McMaster University

Inspiring Innovation and Discovery

A Template for an Assurance Case Shall Be Known as an Assurance Case Template

Alan Wassyng

With lots of help if not always encouragement from:

Tom Maibaum, Mark Lawford, Neeraj Singh, Paul Joannou

McSCert

VeriSure: July 2015

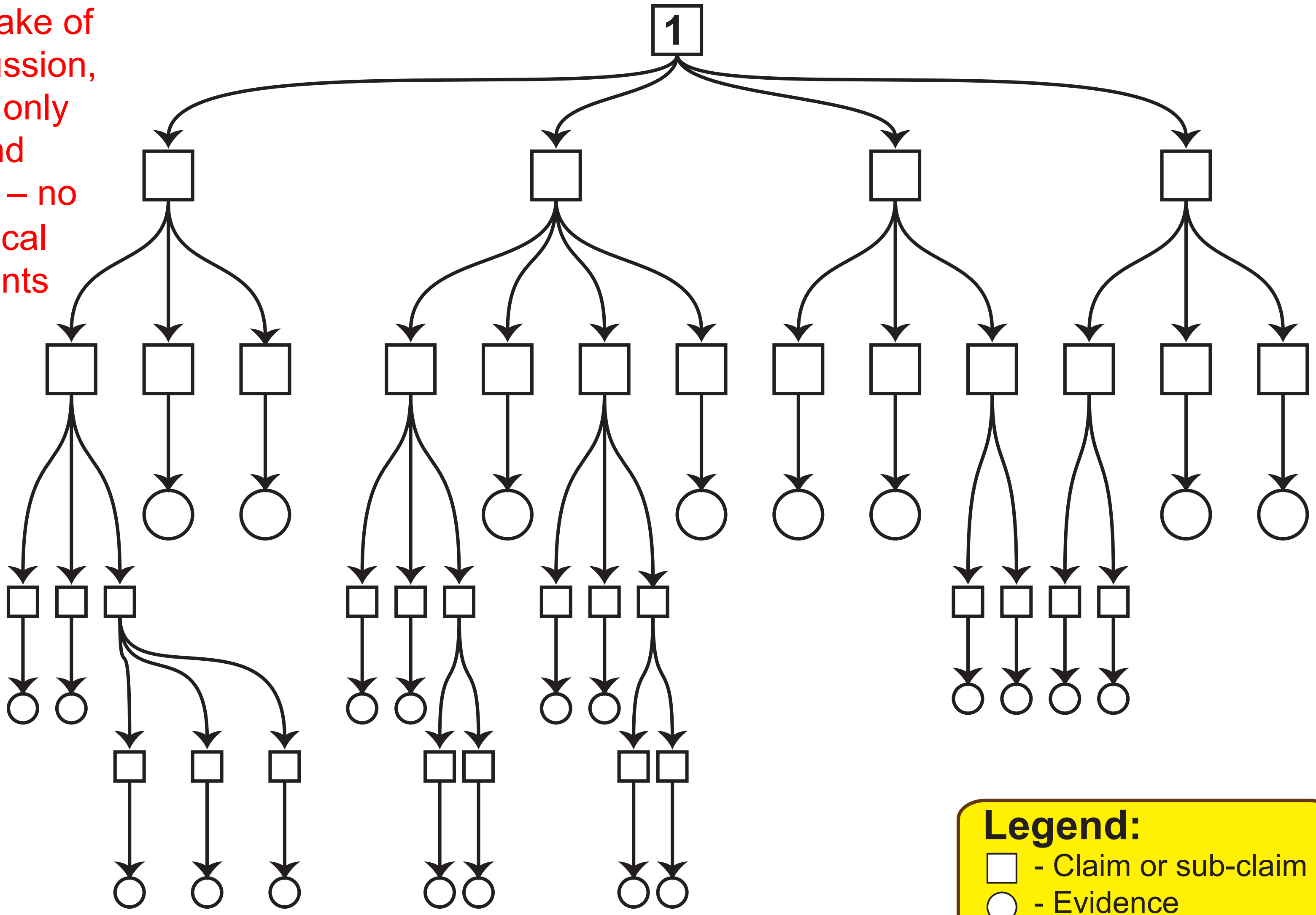
Motivation for the Title

Objective

- Build an assurance case so that it is effective for a class of products within an application domain – infusion pumps, for example
- The assurance case has placeholders for specific evidence and other entities

