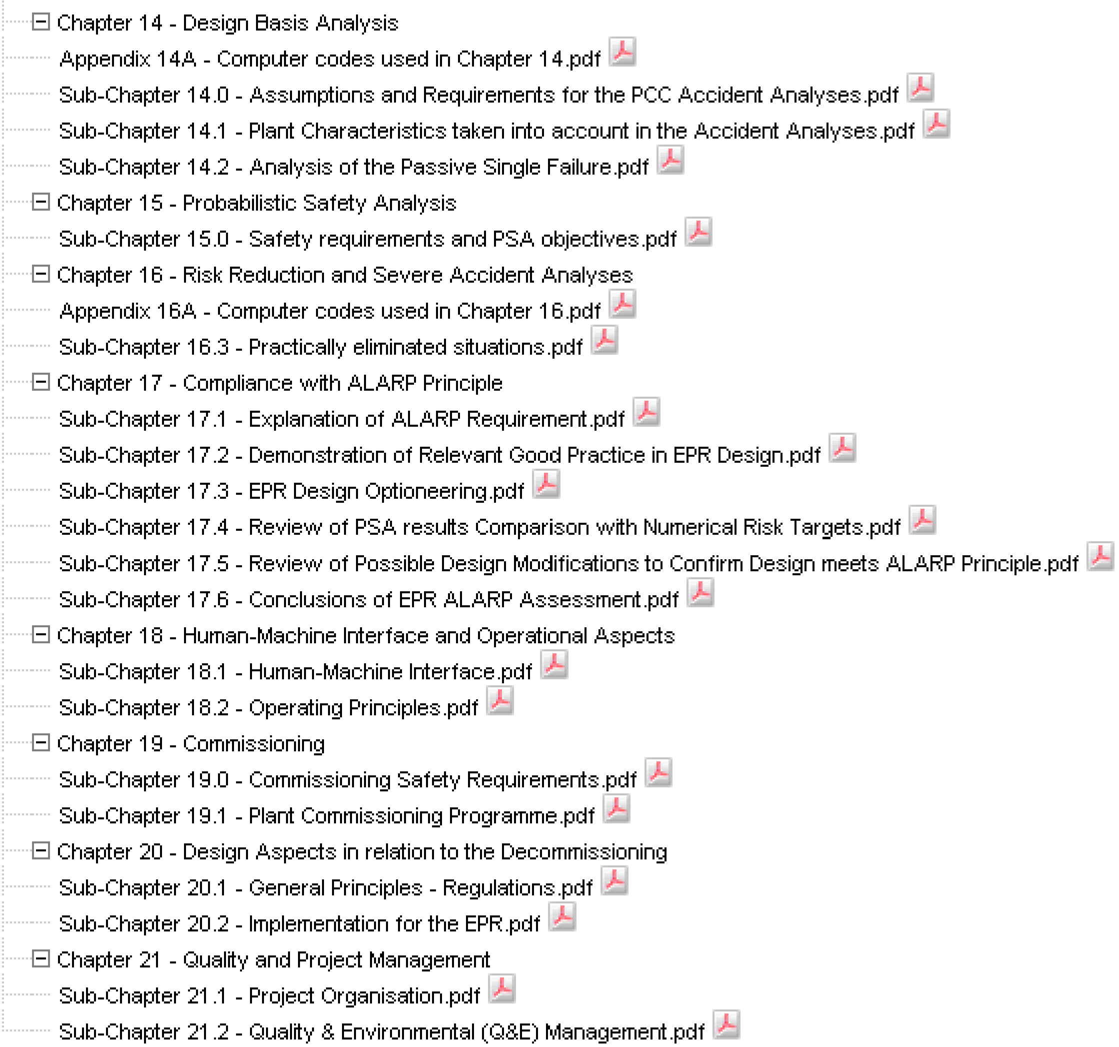
- Sub-Chapter 11.0 - Safety Requirements.pdf
- Sub-Chapter 11.1 - Sources of radioactive materials.pdf
- Sub-Chapter 11.2 - Details of the effluent management process.pdf
- Sub-Chapter 11.3 - Outputs for the operating installation.pdf
- Sub-Chapter 11.4 - Effluent and waste treatment systems design archite
- Sub-Chapter 11.5 - Interim storage facilities and disposability for UK EPR

⊟ Chapter 12 - Radiation Protection

- Sub-Chapter 12.0 - Radiation Protection Requirements.pdf
- Sub-Chapter 12.1 - Radiation protection approach.pdf
- Sub-Chapter 12.2 - Definition of radioactive sources in the primary circui
- Sub-Chapter 12.3 - Radiation protection measures.pdf
- Sub-Chapter 12.4 - Dose uptake prediction.pdf
- Sub-Chapter 12.5 - Post accident accessibility.pdf

⊟ Chapter 13 - Hazards Protection

- Sub-Chapter 13.1 - External Hazards Protection.pdf
- Sub-Chapter 13.2 - Internal Hazards Protection.pdf



⊟ Chapter 14 - Design Basis Analysis

- Appendix 14A - Computer codes used in Chapter 14.pdf
- Sub-Chapter 14.0 - Assumptions and Requirements for the PCC Accident Analyses.pdf
- Sub-Chapter 14.1 - Plant Characteristics taken into account in the Accident Analyses.pdf
- Sub-Chapter 14.2 - Analysis of the Passive Single Failure.pdf

⊟ Chapter 15 - Probabilistic Safety Analysis

- Sub-Chapter 15.0 - Safety requirements and PSA objectives.pdf

⊟ Chapter 16 - Risk Reduction and Severe Accident Analyses

- Appendix 16A - Computer codes used in Chapter 16.pdf
- Sub-Chapter 16.3 - Practically eliminated situations.pdf

⊟ Chapter 17 - Compliance with ALARP Principle

- Sub-Chapter 17.1 - Explanation of ALARP Requirement.pdf
- Sub-Chapter 17.2 - Demonstration of Relevant Good Practice in EPR Design.pdf
- Sub-Chapter 17.3 - EPR Design Optioneering.pdf
- Sub-Chapter 17.4 - Review of PSA results Comparison with Numerical Risk Targets.pdf
- Sub-Chapter 17.5 - Review of Possible Design Modifications to Confirm Design meets ALARP Principle.pdf
- Sub-Chapter 17.6 - Conclusions of EPR ALARP Assessment.pdf

⊟ Chapter 18 - Human-Machine Interface and Operational Aspects

- Sub-Chapter 18.1 - Human-Machine Interface.pdf
- Sub-Chapter 18.2 - Operating Principles.pdf

⊟ Chapter 19 - Commissioning

- Sub-Chapter 19.0 - Commissioning Safety Requirements.pdf
- Sub-Chapter 19.1 - Plant Commissioning Programme.pdf

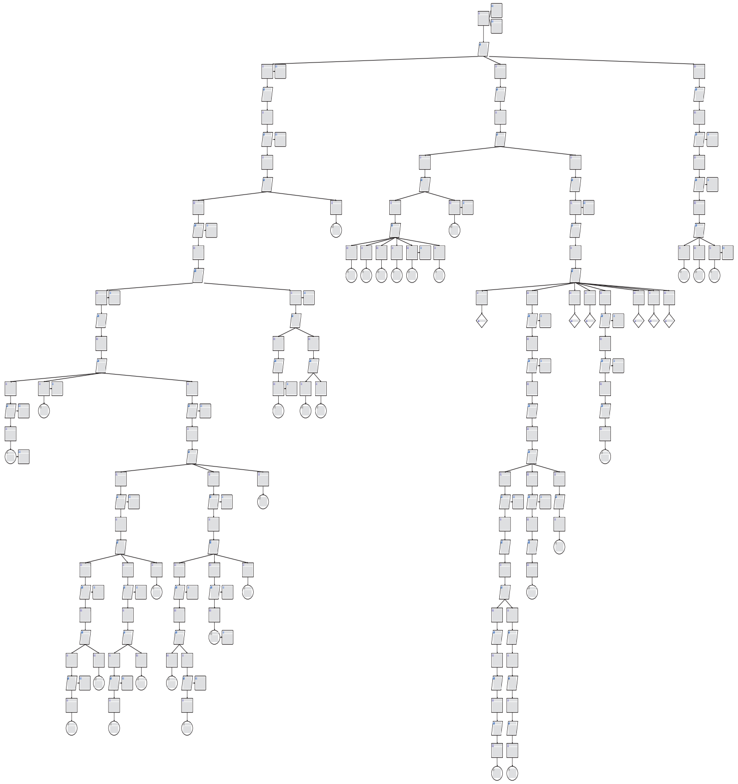
⊟ Chapter 20 - Design Aspects in relation to the Decommissioning

- Sub-Chapter 20.1 - General Principles - Regulations.pdf
- Sub-Chapter 20.2 - Implementation for the EPR.pdf

⊟ Chapter 21 - Quality and Project Management

- Sub-Chapter 21.1 - Project Organisation.pdf
- Sub-Chapter 21.2 - Quality & Environmental (Q&E) Management.pdf

Problem



- Even mundane checking must be done by a human reviewer.
- Issues to check may span across 100s of connected arguments,
- each of which is frequently updated by many hands.

