Contents

CHAPTER 1

New Standards for Trustworthy Cyber-Physical Systems

Alan Wassyng

McMaster Centre for Software Certification, McMaster University, Canada

Paul Joannou

McMaster Centre for Software Certification, McMaster University, Canada

Mark Lawford

McMaster Centre for Software Certification, McMaster University, Canada

Tom Maibaum

McMaster Centre for Software Certification, McMaster University, Canada

Neeraj Kumar Singh

INPT-ENSEEIHT/IRIT, University of Toulouse, Toulouse, France

CONTENTS

Cyber-Physical Systems (CPS) are extremely complex systems that combine components with both physical and cyber interfaces and potentially complex interactions between these parts. They are also often both security and safety-critical if the physical system being controlled can harm people. It is imperative that these systems be developed and certified to be safe, secure and reliable – hence the focus on *Trustworthy Cyber-Physical Systems*. Current safety-critical or high-integrity standards primarily set out objectives on the process, as is typical in much of software engineering. Thus, the acceptance criteria in these standards apply to the development process much more than to the product being manufactured. Having manufacturers use these "good" processes is, indeed, advantageous. However, their use does not guarantee a "good" product, except in a statistical sense. We need to evaluate the quality of the product, not only the process by which it was built [28].

Regulators, for instance the U.S. Food and Drug Administration (FDA), have been dissatisfied with the frequency of the recalls of many of the products evaluated in such a process based regime. Of course, we can, and should, make our standards specify the product-focused evidence that is required, as well as acceptance criteria for this evidence. However, software engineering does not have a good track record in this regard. One approach we can take is to identify critical properties of a system that are necessary in order to achieve tolerable risk regarding the safety, security and reliability of that system. *Assurance Cases* have been gaining traction as a way of documenting such

claims about these critical properties of a system, together with evidence and associated reasoning as to why the claims are valid. Not surprisingly, the FDA have turned to assurance cases to help improve the quality of products submitted for approval [37]. Assurance cases provide one way of documenting a convincing argument regarding the trustworthiness of the resulting system – built on the identification of specific items of evidence, and the satisfaction of explicit acceptance criteria involving both product and process.

We have been exploring the use of *Assurance Case Templates* that can be used to drive development of a software-intensive critical system. Such a template will also provide explicit guidance on an effective assurance case for a specific product within the template's identified product scope. We believe that a *product-domain specific template* can serve as a standard for development and certification of a CPS in that specific *product-domain*.

We have two distinct user groups in mind in this endeavour:

- The developers and certifiers of safety-critical CPS
- Standards developers in this community.

We further believe that assurance case templates can be much more effective than the standards we have now, for both groups. It is probably quite easy for readers to realize why this should be true for the first group – developers and certifiers. However, it may not be so readily obvious for the second group – standards developers. If we are correct, and it is possible to build product-domain specific assurance case templates suitable for use as standards, then the structure of the assurance case template will help standards developers in a number of crucial ways, described more completely later in this chapter. A distinct benefit to standards developers is that the assurance case structure actually guides development of the standard such that we (can) have a better understanding as to why compliance with the standard will result in a high integrity system. This is the way in which we envisage standards developers using such templates. There is another way. Perhaps it would be politically more palatable, but technically not as satisfying. This would be to use an assurance case template as a guide in creating a more traditionally organized standard.

This chapter presents the motivation for the use of assurance case templates as standards by first describing the role of standards (Section 1.1.1), and the shortcomings of current standards (Section 1.1.2). It also includes a discussion of the basic concepts involved in assurance cases (Section 1.2), what an assurance case template is (Section 1.3.1), essential components and concepts of such a template (Section 1.3.2), and the problems Section 1.4.2) and advantages of an assurance case template based standard (Section 1.4.3).

1.1 STANDARDS-BASED DEVELOPMENT

1.1.1 The Role of Standards

The construction of a standard is a community effort that uses state-of-the practice knowledge to describe requirements on the development process and (hopefully) the product that will be developed through application of the process. In any industrial domain, we can typically find a number of international standards that fulfill this role. In addition, if the domain is regulated, such as the nuclear and medical device domains, a government appointed regulator will likely issue regulatory requirements that may reference international standards. In many regulatory jurisdictions, compliance with relevant standards and regulatory guidelines is a necessary prerequisite to obtaining approval to market that device.

A useful definition of a standard is given by the ISO:

Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose.

There are at least two important points to note about this definition. The first is that it is an "agreement", and the parties to the agreement are members of the community involved in manufacturing, developing and certifying the products, processes, services or notations governed by the standard. The second point is that the goal of the standard is to ensure that the entities governed by the standard are "fit for purpose". This is especially pertinent to our discussion when "fit for purpose" includes safety and security attributes.