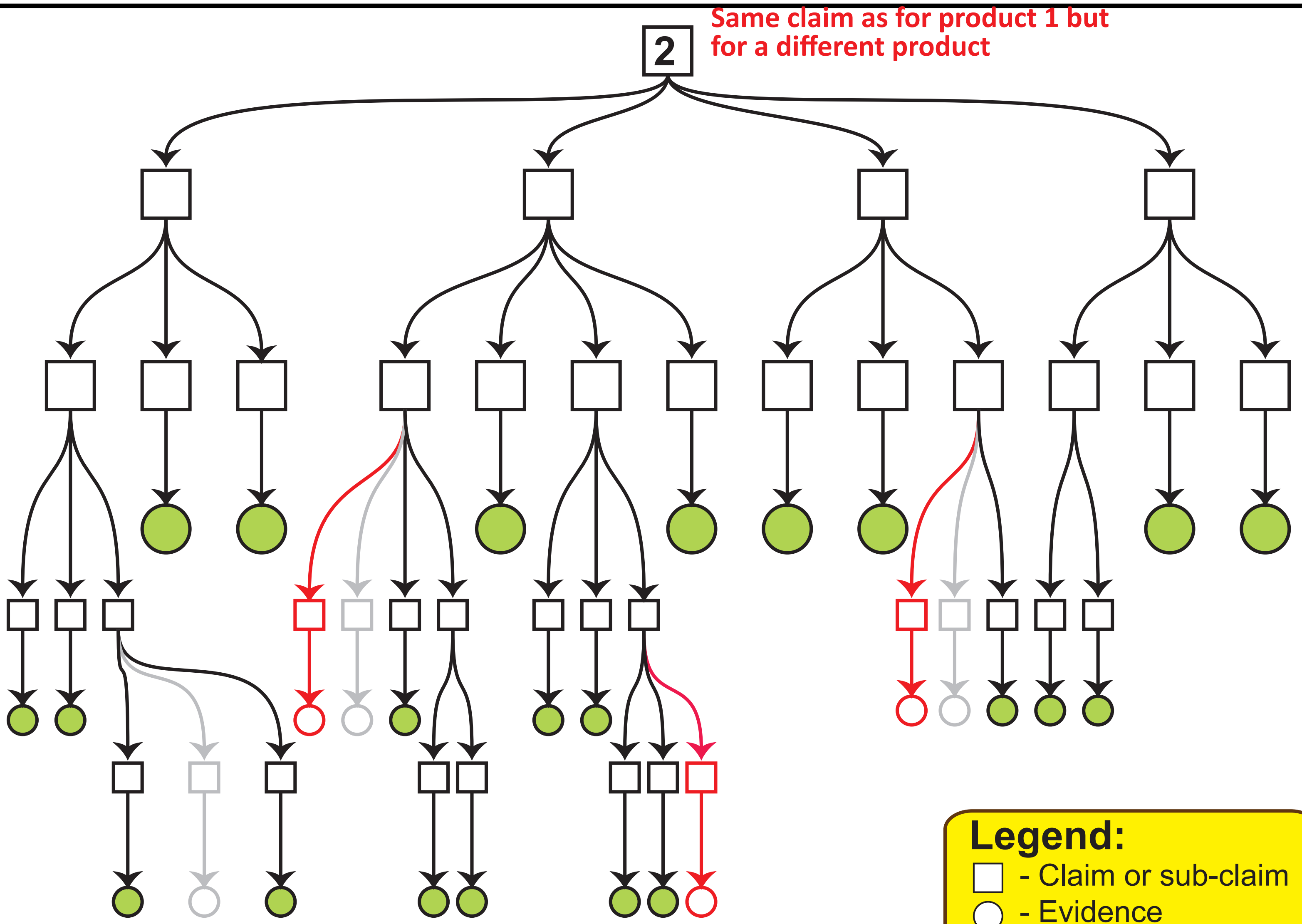
Assurance Case for Product 1

For the sake of this discussion, we show only claims and evidence – no other typical components



Legend:
- □ - Claim or sub-claim
- ○ - Evidence
- A → B - A is claim B is premise

Assurance Case for Product 2



Legend:

□ - Claim or sub-claim

○ - Evidence

A → B - A is claim
B is premise

added removed content changed

3 Types of Changes

- To build a comprehensive template, we need to understand what could be different in the assurance cases for different (but very similar products)
- We have identified 3 kinds of changes that may occur when developing one product after another
  - Content – evidence in the different cases can be different even if the evidence supports the same claim
  - Additions – the newer product may have features not included in the older product and this may necessitate claims, sub-claims and evidence not in the earlier product
  - Removals – the newer product may not have features included in the older product and this may necessitate removal of claims, sub-claims and evidence in the earlier product

3 Types of Changes

- To build a comprehensive template, we need to understand what could be different in the assurance cases for different (but very similar products)
- We have identified 3 kinds of changes that may occur when going from one product to another
  - Content – evidence in the different cases can be different even if the evidence supports the same claim
  - Additions – the newer product may have features not included in the older product and this may necessitate claims, sub-claims and evidence not in the earlier product
  - Removals – the newer product may not have features included in the older product and this may necessitate removal of claims, sub-claims and evidence in the earlier product

Looks like we want to do incremental assurance, but it turns out to be different in some important ways

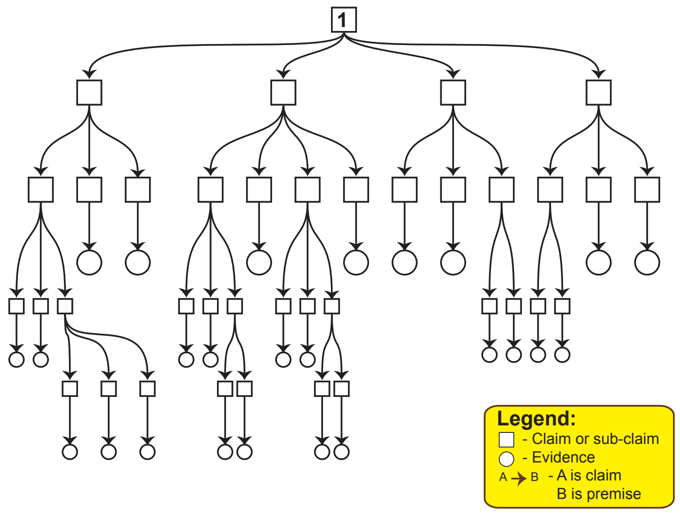
It is also not the same as simply instantiating an assurance case pattern with different results