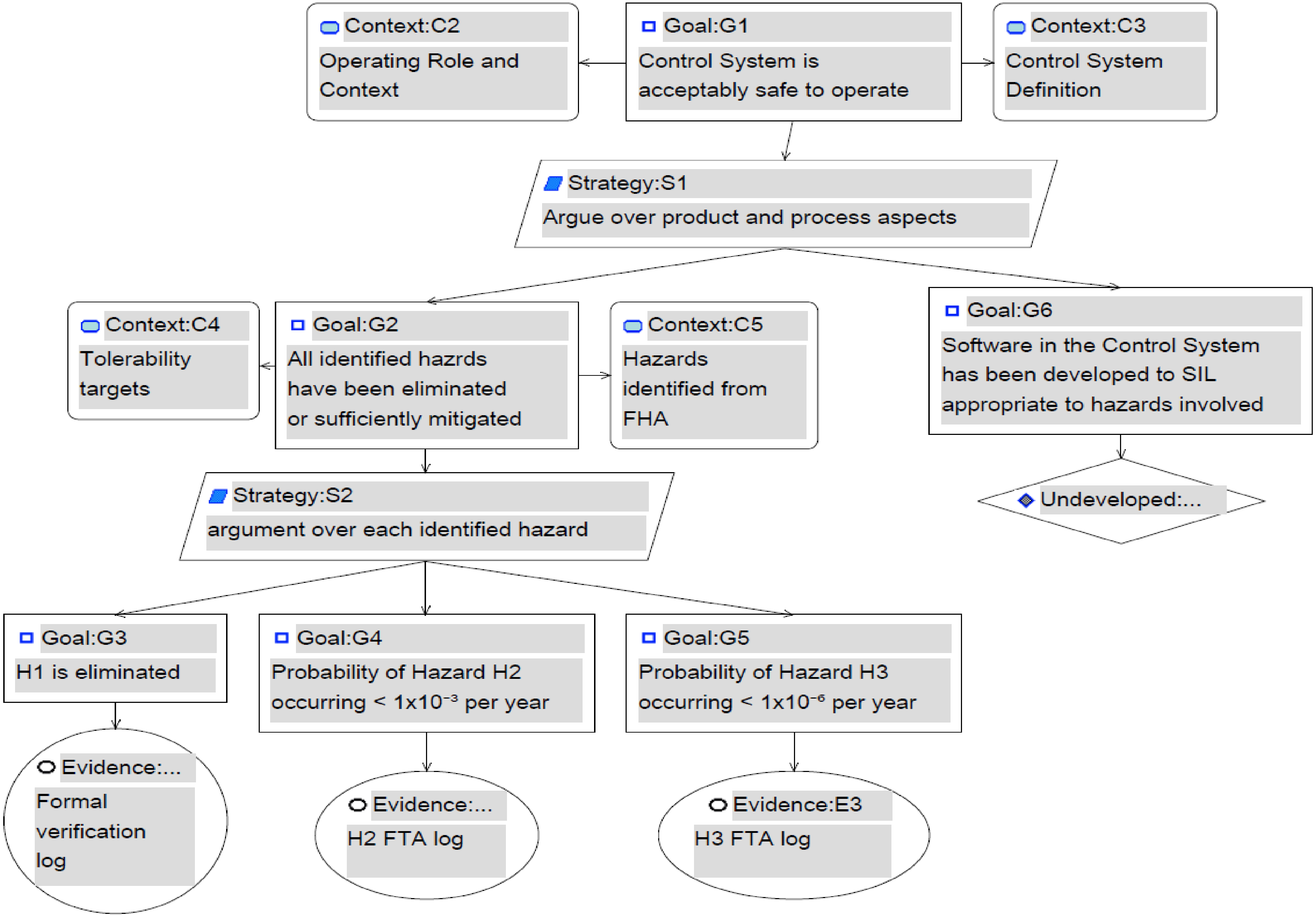
- Let machines check what they can check

  and let reviewers concentrate on exercising expert judgment.

Approach: Formal Assurance Case

- Argument cannot be checked without knowing its basis.
- Human reviewers can gather the relevant **ontology** from their understanding of natural language description.
  - ***what things*** are a system and environment made of?
  - ***what properties*** *and relations* are required of / are assumed / constrain the sys. and env.?
- Often, these concepts are introduced in nat. lang. arguments not by defining them but by just using them.
- **Formal Assurance Case**
  ＝⟨ machine-understandable ontology
  , argument based on that ontology⟩

(A human reviewer knows what to check about this)



Context:C2
Operating Role and Context

Goal:G1
Control System is acceptably safe to operate

Context:C3
Control System Definition

Strategy:S1
Argue over product and process aspects

Context:C4
Tolerability targets

Goal:G2
All identified hazrds have been eliminated or sufficiently mitigated

Context:C5
Hazards identified from FHA

Goal:G6
Software in the Control System has been developed to SIL appropriate to hazards involved

Undeveloped:...

Strategy:S2
argument over each identified hazard

Goal:G3
H1 is eliminated

Goal:G4
Probability of Hazard H2 occurring < 1x10^{-3} per year

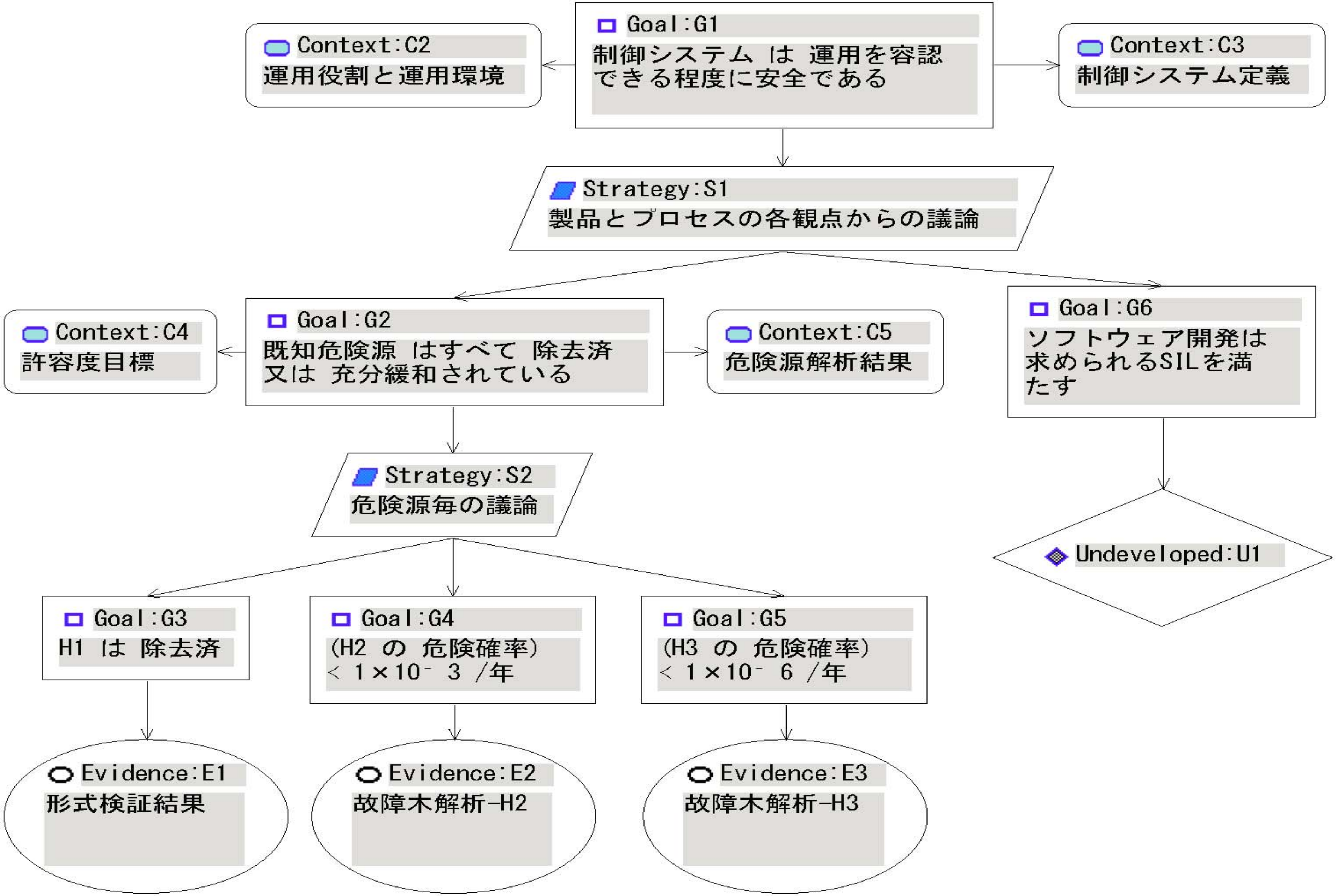
Goal:G5
Probability of Hazard H3 occurring < 1x10^{-6} per year

Evidence:...
Formal verification log

Evidence:...
H2 FTA log

Evidence:E3
H3 FTA log

(A computer feels like looking at this)