There are many different kinds of standards, and for the remainder of this chapter we are going to constrain our discussion to standards that govern the development of safety-critical CPS/products. We have seen that standards have the potential to help developers/manufacturers by documenting best practice as determined through community consensus. Of course, the assumption here is that the standard is "good", truly reflects best practice, and is written so that it is reasonably easy to understand and follow. Additionally, standards promote common understanding amongst the various stakeholders – manufacturers, suppliers and consultants, and certifiers/regulators. It is true that many excellent products are built through compliance with current standards.

However, to be truly useful, a standard should include acceptance criteria on the process and/or product that will help to reduce variance in conformance to critical properties of the product, such as safety, security and reliability. Our observation is that this is woefully lacking in most current standards. In general, there are additional problems and limitations with current standards that affect how useful they are in governing the development and certification of CPS, and other safety-critical software intensive systems. In the following section we discuss potential weaknesses in current standards as motivation for doing something different.

1.1.2 Challenges with Standards

Reference [38] described eight potential problems with current standards. These were:

- Most software related standards are primarily process based;
- Outdated standards [3];
- Complex and ambiguous standards;
- Checklist based completeness;
- Lack of design and technical detail [21];
- Excessive documentation; and
- Not adopted by small companies [1].

A similar list was compiled by Rinehart et al [31]:

- Unknown assurance performance (same as "primarily process based");
- Lack of flexibility;
- Development cost; and
- Maintenance cost.

All of these potential deficiencies impact our ability to consistently develop, manufacture and certify CPS that are safe, secure and reliable. However, in our opinion, by far the most important of these is ***"Most software related standards are primarily process based"***, and it is on this point we now focus our attention.

There have been a number of papers written on product versus process based certification [16, 28], and the role each plays in the confidence we have in the quality of the resulting system. The process requirements embedded in standards are not a guarantee that the resulting product will exhibit the desired level of reliability, safety and security. The reasons for choosing those requirements are statistical, in that we have historical evidence (mostly anecdotal) to show that a certain percentage of products developed using those process requirements were successful. We think of these requirements as being derived from principles, such as stepwise refinement, but such principles are still only supported by a statistical argument as to their effectiveness for any one specific system. At best therefore, we can hope that there is a reasonable/good chance that the product will be satisfactory. Of course, there are required processes that should result in safe and secure products – take hazard analysis for example. The fact that our process based standards are also not very prescriptive usually diminishes the reliance we can place on such expectations.

If we want to use process as a basis for our standards, there are a couple of approaches we can take in order to make the outcomes more predictable.

- We can specify more acceptance criteria on aspects of the process, and more precise acceptance criteria on the product.
- We can specify more detailed process steps. Vague or overview process requirements will never be enough to affect the outcome of the process to the degree we need.

Without requirements on the product as well as the process, we are really not able to claim with any certainty that our developed systems are reliable, safe and secure. In practice, therefore, successful companies sometimes impose their own outcome based requirements on specific processes in the development life-cycle. An obvious example of one area in which we have done a little better is testing. Some standards at least have requirements not only on the process, but also on the outcome of the process. The civil aviation standard DO-178B (and now DO-178C) is a good example in this regard, with its mandating the use of *Modified Condition/Decision Coverage* [32, 33]. This may not have been by intent, but it turns out that this test coverage criterion itself also imposes requirements on the outcome of the process.

Other benefits of specifying requirements on the outcome of a process are that:

- It informs developers how the acceptability of their products will be judged; and
- It results in consistency of assessment when different assessors evaluate the outcomes, so the acceptability is less prone to be a result of the views of an individual assessor.

In the end, we need to identify what it is specifically about process based standards that is hampering us in achieving more consistent and higher quality outcomes. We believe that there are two essential elements that are missing in most current standards:

1. Identification of *evidence* that must be produced (the evidence has to relate to product as well as process entities); and
2. *Acceptance criteria* for the evidence produced to be regarded as valid, enabling us to use the evidence to argue that the CPS has the desired properties.

1.2 ASSURANCE CASES

Assurance Cases are based on Safety Cases [2], and safety cases have been in use for over 50 years in the United Kingdom. An assurance case presents a *structured argument* in which the developer of a product makes a claim regarding critical properties of the product (e.g., safety, security, reliability), and then systematically decomposes the claim into sub-claims so that, eventually,

the lowest level sub-claims are supported by evidence. There are a number of notations and tools for assurance cases, the most popular notation being *Goal Structuring Notation (GSN)*, developed by Tim Kelly [25]. Figure 1.1 shows what an assurance case may look like, represented in GSN. The major components in a GSN diagram [15] are:

- *Assumptions* – identified by 'A' followed by a number;
- *Contexts* – identified by 'C' followed by a number;
- *Goals* – represent claims and are identified by 'G' followed by a number;
- *Justifications* – explain why a strategy was chosen and is appropriate, and are identified by 'J' followed by a number;
- *Solutions* – represent evidence, identified by 'Sn' followed by a number;
- *Strategies* – explain why and how a claim was decomposed into specific sub-claims, and are identified by 'S' followed by a number.