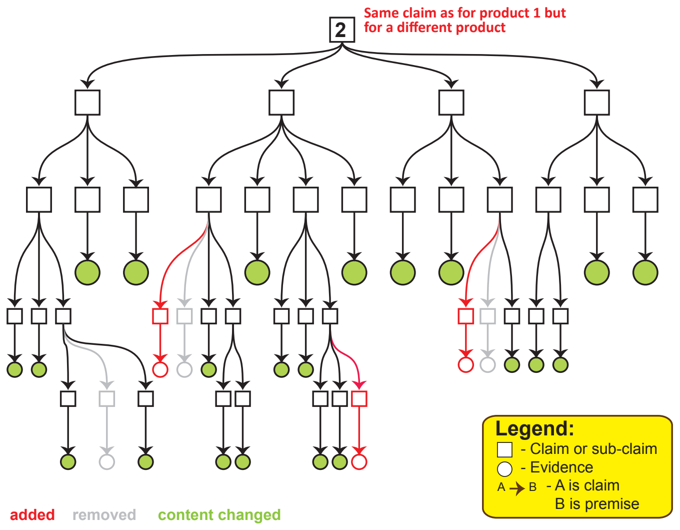
~~3~~ 2 Types of Changes

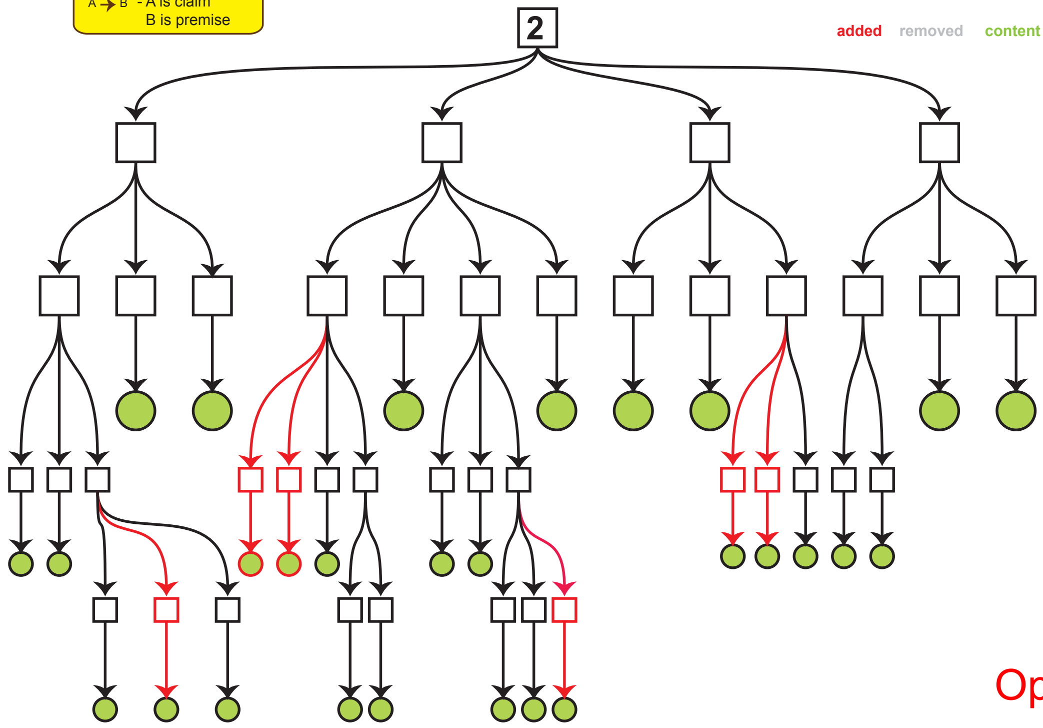
- If you look at the situation a little differently – and not as an incremental approach, then additions and removals are not really different
  - They just seem different if we think of a temporal ordering of the products
  - However, that is not what we want in a template
  - We want to cope with a set of products from the outset that are essentially very similar – but do have differences

  - Actually, we are left with only 2 types of changes
    - Content changes in evidence
    - “Optional” (claim, sub-claim, evidence) paths