Goal:G1
制御システム は 運用を容認できる程度に安全である

Context:C2
運用役割と運用環境

Context:C3
制御システム定義

Strategy:S1
製品とプロセスの各観点からの議論

Goal:G2
既知危険源 はすべて 除去済又は 充分緩和されている

Context:C4
許容度目標

Context:C5
危険源解析結果

Goal:G6
ソフトウェア開発は求められるSILを満たす

Undeveloped:U1

Strategy:S2
危険源毎の議論

Goal:G3
H1 は 除去済

Goal:G4
(H2 の 危険確率) < 1×10－ 3 /年

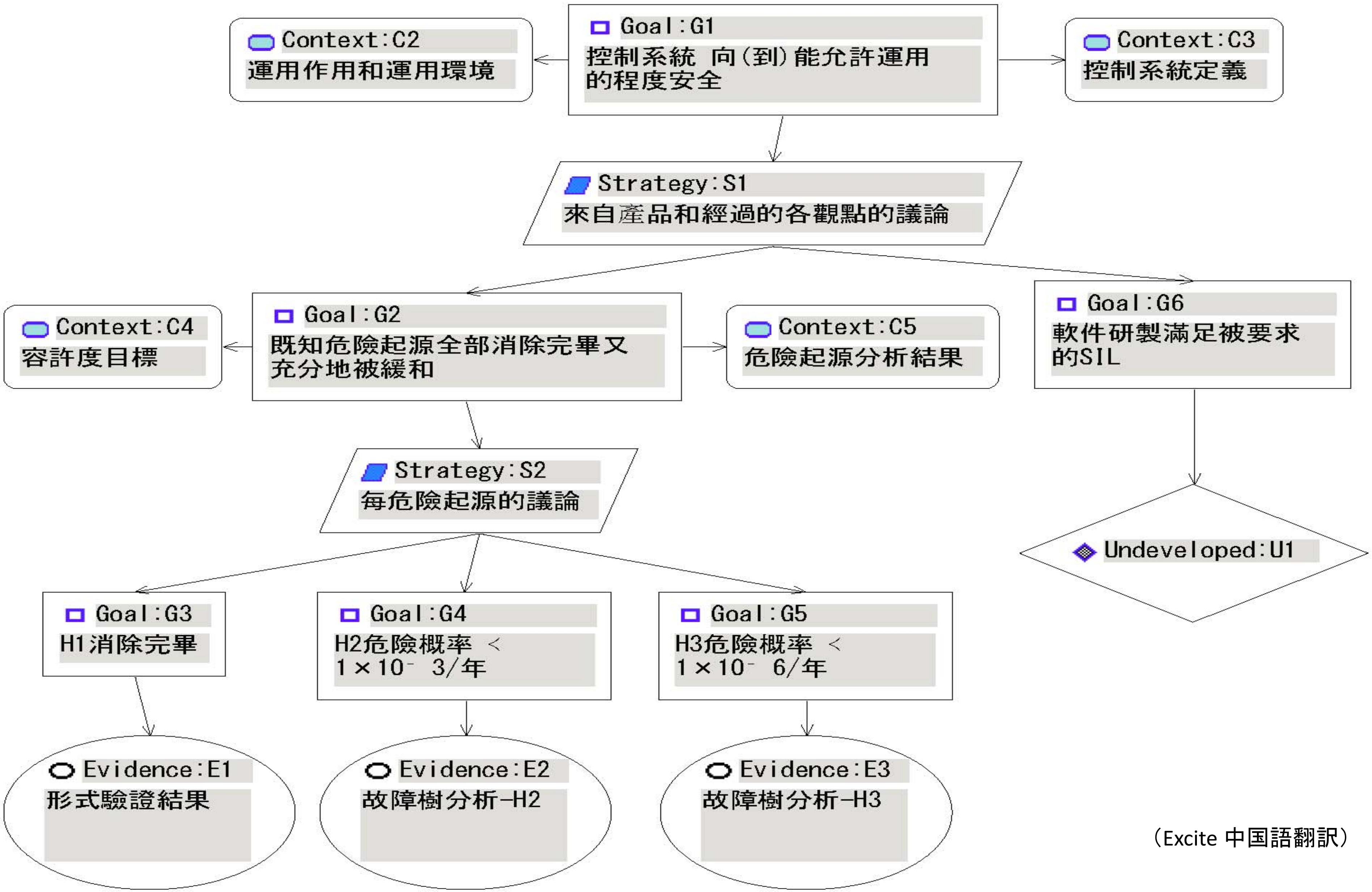
Goal:G5
(H3 の 危険確率) < 1×10－ 6 /年

Evidence:E1
形式検証結果

Evidence:E2
故障木解析-H2

Evidence:E3
故障木解析-H3

(or this)



Goal:G1
控制系統 向(到)能允許運用的程度安全

Context:C2
運用作用和運用環境

Context:C3
控制系統定義

Strategy:S1
來自産品和經過的各觀點的議論

Goal:G2
既知危險起源全部消除完畢又充分地被緩和

Context:C4
容許度目標

Context:C5
危險起源分析結果

Goal:G6
軟件研製滿足被要求的SIL

Strategy:S2
每危險起源的議論

Undeveloped:U1

Goal:G3
H1消除完畢

Goal:G4
H2危險概率 <
1×10－ 3/年

Goal:G5
H3危險概率 <
1×10－ 6/年

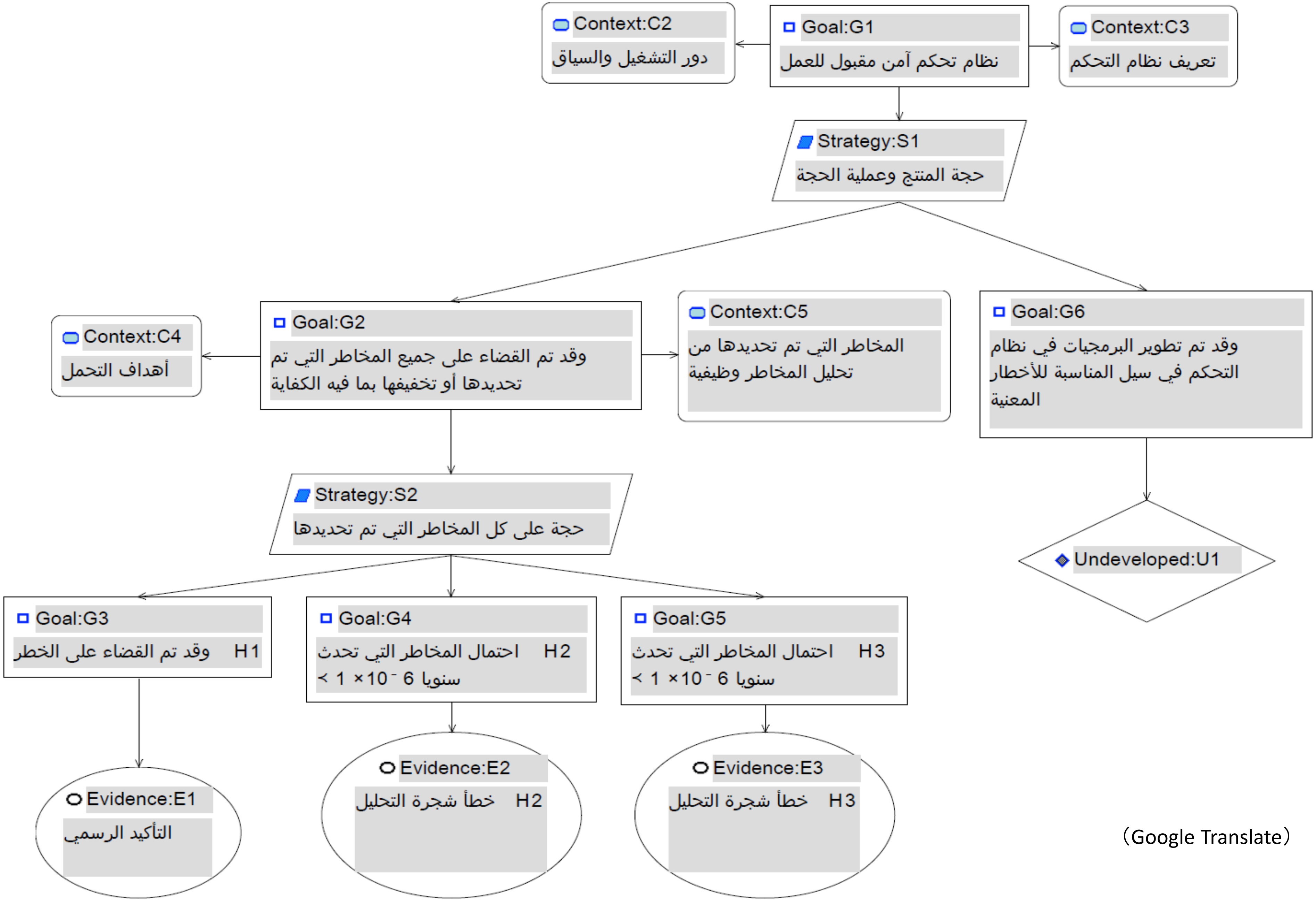
Evidence:E1
形式驗證結果

Evidence:E2
故障樹分析-H2

Evidence:E3
故障樹分析-H3

（Excite 中国語翻訳）

(or this)



Goal:G1
نظام تحكم آمن مقبول للعمل

Context:C2
دور التشغيل والسياق

Context:C3
تعريف نظام التحكم

Strategy:S1
حجة المنتج وعملية الحجة

Goal:G2
وقد تم القضاء على جميع المخاطر التي تم تحديدها أو تخفيفها بما فيه الكفاية