A recent Technical Report by Rushby [35] is an excellent description of the history of assurance cases, current assurance case practice, and also delves deeply into the essential characteristics of assurance cases.

1.2.1 The Role of an Assurance Case

The traditional assurance case was designed to document a structured argument that some critical properties of a system, product or process are satisfied to a tolerable level, for the purpose for which they were constructed. Its primary role was thus seen to be as providing a believable demonstration of compliance. An assurance case that assures safety for a product, for example, must demonstrate that the product, when used as intended, in its intended environment, will function with tolerable risk of harm to anyone, over its lifetime. That role is likely to remain a crucial role for assurance cases for many years to come. In the past few years, an additional role has slowly been emerging – that of driving development of the product, for example, in such a way that the resulting product will satisfy, to a required level, the critical properties of interest, and this assurance is documented by way of the assurance case [11].

1.2.2 State of the Practice

One of the problems researchers have in the field of assurance cases, is that many of the real, industrial assurance cases are proprietary. This makes it difficult to ascertain exactly what the state of the practice is. It also makes it difficult to back up any claims we make regarding the state of the practice. We have seen real, industrial assurance cases and formed an opinion about the state of the practice. We have also taken note of what our colleagues have

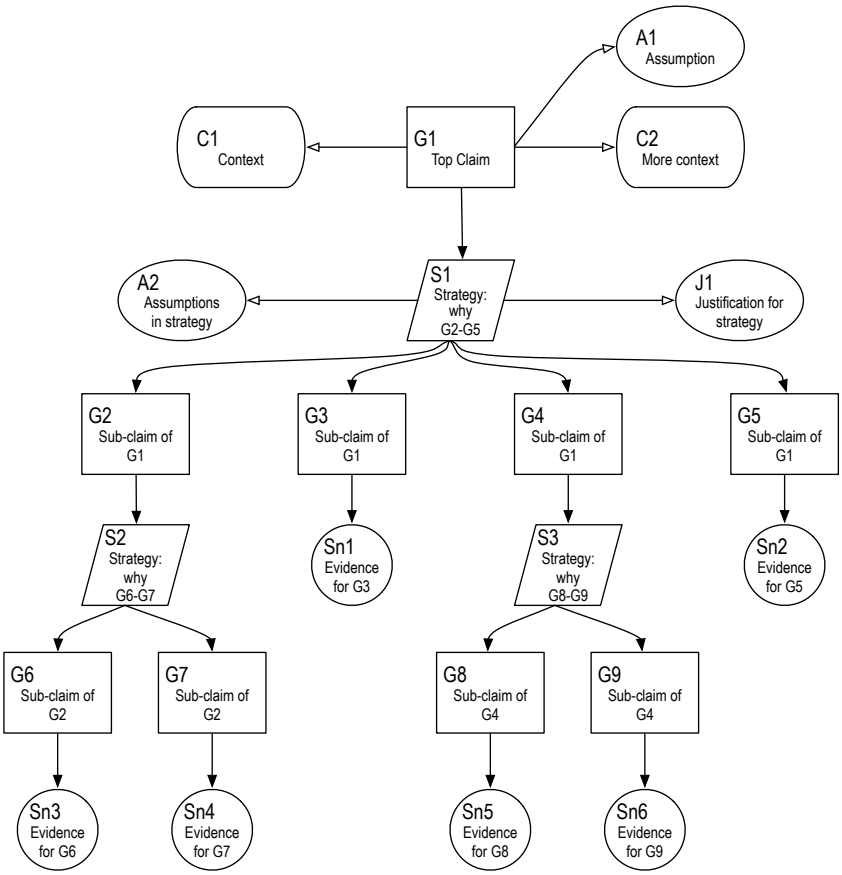

FIGURE 1.1 Assurance Case Structure in Goal Structuring Notation

said about the cases they have been able to examine and evaluate. However, we still cannot cite specifics and cannot show extracts from these proprietary assurance cases.

We take "State of the Practice" to mean what most practitioners do, day-in and day-out. It is likely not to be "best practice", especially in a relatively new field, since there will be relatively large variations in practioners' knowledge and skills. An excellent description of current practice for assurance cases in aviation is given in [31].

Our impression of industrial level assurance cases is that:

- *Improved over the past 5 years:* The concepts and basic idea behind an assurance case seem to be better understood and more widely spread now. In earlier years we even saw one submitted assurance case that had a top-level claim equivalent to *"we followed an accepted process"*.