Putting It Together



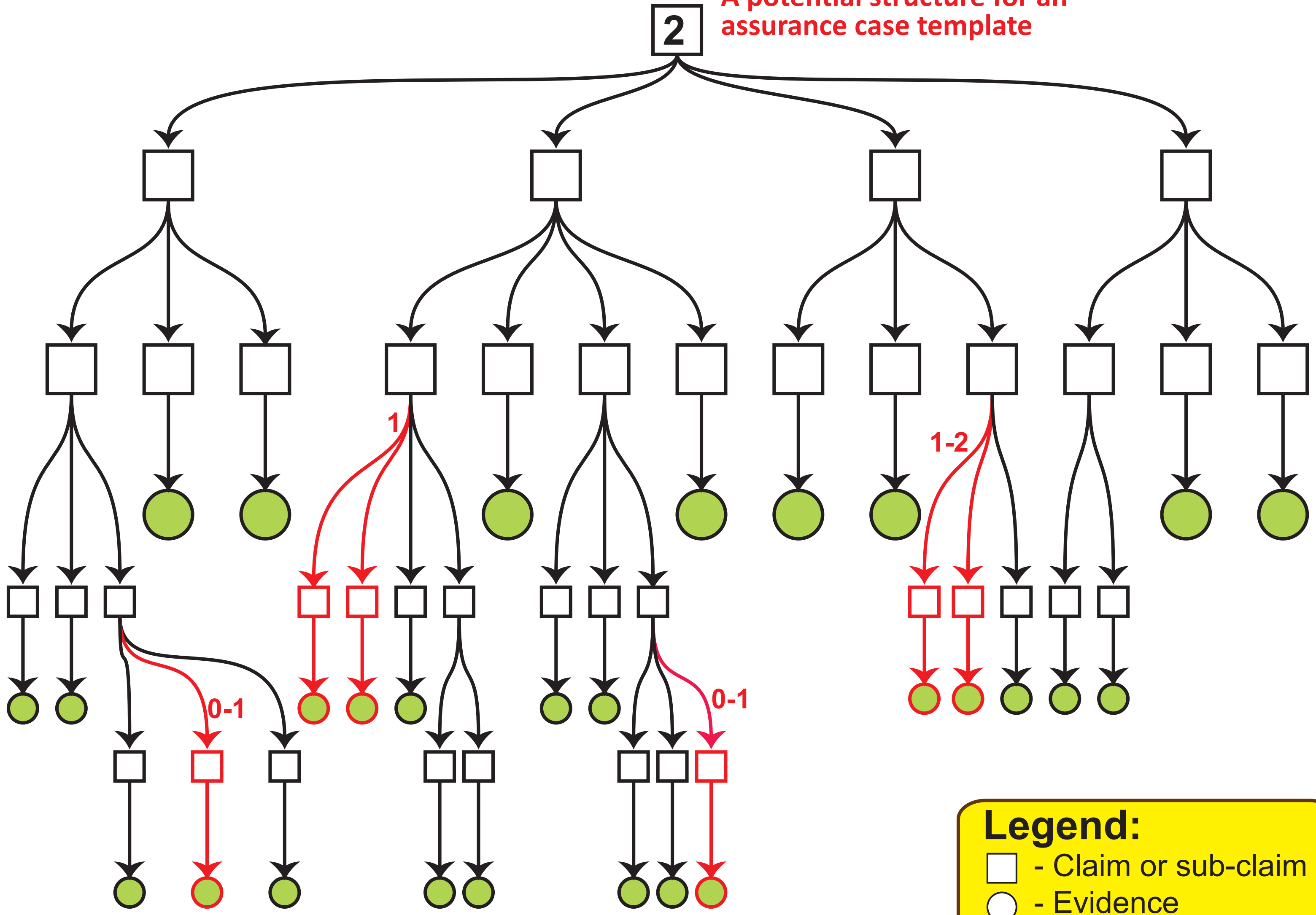




Optional paths in red

A Suggested Template Mechanism

A potential structure for an assurance case template

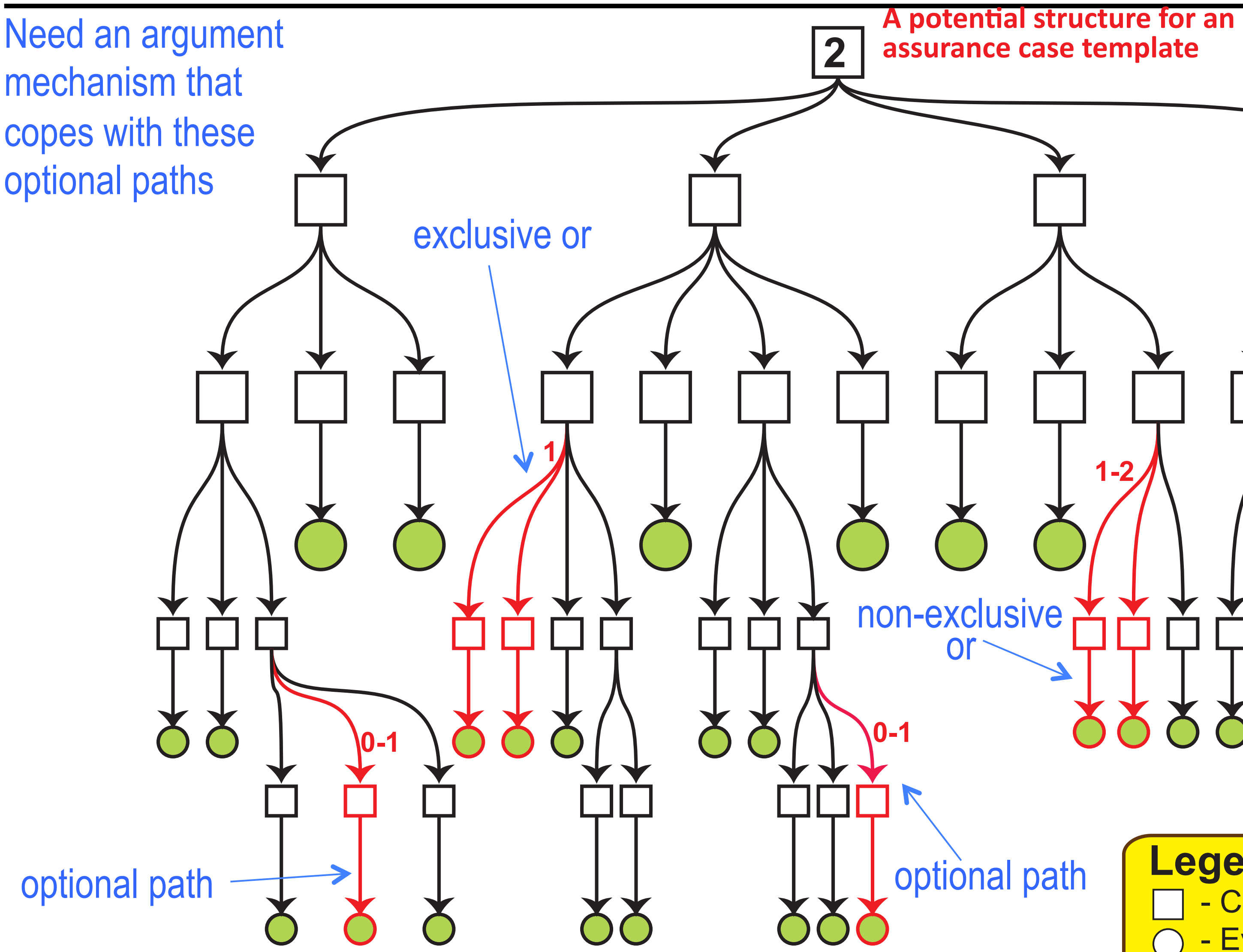


Legend:
- □ - Claim or sub-claim
- ○ - Evidence
- A → B - A is claim
  B is premise

optional paths - need notation to specify number of paths required
acceptance criteria on evidence required specified in the template

A Suggested Template Mechanism

Need an argument mechanism that copes with these optional paths



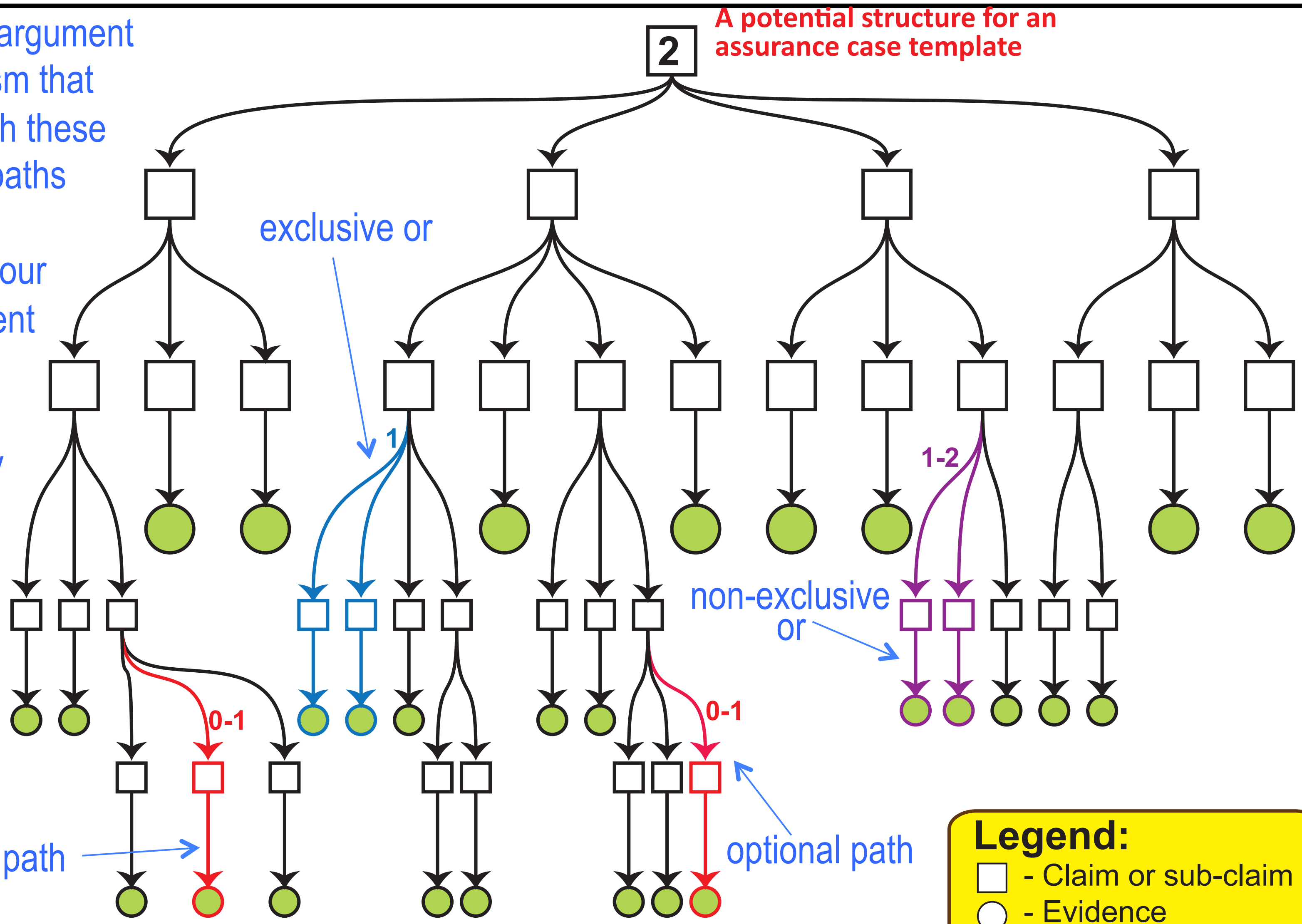
Legend:
- □ - Claim or sub-claim
- ○ - Evidence
- A → B - A is claim, B is premise

optional paths - need notation to specify number of paths required
acceptance criteria on evidence required specified in the template