Context:C4
أهداف التحمل

Context:C5
المخاطر التي تم تحديدها من تحليل المخاطر وظيفية

Goal:G6
وقد تم تطوير البرمجيات في نظام التحكم في سيل المناسبة للأخطار المعنية

Undeveloped:U1

Strategy:S2
حجة على كل المخاطر التي تم تحديدها

Goal:G3
وقد تم القضاء على الخطر H1

Goal:G4
احتمال المخاطر التي تحدث H2
< 1 ×10 - 6 سنويا

Goal:G5
احتمال المخاطر التي تحدث H3
< 1 ×10 - 6 سنويا

Evidence:E1
التأكيد الرسمي

Evidence:E2
خطأ شجرة التحليل H2

Evidence:E3
خطأ شجرة التحليل H3

（Google Translate）

Approach: Formal Assurance Case

- The basis of argument's integrity checking
  = the relevant ontology
  - ***what things*** are a system and environment made of?
  - ***what properties*** *and relations* are required of / are assumed / constrain the sys. and env.?
- **Formal Assurance Case**
  ＝ machine-understandable ontology definition
  & argument based on that ontology
- Mundane checking is reduced to mechanical checking of **"Is this argument properly based on the given ontology?"**
  (whether the ont. is appropriate or not is for human to judge.)

Formal AC as *Theory & Proof*

- Formal Assurance Case

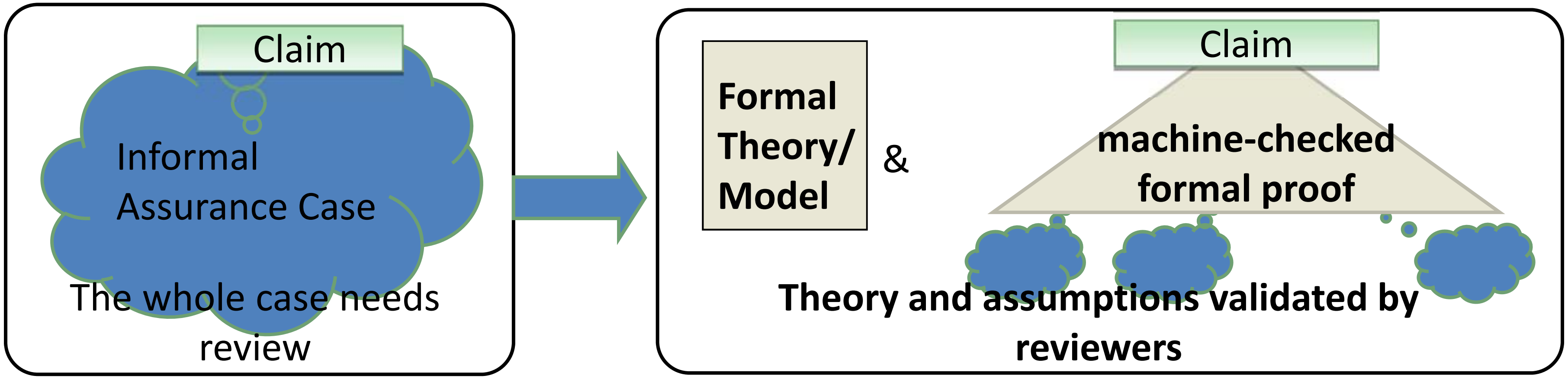
  ＝machine understandable ontology & arg. based on that ontology

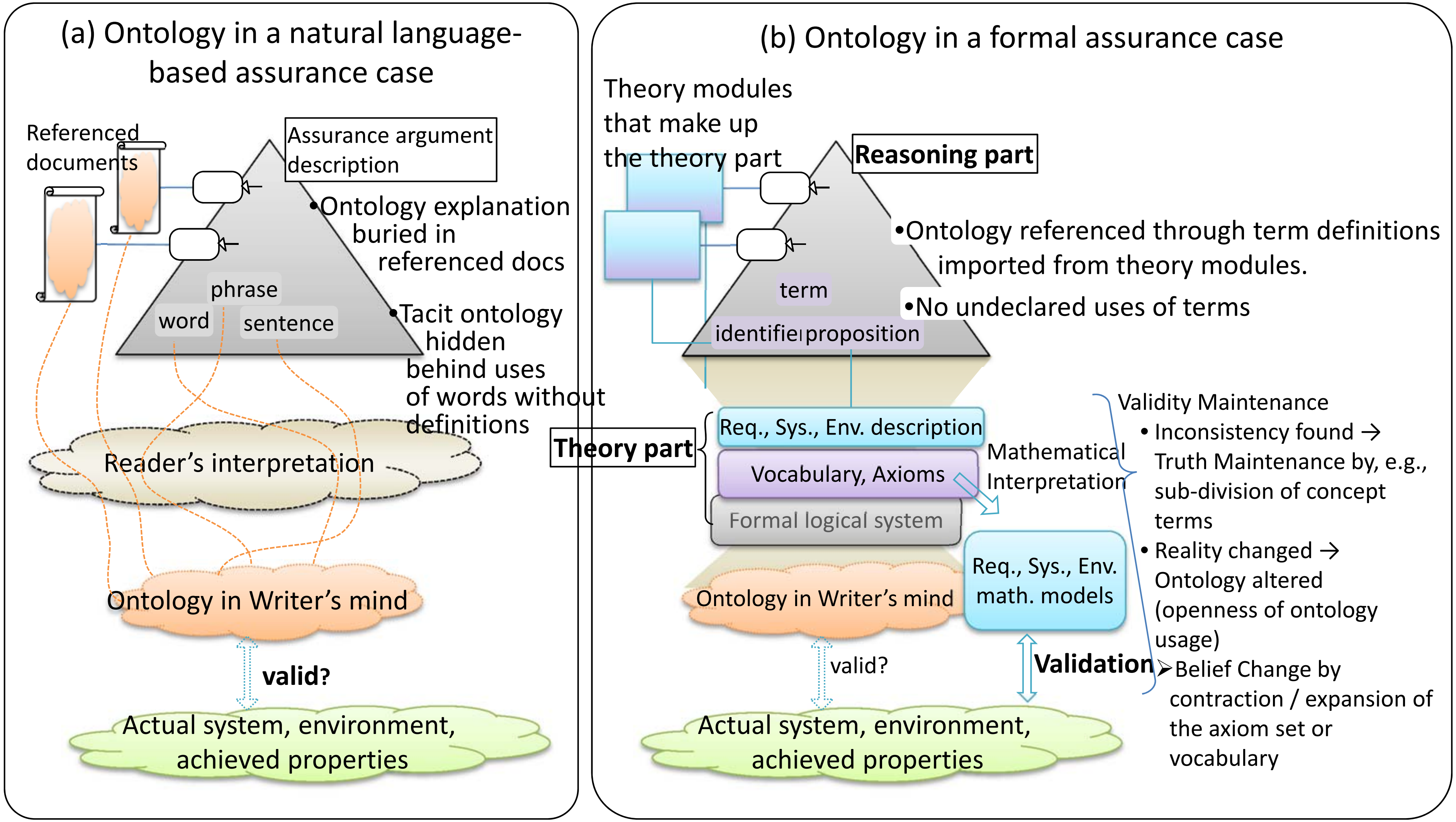
  ＝Formal theory & Formal proof in the formal thy.

Theory part of AC:
signature (vocabulary), axioms, defined terms, derived inferences,...

Reasoning part of AC:
legal combination of terms and inferences

- Mundane checking reduced to mechanical checking of **"Is this a formal proof in the given formal theory?"**


Claim
Informal
Assurance Case
The whole case needs review
Formal Theory/ Model
&
Claim
machine-checked formal proof
Theory and assumptions validated by reviewers


(a) Ontology in a natural language-based assurance case
Referenced documents
Assurance argument description
•Ontology explanation buried in referenced docs
phrase
word
sentence
•Tacit ontology hidden behind uses of words without definitions
Reader's interpretation
Ontology in Writer's mind
valid?
Actual system, environment, achieved properties
(b) Ontology in a formal assurance case
Theory modules that make up the theory part
Reasoning part
•Ontology referenced through term definitions imported from theory modules.
•No undeclared uses of terms
term
identifie proposition
Theory part
Req., Sys., Env. description
Vocabulary, Axioms
Formal logical system
Mathematical Interpretation
Req., Sys., Env. math. models
Ontology in Writer's mind
valid?
Validation
Actual system, environment, achieved properties
Validity Maintenance
• Inconsistency found → Truth Maintenance by, e.g., sub-division of concept terms
• Reality changed → Ontology altered (openness of ontology usage)
➢Belief Change by contraction / expansion of the axiom set or vocabulary

Formal AC as *Library & Program*

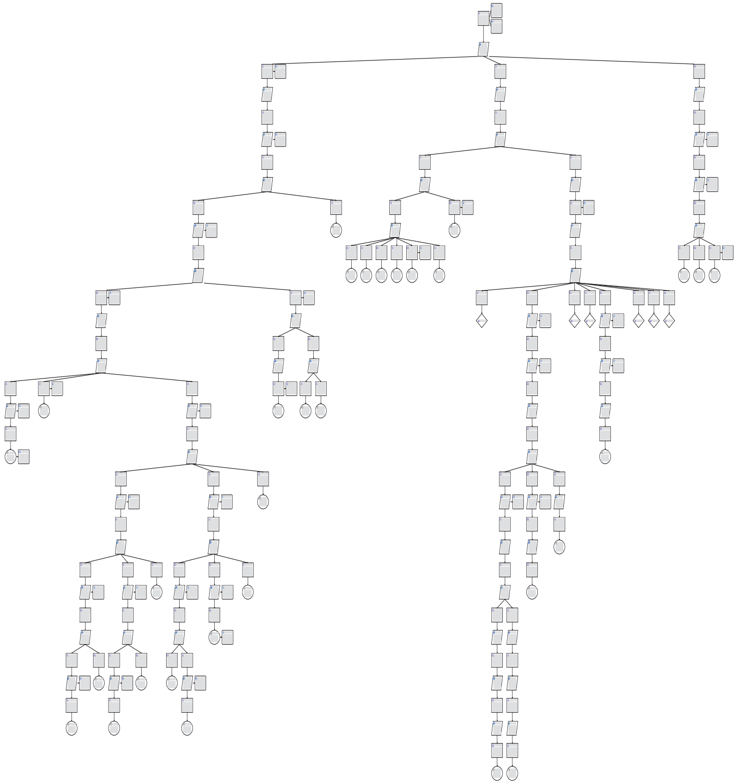
- Conventional formulations of formal thy/proof do not scale.
  - no definitional mechanism, no structuring / organization
- Write them as programs, applying "**propositions as types, proofs as programs**" paradigm.
  - Prop. *G* = Type *G* of data that count as direct evidence of *G.*
  - Prf. *p* of *G*= Program *p* that constructs data of type *G.*
- **Agda** supports this, with convenient prog. lang. features.
- **Formal Assurance Case in Agda** (FACIA)

  ＝ machine-understandable ontology & arg. based on that ontology

  ＝ Agda library providing types and functions

  & Agda program using the library

Problem



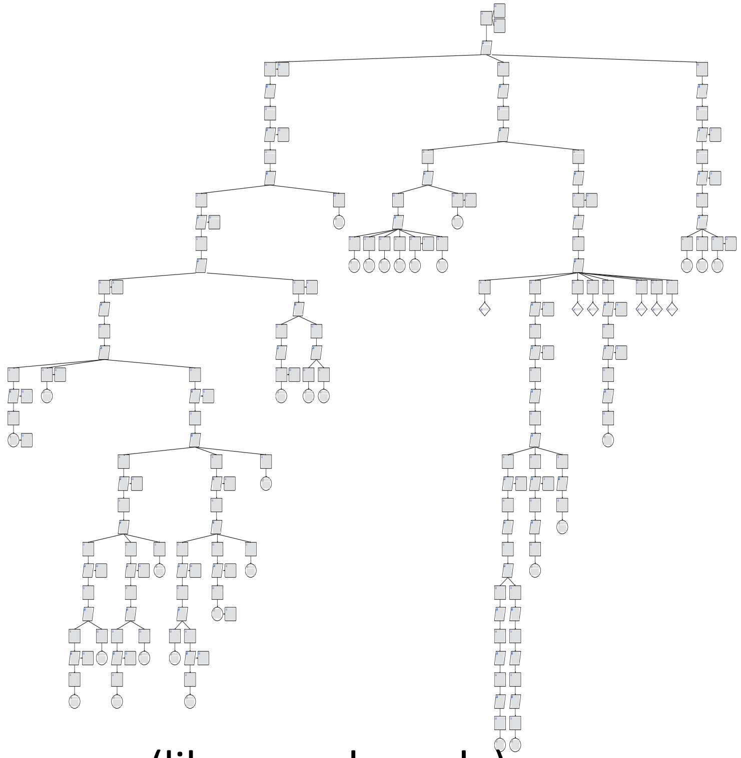
- Even mundane checking must be done by a human reviewer.
- Issues to check may span across 100s of connected arguments,
- each of which is frequently updated by many hands.

- Let machines check what they can check

  and let reviewers concentrate on exercising expert judgment.

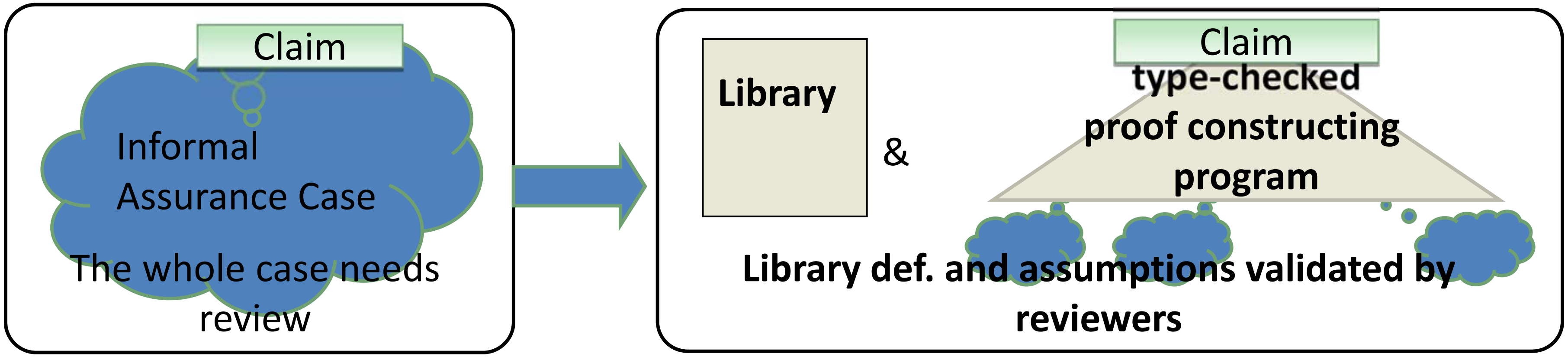
Solution with Formal Assurance Case In Agda

- FACIA = Library & Program
- **Checking an argument**
  = type-checking a program
- **Checking 100 connected args**
  = doing a build on a project files
- All the sw-eng. / programming techniques can be applied for constructing understandable, maintainable, large argument.
  (abstraction, modularisation, chg-mgmt, …)



(like pseudo-code)

(like compilable code)