A Suggested Template Mechanism

Need an argument mechanism that copes with these optional paths

Could colour the different kinds of paths differently



Legend:

□	- Claim or sub-claim
○	- Evidence
A → B	- A is claim B is premise

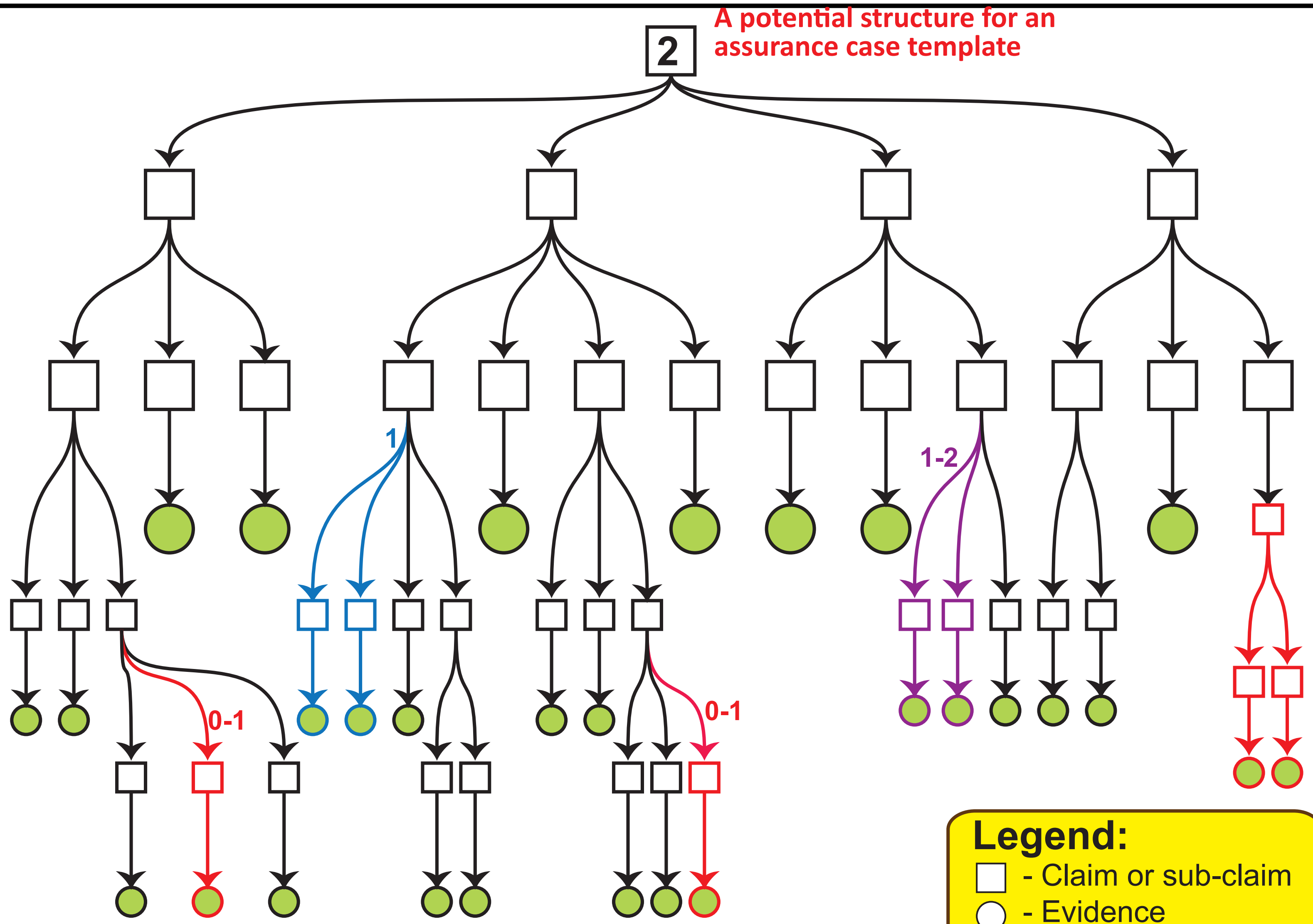
acceptance criteria on evidence required specified in the template

Insulin Pump Examples

- Optional paths
  - A light to help people use the pump in the dark
- Exclusive or paths
  - Some pumps use a standard Luer connector for their infusion sets – others do not
- Non-exclusive or paths
  - Some pumps allow you to input glucose readings directly from an associated meter, or to input those readings manually, and some hazards for these two options are likely to be different

Do We Need This?

McSCert



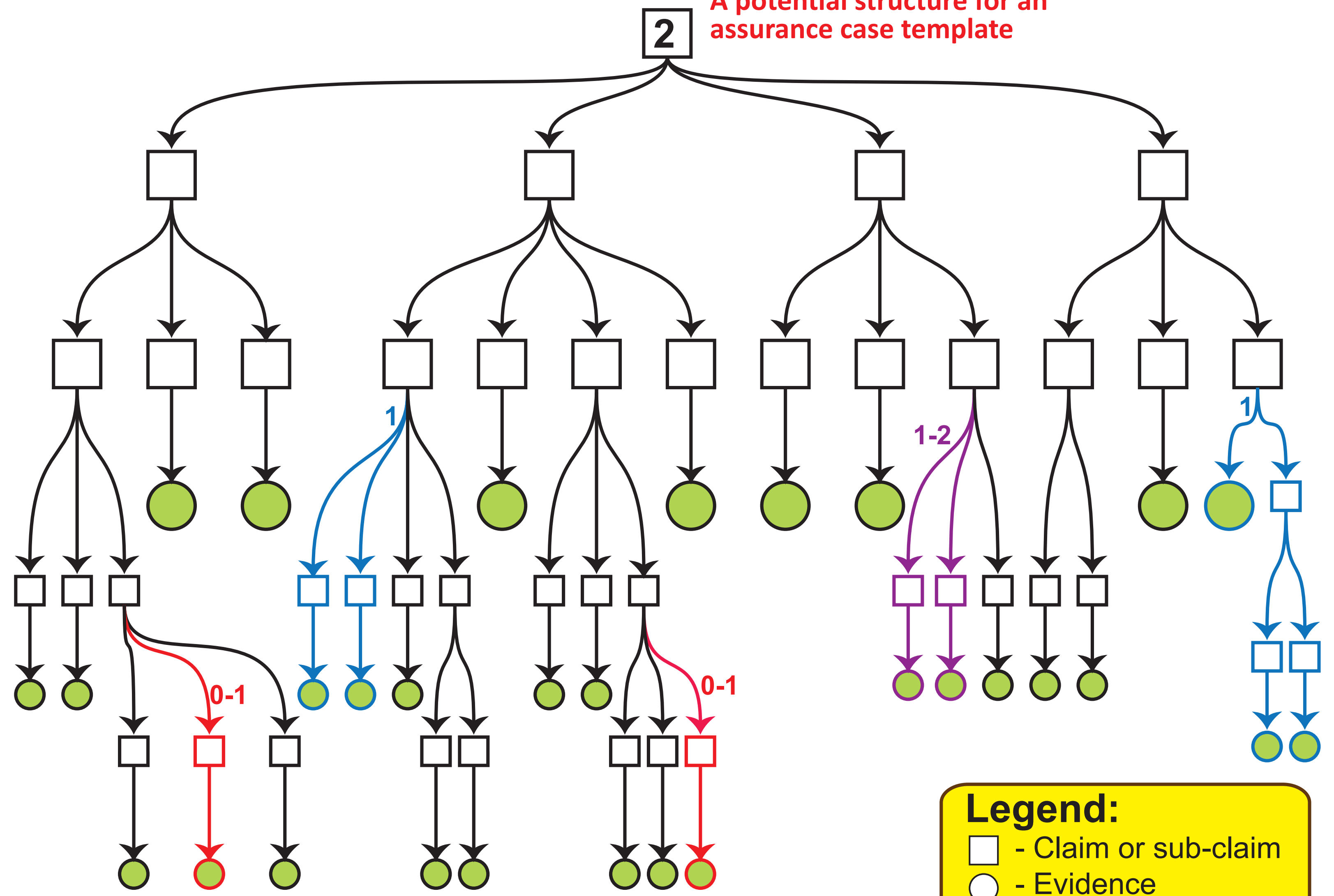
Legend:

□	- Claim or sub-claim
○	- Evidence
A → B	- A is claim B is premise

acceptance criteria on evidence required specified in the template

I Do Not Think So – We Can Do This

A potential structure for an assurance case template



Legend:

□	- Claim or sub-claim
○	- Evidence
A → B	- A is claim B is premise

acceptance criteria on evidence required specified in the template

Characteristics of an Assurance Case Template

- Strategies/Justifications and Argument(s) are explicit
- Contexts are explicit
- Assumptions are explicit
- Evidence to directly support each “bottom claim”
- *Explicit acceptance criteria for evidence to guide content of evidence “nodes”*
- *Argument structures that deal with optional claim-evidence paths*

Notation: *Red and italics indicates that this may only be necessary for an assurance case template*

Build Structures Robust With Respect to Anticipated Change

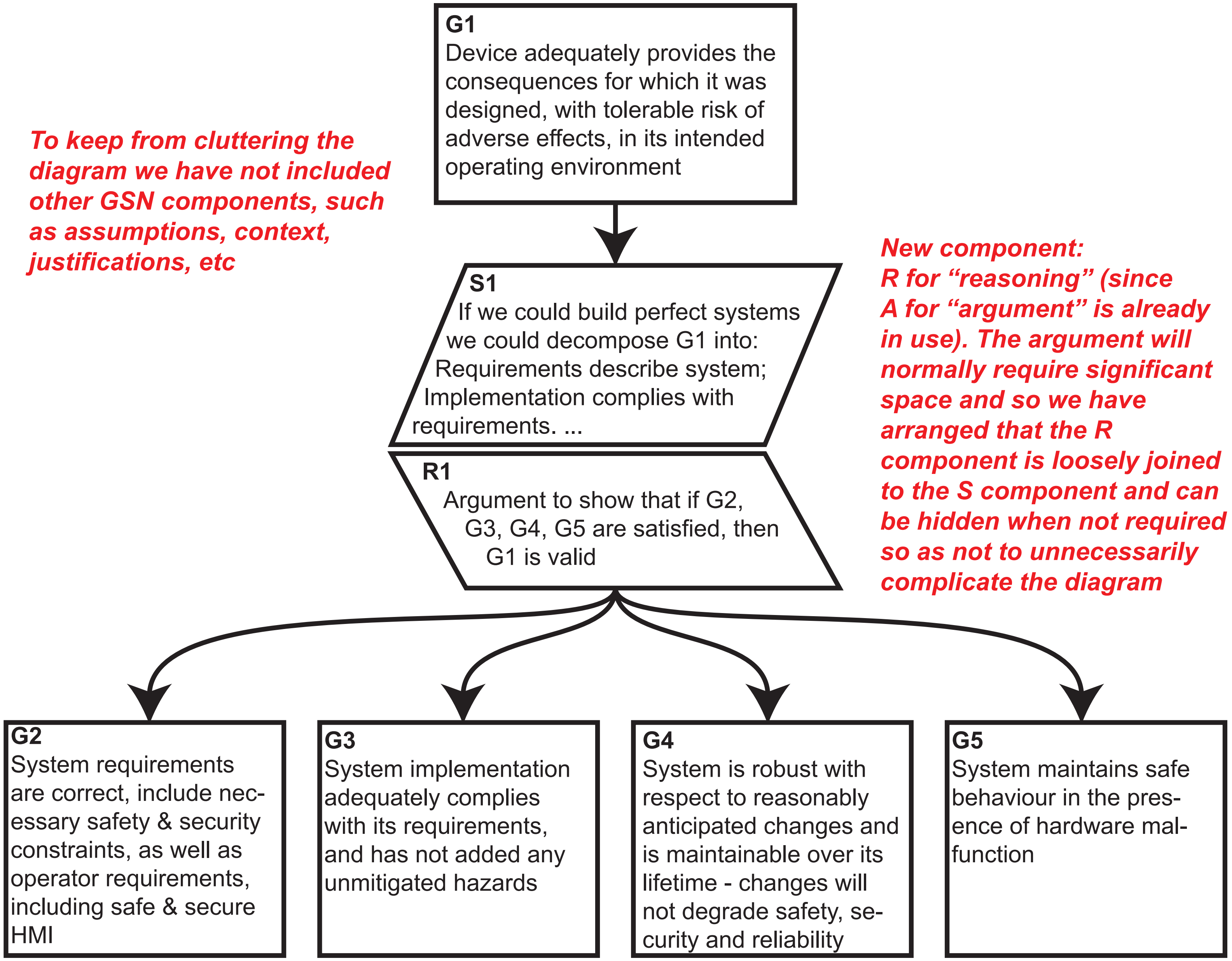
- This is extremely important in general – and may prove to be just as important for templates
  - At the moment we do not seem to have (any) good rules that govern the structure of an assurance case – structure meaning the argument
  - Information Hiding works like magic in modular software design – can we do something similar for assurance cases?
  - The reason we have not done anything like this is probably that we have to cope with safety being a global system property – and those unknown unknowns are a genuine challenge