Claim

Informal Assurance Case

The whole case needs review

Library

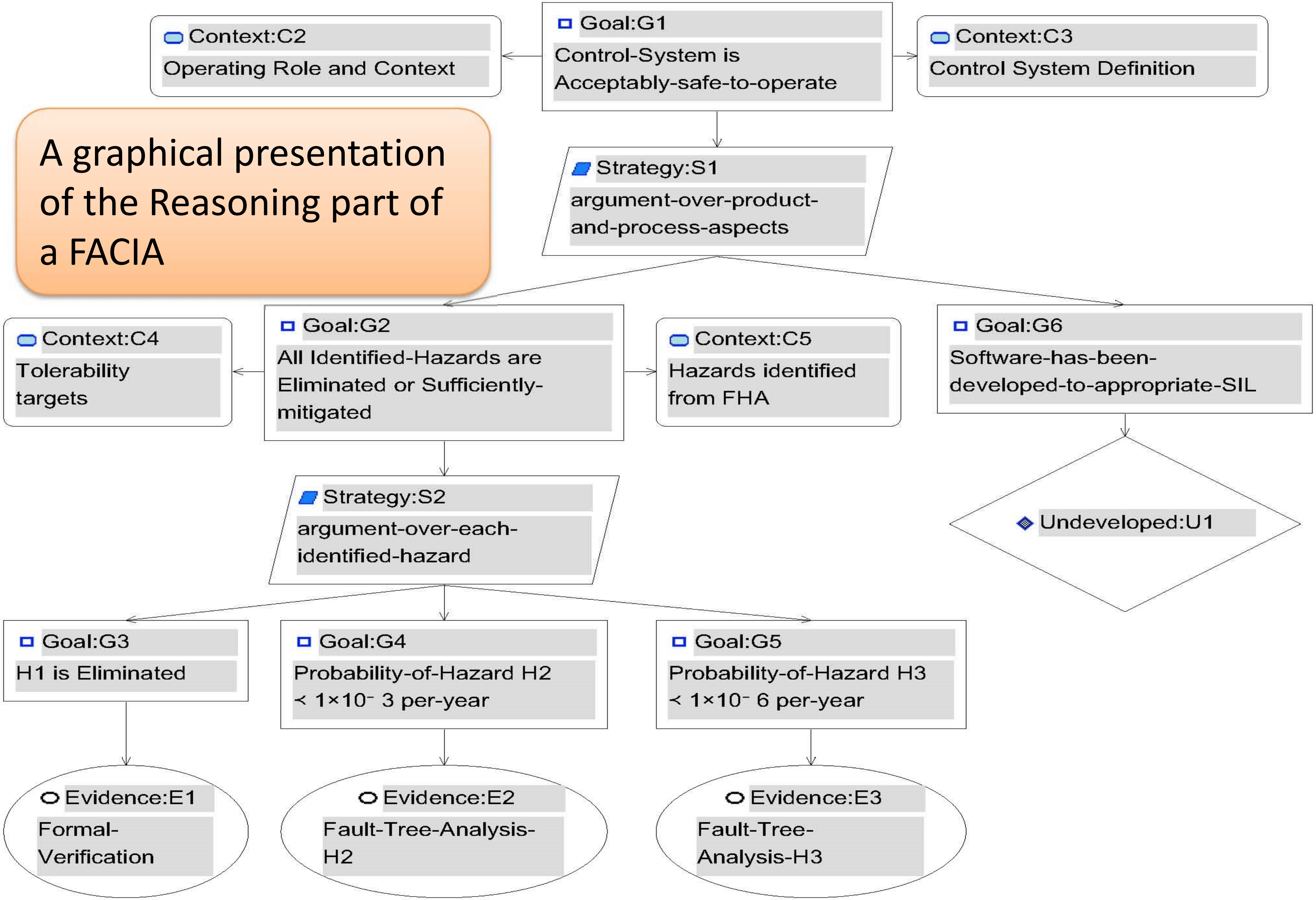
&

Claim

type-checked proof constructing program

Library def. and assumptions validated by reviewers

FACIA: a toy example



Goal:G1

Control-System is Acceptably-safe-to-operate

Context:C2

Operating Role and Context

Context:C3

Control System Definition

Strategy:S1

argument-over-product-and-process-aspects

Goal:G2

All Identified-Hazards are Eliminated or Sufficiently-mitigated

Context:C4

Tolerability targets

Context:C5

Hazards identified from FHA

Goal:G6

Software-has-been-developed-to-appropriate-SIL

Undeveloped:U1

Strategy:S2

argument-over-each-identified-hazard

Goal:G3

H1 is Eliminated

Goal:G4

Probability-of-Hazard H2 $< 1\times10^{-3}$ per-year

Goal:G5

Probability-of-Hazard H3 $< 1\times10^{-6}$ per-year

Evidence:E1

Formal-Verification

Evidence:E2

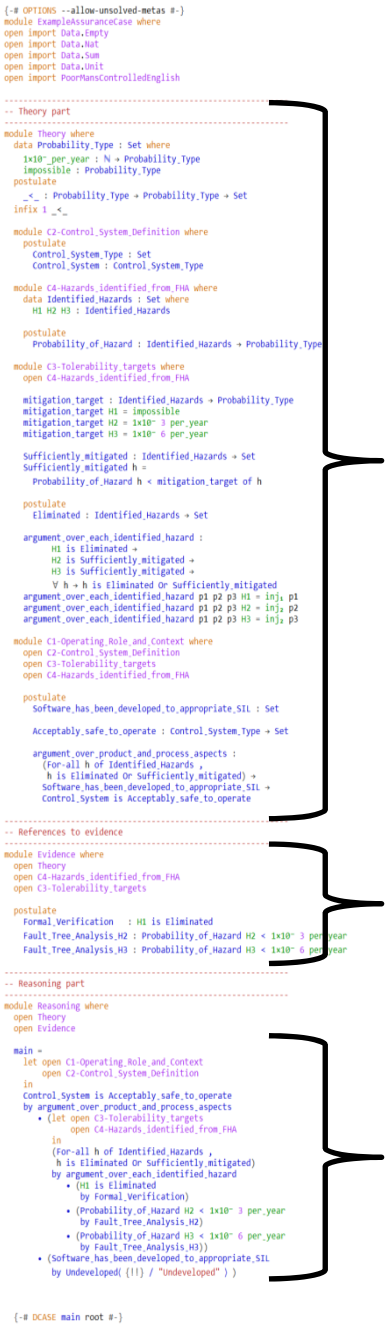
Fault-Tree-Analysis-H2

Evidence:E3

Fault-Tree-Analysis-H3

A graphical presentation of the Reasoning part of a FACIA

FACIA: a toy example



```
{-# OPTIONS --allow-unsolved-metas #-}
module ExampleAssuranceCase where
open import Data.Empty
open import Data.Nat
open import Data.Sum
open import Data.Unit
open import PoorMansControlledEnglish

------------------------------------------------------------
-- Theory part
------------------------------------------------------------
module Theory where
  data Probability_Type : Set where
    1×10⁻_per_year : ℕ → Probability_Type
    impossible : Probability_Type
  postulate
    _<_ : Probability_Type → Probability_Type → Set
  infix 1 _<_

  module C2-Control_System_Definition where
    postulate
      Control_System_Type : Set
      Control_System : Control_System_Type

  module C4-Hazards_identified_from_FHA where
    data Identified_Hazards : Set where
      H1 H2 H3 : Identified_Hazards

    postulate
      Probability_of_Hazard : Identified_Hazards → Probability_Type

  module C3-Tolerability_targets where
    open C4-Hazards_identified_from_FHA

    mitigation_target : Identified_Hazards → Probability_Type
    mitigation_target H1 = impossible
    mitigation_target H2 = 1×10⁻ 3 per_year
    mitigation_target H3 = 1×10⁻ 6 per_year

    Sufficiently_mitigated : Identified_Hazards → Set
    Sufficiently_mitigated h =
      Probability_of_Hazard h < mitigation_target of h

    postulate
      Eliminated : Identified_Hazards → Set

    argument_over_each_identified_hazard :
          H1 is Eliminated →
          H2 is Sufficiently_mitigated →
          H3 is Sufficiently_mitigated →
          ∀ h → h is Eliminated Or Sufficiently_mitigated
    argument_over_each_identified_hazard p1 p2 p3 H1 = inj₁ p1
    argument_over_each_identified_hazard p1 p2 p3 H2 = inj₂ p2
    argument_over_each_identified_hazard p1 p2 p3 H3 = inj₂ p3

  module C1-Operating_Role_and_Context where
    open C2-Control_System_Definition
    open C3-Tolerability_targets
    open C4-Hazards_identified_from_FHA

    postulate
      Software_has_been_developed_to_appropriate_SIL : Set

      Acceptably_safe_to_operate : Control_System_Type → Set

      argument_over_product_and_process_aspects :
        (For-all h of Identified_Hazards ,
         h is Eliminated Or Sufficiently_mitigated) →
        Software_has_been_developed_to_appropriate_SIL →
        Control_System is Acceptably_safe_to_operate

------------------------------------------------------------
-- References to evidence
------------------------------------------------------------
module Evidence where
  open Theory
  open C4-Hazards_identified_from_FHA
  open C3-Tolerability_targets

  postulate
    Formal_Verification   : H1 is Eliminated
    Fault_Tree_Analysis_H2 : Probability_of_Hazard H2 < 1×10⁻ 3 per_year
    Fault_Tree_Analysis_H3 : Probability_of_Hazard H3 < 1×10⁻ 6 per_year

------------------------------------------------------------
-- Reasoning part
------------------------------------------------------------
module Reasoning where
  open Theory
  open Evidence

  main =
    let open C1-Operating_Role_and_Context
        open C2-Control_System_Definition
    in
    Control_System is Acceptably_safe_to_operate
    by argument_over_product_and_process_aspects
       • (let open C3-Tolerability_targets
              open C4-Hazards_identified_from_FHA
          in
          (For-all h of Identified_Hazards ,
           h is Eliminated Or Sufficiently_mitigated)
          by argument_over_each_identified_hazard
             • (H1 is Eliminated
                by Formal_Verification)
             • (Probability_of_Hazard H2 < 1×10⁻ 3 per_year
                by Fault_Tree_Analysis_H2)
             • (Probability_of_Hazard H3 < 1×10⁻ 6 per_year
                by Fault_Tree_Analysis_H3))
       • (Software_has_been_developed_to_appropriate_SIL
          by Undeveloped( {!!} / "Undeveloped" ) )


  {-# DCASE main root #-}
```

“Theory” part: (Library definition)

- Declares and defines the basis of the argument:
  Primitive terms for primitive things and concepts,
  Defined terms for defined things and concepts,
  Presumptive relationship among legal terms.
- Gives definite meaning to any legal combination of terms.
- Must be agreed / approved through supporting process.
- Organized into modules corresponding to contexts.