Some Ideas for Robust With Respect to Anticipated Change

- It should be possible to structure the upper levels of the assurance case so that it is stable for a broad set of products – try to move product specific entities lower down in the structure
- Try to make paths independent of each other – paths are determined not only by argument, they are also determined by product features and related properties – context and assumptions, for example – (this does not guarantee non-existence of emergent behaviour)
- Can acceptance criteria help achieve robustness?

A Robust Top Level

To keep from cluttering the diagram we have not included other GSN components, such as assumptions, context, justifications, etc

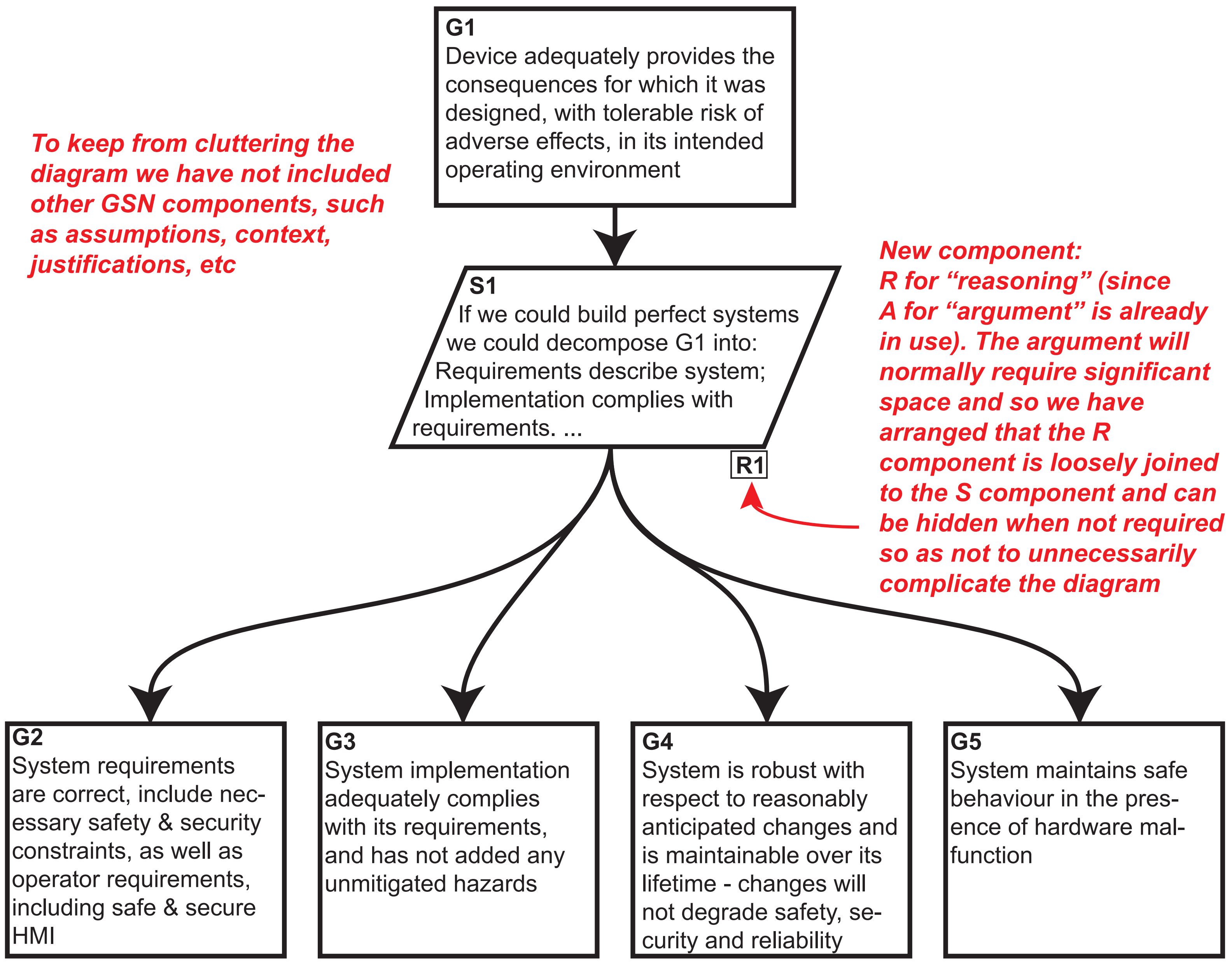


G1
Device adequately provides the consequences for which it was designed, with tolerable risk of adverse effects, in its intended operating environment
S1
If we could build perfect systems we could decompose G1 into: Requirements describe system; Implementation complies with requirements. ...
R1
Argument to show that if G2, G3, G4, G5 are satisfied, then G1 is valid
G2
System requirements are correct, include necessary safety & security constraints, as well as operator requirements, including safe & secure HMI
G3
System implementation adequately complies with its requirements, and has not added any unmitigated hazards
G4
System is robust with respect to reasonably anticipated changes and is maintainable over its lifetime - changes will not degrade safety, security and reliability
G5
System maintains safe behaviour in the presence of hardware malfunction

New component: R for "reasoning" (since A for "argument" is already in use). The argument will normally require significant space and so we have arranged that the R component is loosely joined to the S component and can be hidden when not required so as not to unnecessarily complicate the diagram

An Assurance Case Template for X

To keep from cluttering the diagram we have not included other GSN components, such as assumptions, context, justifications, etc



G1
Device adequately provides the consequences for which it was designed, with tolerable risk of adverse effects, in its intended operating environment

S1
If we could build perfect systems we could decompose G1 into: Requirements describe system; Implementation complies with requirements. ...

New component: R for "reasoning" (since A for "argument" is already in use). The argument will normally require significant space and so we have arranged that the R component is loosely joined to the S component and can be hidden when not required so as not to unnecessarily complicate the diagram

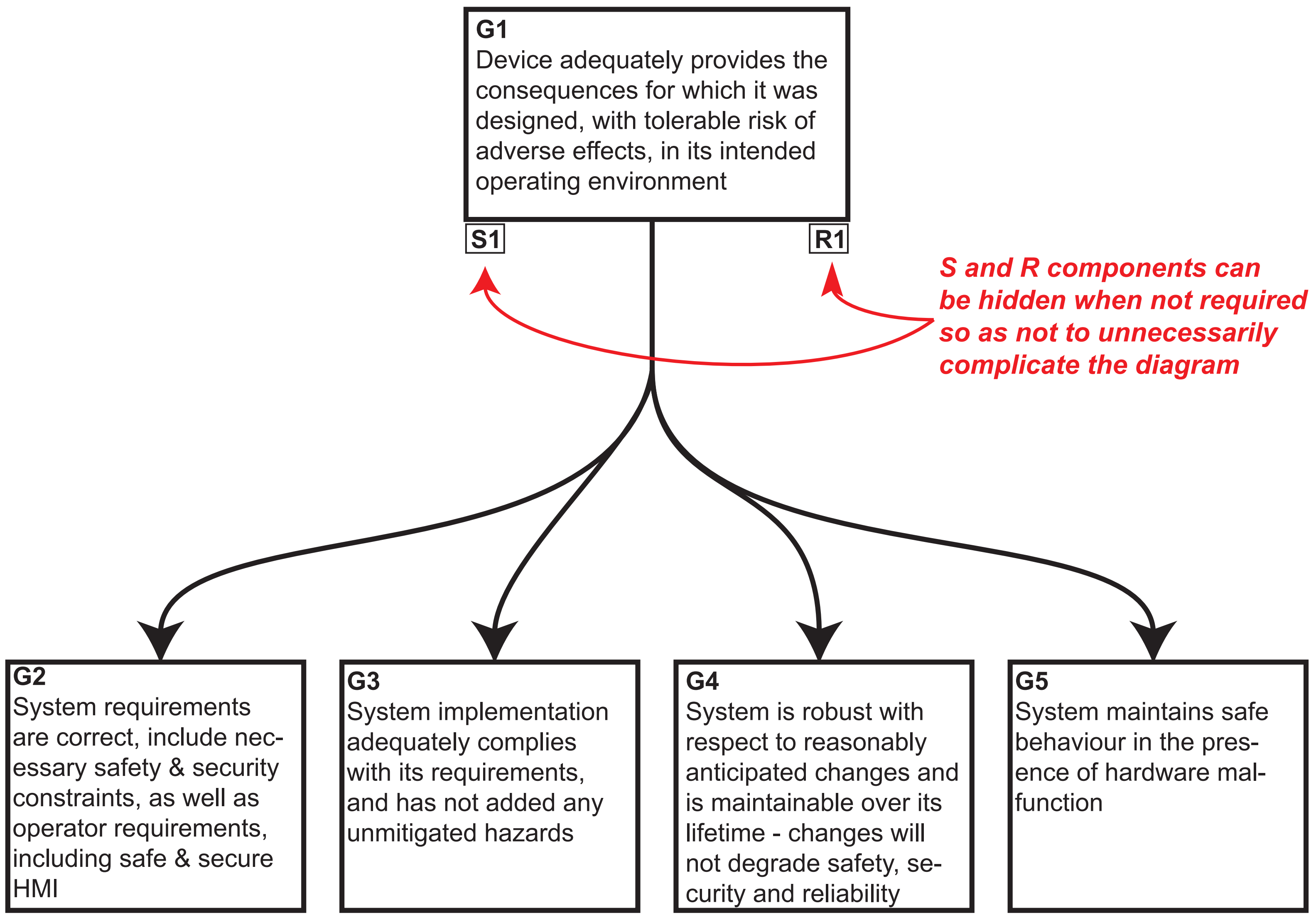
G2
System requirements are correct, include necessary safety & security constraints, as well as operator requirements, including safe & secure HMI

G3
System implementation adequately complies with its requirements, and has not added any unmitigated hazards

G4
System is robust with respect to reasonably anticipated changes and is maintainable over its lifetime - changes will not degrade safety, security and reliability

G5
System maintains safe behaviour in the presence of hardware malfunction

An Assurance Case Template for X


G1
Device adequately provides the consequences for which it was designed, with tolerable risk of adverse effects, in its intended operating environment
S1
R1
S and R components can be hidden when not required so as not to unnecessarily complicate the diagram
G2
System requirements are correct, include necessary safety & security constraints, as well as operator requirements, including safe & secure HMI
G3
System implementation adequately complies with its requirements, and has not added any unmitigated hazards
G4
System is robust with respect to reasonably anticipated changes and is maintainable over its lifetime - changes will not degrade safety, security and reliability
G5
System maintains safe behaviour in the presence of hardware malfunction

Arguments!

- The argument about arguments seems to be heating up – and that is probably good
  - Proponents of inductive reasoning in all sorts of ways, some of them completely bottom-up, starting from the evidence level
  - Proponents of mixed reasoning – deductive and inductive
  - Proponents of defeasible reasoning
- What is obvious though is that most current assurance cases do not have an explicit argument at all!
  - Example: “Argue over mitigation of hazards”
  - So, a proof that there are infinitely many primes could be:
    - “Argue over the product of known primes + 1”
  - We need to do better than this

Why an Assurance Case Template?

- Better than building the assurance case as you develop the system – build assurance in from the start
- And (clearly) much better than building it simply to present a documented case to a regulator

Why an Assurance Case Template?

- We can use such a template as a Standard (John Knight suggested something similar for DO-178 in 2008, and we did at NII Shonan in 2014)
  - Need community involvement and buy-in just as in development of Standards
  - Need to “solve” some of the technical problems
  - Will be for the system – not just the software
  - Can reduce confirmation bias
  - Greater predictability in developing and certifying systems
  - Complex – yes, but more consistent and much more explicit
  - Do we need a confidence case if we have acceptance criteria?
- And I think this just could be our Sanity Clause

Thanks