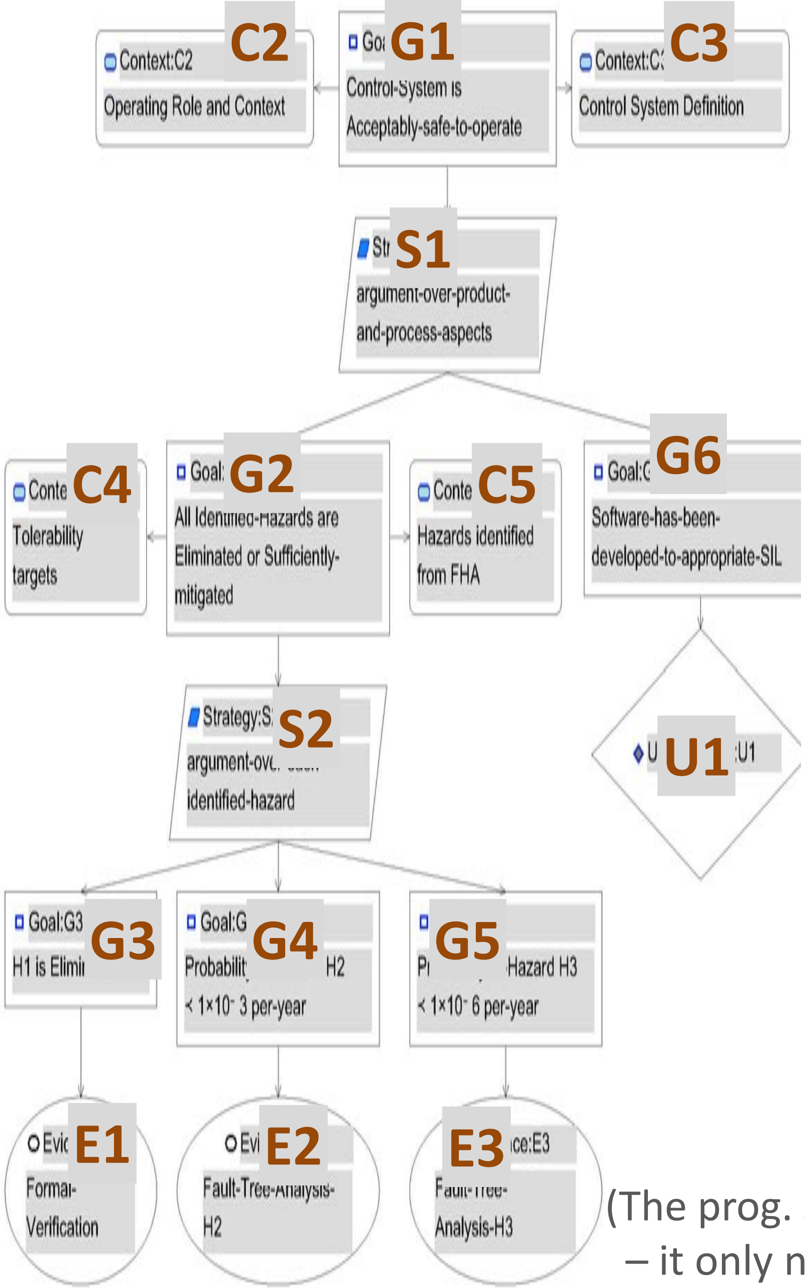
“Evidence” part

- Declares presumptive existence of evidence to some claim-terms
- Must be agreed / approved through supporting process

“Reasoning” part: (Main routine)

- Exhibits a combined term as a proof for the top claim-term.
  That this is a legal proof is machine-checked.
- The top claim-term must be agreed / approved.

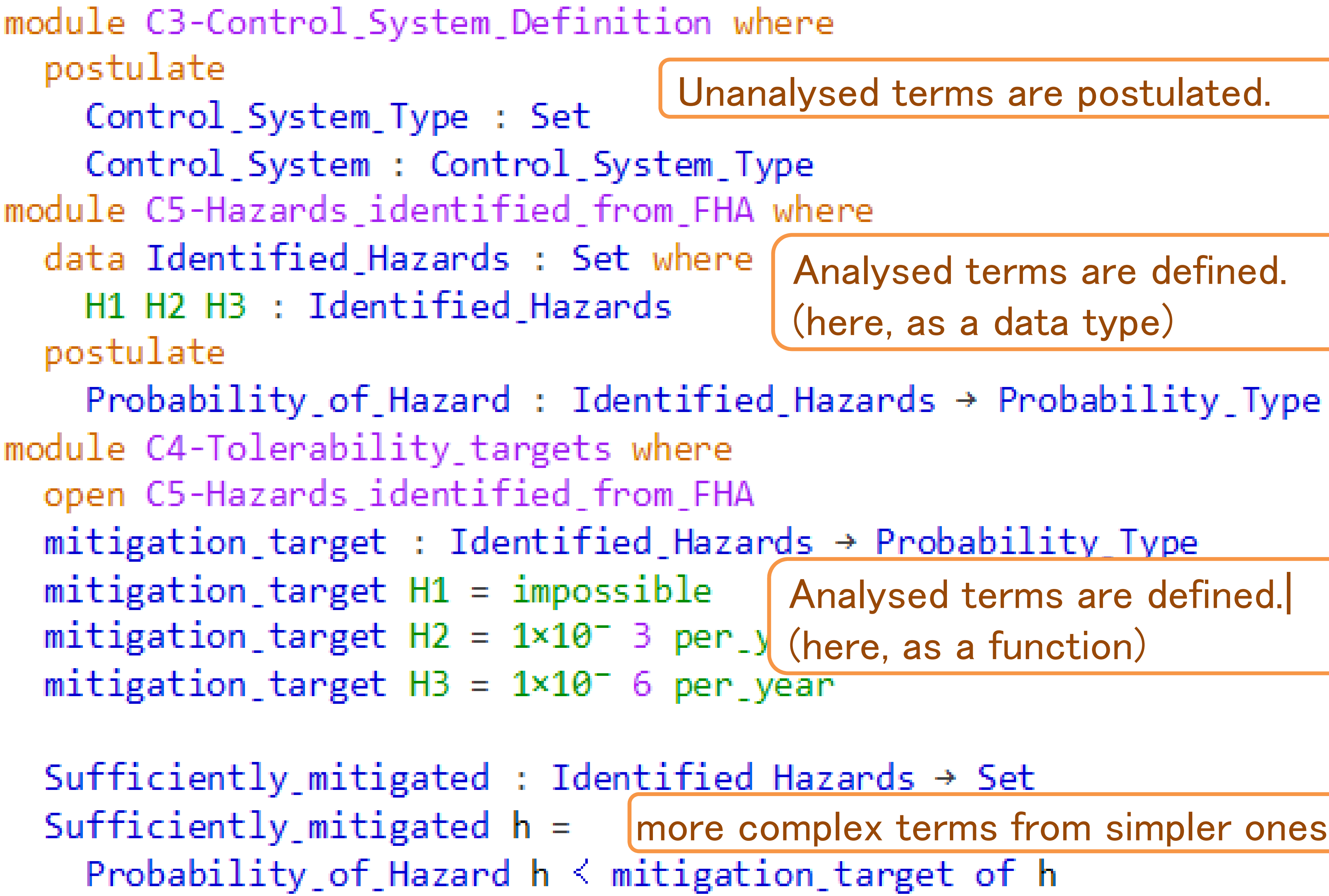
Tree ~ Program



```
let open C2-Operating_Role_and_Context
    open C3-Control_System_Definition
in
Control_System is Acceptably_safe_to_operate -- G1
by argument_over_product_and_process_aspects -- S2
   -- left sub-case.
   • (let open C4-Tolerability_targets
          open C5-Hazards_identified_from_FHA
      in
      (All Identified_Hazards are
       Eliminated or Sufficiently_mitigated)   -- G2
      by argument_over_each_identified_hazard  -- S2
         • (H1 is Eliminated            -- G3
            by Formal_Verification)     -- E1
         • (Probability_of_Hazard H2 < 1×10¯ 3 per_year -- G4
            by Fault_Tree_Analysis_H2)                  -- E2
         • (Probability_of_Hazard H3 < 1×10¯ 6 per_year -- G5
            by Fault_Tree_Analysis_H3))                 -- E3
   -- right sub-case
   • (Software_has_been_developed_to_appropriate_SIL -- G6
      by Undeveloped( {!!} / "Undeveloped" ) )       -- U1
```

(The prog. side is not limited to a tree form
– it only needs to compute to a tree.)

Vocabulary ~ Def/decl of types and funcs



```
module C3-Control_System_Definition where
  postulate
    Control_System_Type : Set
    Control_System : Control_System_Type
module C5-Hazards_identified_from_FHA where
  data Identified_Hazards : Set where
    H1 H2 H3 : Identified_Hazards
  postulate
    Probability_of_Hazard : Identified_Hazards → Probability_Type
module C4-Tolerability_targets where
  open C5-Hazards_identified_from_FHA
  mitigation_target : Identified_Hazards → Probability_Type
  mitigation_target H1 = impossible
  mitigation_target H2 = 1×10⁻ 3 per_y
  mitigation_target H3 = 1×10⁻ 6 per_year

  Sufficiently_mitigated : Identified Hazards → Set
  Sufficiently_mitigated h =
    Probability_of_Hazard h < mitigation_target of h
```

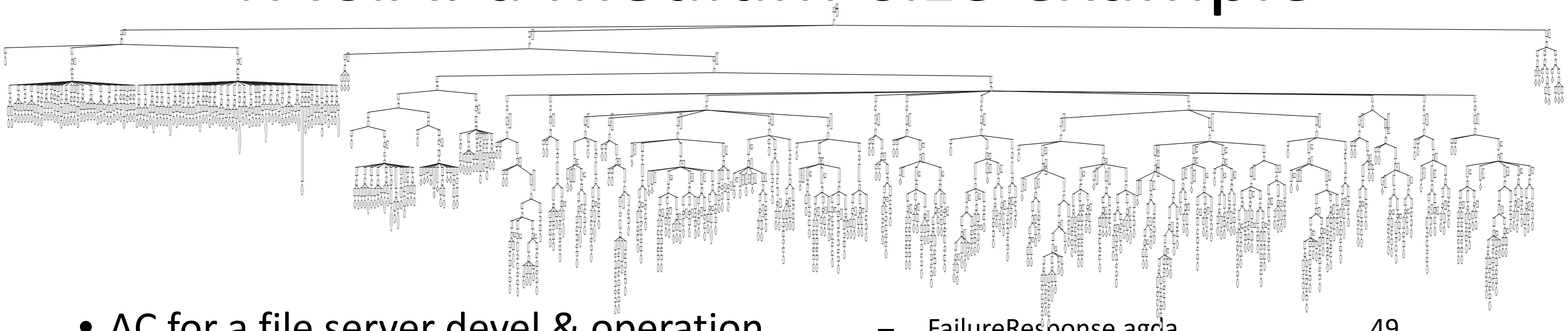
Unanalysed terms are postulated.

Analysed terms are defined.
(here, as a data type)

Analysed terms are defined.
(here, as a function)

more complex terms from simpler ones

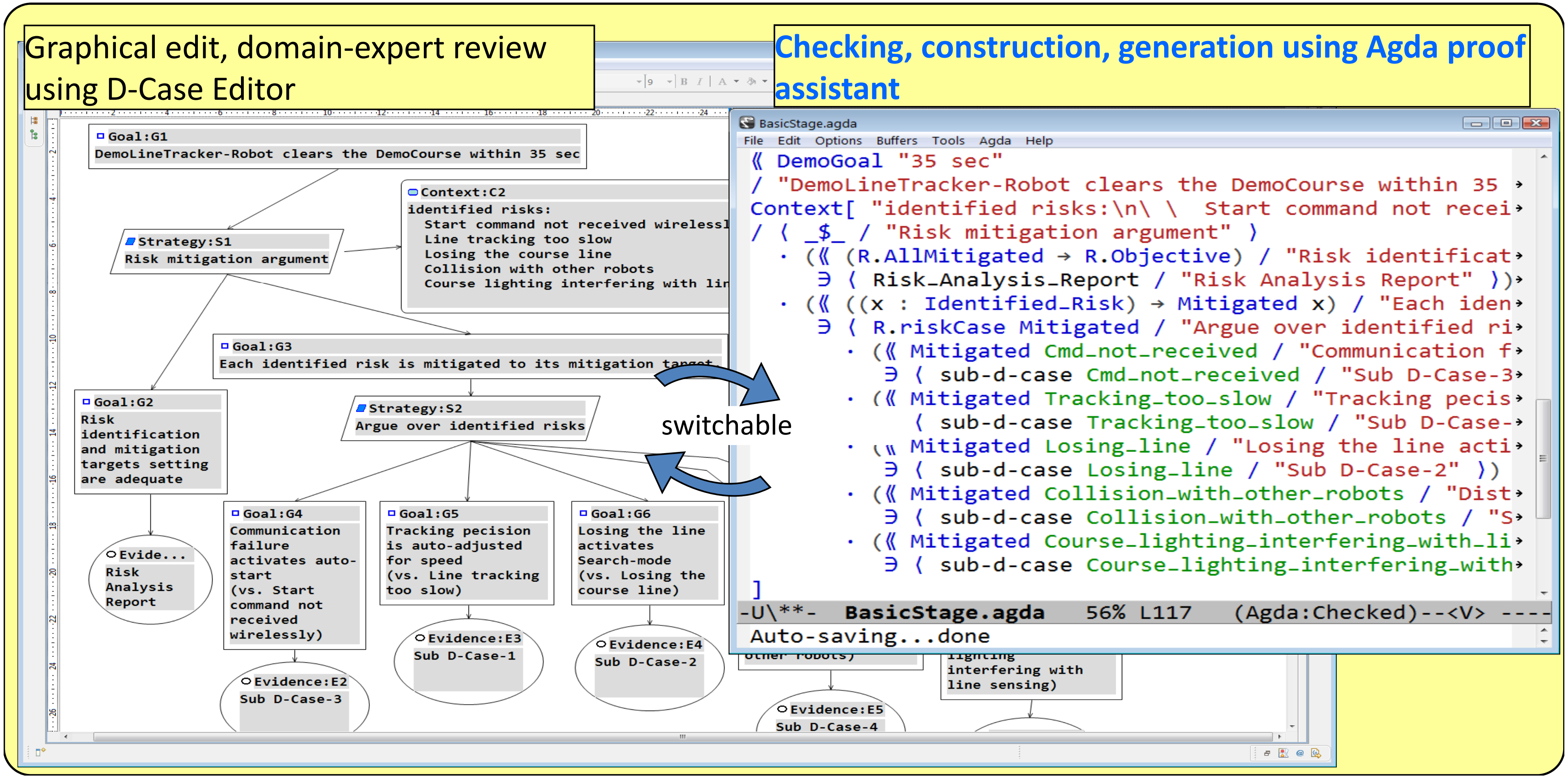
FACIA: a medium-size example



- AC for a file server devel & operation.
- 29 Files, 4 KLOC
- TopLevelArgument.agda 25
- Context
  - DEOSProcess.agda 24
  - DEOSProcess
    - ChangeAccommodation.agda 112
    - FailureResponse.agda 83
    - NormalOperation.agda 34
  - UserRequirements.agda 131
  - SpecifiedRequirements.agda 179
  - DesignSpecification.agda 176
  - SpecifiedReqToUserReq.agda 61
  - DesignSpecToSpecifiedReq.agda 81
  - RiskTreatment.agda 230
  - SpecifiedReqTest.agda 40
- Argument
  - NormalOperation.agda 22
  - NormalOperation
    - SpecifiedReqToUserReq.agda 83
    - DesignSpecToSpecifiedReq.agda 127
  - FailureResponse.agda 49
  - FailureResponse
    - 議論木
    - RiskTreatment.agda 69
    - RiskTreatment
      - RiskCasesKido.agda 2110
    - SpecifiedReqToUserReq.agda 40
    - DesignSpecToSpecifiedReq.agda 58
    - SpecifiedReqTest.agda 43
  - ChangeAccommodation.agda 34
- Evidence
  - DesignPremises.agda 78
  - TestPremises.agda 13
  - TestResults.agda 140
- Utilities.agda 6
- Utilities
  - DCaseSpecConvenience.agda 41
  - KDCase.agda 93
  - KDCaseShallow.agda 40

D-Case/Agda (“D-Case in Agda” Verification Tool)

- Provides translation between arg. in graphical form and in Agda program form.
- AC as an Agda program is checked in Agda devlopement environment, which is also a proof-assistant for constructing args as programs.



Graphical edit, domain-expert review using D-Case Editor

Goal:G1
DemoLineTracker-Robot clears the DemoCourse within 35 sec

Context:C2
identified risks:
  Start command not received wirelessl
  Line tracking too slow
  Losing the course line
  Collision with other robots
  Course lighting interfering with lin

Strategy:S1
Risk mitigation argument

Goal:G3
Each identified risk is mitigated to its mitigation target.

Goal:G2
Risk identification and mitigation targets setting are adequate

Strategy:S2
Argue over identified risks

Evide...
Risk Analysis Report

Goal:G4
Communication failure activates auto-start
(vs. Start command not received wirelessly)

Goal:G5
Tracking pecision is auto-adjusted for speed
(vs. Line tracking too slow)

Goal:G6
Losing the line activates Search-mode
(vs. Losing the course line)

Evidence:E2
Sub D-Case-3

Evidence:E3
Sub D-Case-1

Evidence:E4
Sub D-Case-2

other robots)

lighting
interfering with
line sensing)

Evidence:E5
Sub D-Case-4

Checking, construction, generation using Agda proof assistant

```
《 DemoGoal "35 sec"
/ "DemoLineTracker-Robot clears the DemoCourse within 35 →
Context[ "identified risks:\n\ \  Start command not recei→
/ 〈 _$_ / "Risk mitigation argument" 〉
  • (《 (R.AllMitigated → R.Objective) / "Risk identificat→
     ∋ 〈 Risk_Analysis_Report / "Risk Analysis Report" 〉)→
  • (《 ((x : Identified_Risk) → Mitigated x) / "Each iden→
     ∋ 〈 R.riskCase Mitigated / "Argue over identified ri→
       • (《 Mitigated Cmd_not_received / "Communication f→
          ∋ 〈 sub-d-case Cmd_not_received / "Sub D-Case-3→
       • (《 Mitigated Tracking_too_slow / "Tracking pecis→
            〈 sub-d-case Tracking_too_slow / "Sub D-Case-→
       • (《 Mitigated Losing_line / "Losing the line acti→
          ∋ 〈 sub-d-case Losing_line / "Sub D-Case-2" 〉)
       • (《 Mitigated Collision_with_other_robots / "Dist→
          ∋ 〈 sub-d-case Collision_with_other_robots / "S→
       • (《 Mitigated Course_lighting_interfering_with_li→
          ∋ 〈 sub-d-case Course_lighting_interfering_with→
]
-U\**-  BasicStage.agda    56% L117    (Agda:Checked)--<V> ----
Auto-saving...done
```

Summary

- Informal AC = unspecified ontology & arg. in nat. lang.

→ **explicit ontology** & **arg. based on that**

→ **Formal AC = formal theory & formal proof in it**

→ **FACIA = Library of types/funcs & program**

- **Checking an argument = Type checking a program**
- Software engineering applied to argument construction.
- The approach itself is contents-neutral / contents-free. No hints for what should be argued in an AC.

→ Currently working on a **Framework for Formal AC for "Open Systems Dependability"** (FFO) that provides contents for certain AC applications,

like a software framework does in some app domain.