- *Developed for certification:* In spite of the fact that some companies and many researchers recognize the need to develop the assurance case in tandem with system development [11], the cases we have seen were clearly constructed late(r) in the development cycle. In discussion with manufacturers, the development of the assurance case is seen as a necessary evil, that consumes time and effort, and its purpose is to convince a certifier that the system is of sufficient quality – and safe.

- *Structured to mimic hazard analysis results:* Most assurance cases, since they all had safety concerns, have a top-level structure based on hazard mitigation. We do not believe that this is the best way to structure assurance cases. Some of our reasons are presented in Section 1.3.7.

- *It is not clear whether the precise nature of the evidence is predetermined:* In most assurance cases the evidence that supports a terminal claim is simply presented, and it is not clear why that specific evidence is used, or why it supports the claim.

- *No acceptance criteria:* Presumably because the cases we have seen or heard about were created post development, we are not aware of any attempt to define acceptance criteria for evidence. We have seen discussion, after the fact, as to why the evidence is applicable and "good enough". There is significant attention paid to the *confidence* we have in the evidence and other aspects of the assurance case. Acceptance criteria could have a positive effect on some aspects of this confidence.

- *No explicit argument:* It seems that the definition of "explicit argument" currently accepted is:

  - the tree structure of the claim, sub-claim and evidence makes the argument structure explicit;
  - the claim decomposition rationale (described in the strategy in GSN, for example) explicitly informs us as to the nature of the argument.

  Both of these aspects are useful, maybe even essential, but the actual "reasoning" involved in the argument is not explicit. Our experience is that it is never provided.

1.2.3 Improving the State of the Practice

As mentioned above, there has been a rapid improvement in many aspects of the state of the practice as regards assurance cases. However, there are some things we can do that will improve it even more – and they can be done reasonably easily and thus, soon.

- *Prepare the assurance case before developing the system:* Critical properties that we want in our system must be designed in – they cannot be added successfully after development of the system [17]. There are many reasons why the assurance case should not be created after system development (or late during system development). Two primary reasons are:
  - Safety, security, dependability have to be built-in to the system from the start, they cannot be bolted on after the fact.
  - Companies complain about the cost and effort involved in constructing the assurance case. They mistakenly believe that they construct the assurance case to get whatever certification they are seeking. It is true that developing the assurance case after the fact is costly and time consuming. It is also a duplication of effort. If the company does its work well, it will have had to plan the development of the system in a way that involves many aspects of what will be repeated in the assurance case. Combining the planning and the documented assurance case right from the start is much more cost-effective. This is explicitly recognized and developed in the work on *"Assurance Based Development"* [11]. As mentioned earlier, some companies already do this, but our experience is that companies considering assurance cases, or who have recently started using assurance cases, are still considering them to be certification hurdles.
- *Use a robust structure that facilitates defence-in-depth:* Defence-in-depth is a long standing principle used to ensure the safety of a system. Its use predates the use of software in critical systems. The idea is that safety must not be dependent on any single element of design, implementation, maintenance, or operation. The decomposition structure we use for the assurance case impacts the ease and stability with which we can make changes to the assurance case, and also influences how easy it will be to plan and document a defence-in-depth strategy for the development process in addition to the system. This is discussed in a little more detail in Section 1.3.7, and Figure 1.5 illustrates a top-level decomposition that we believe will lead to a much more robust structure, as compared with basing the top-level decomposition on hazard mitigation.
- *Decide and document ahead of time what evidence must be obtained:* Creating the assurance case early – preferably before system development starts enables us to plan what items of evidence must be produced to support pre-determined claims. This has a number of advantages, the main one being that it provides direction to the developers. They know ahead of time what their process should produce, whether it be documentation or system components. This, together with acceptance criteria as discussed below, provides direction that process alone simply cannot achieve. It is true that we will come across situations in which

the "desired" evidence cannot be obtained. In these cases alternative evidence has to be substituted. The substitution needs to be noted (just as we would note a deviation from a standard). If the reasoning (argument) is really made to be explicit (see below), then we have a tool whereby we can evaluate whether or not the alternative evidence is sufficient to support the specific claim.

- *Determine and document the acceptance criteria for evidence:* Determining what evidence to produce is one thing. We also need to specify what must be true of that evidence for it to be sufficient in supporting a specific claim. This is discussed more in Section 1.3.3.
- *Make the reasoning explicit – not just the claim decomposition structure:* One of the original claims for assurance cases is that their argument structure would be explicit. Technically, that is true. The tree structure showing the claim, sub-claim, evidence is explicit, at least it is in the graphical notations for assurance cases. However, the structure leaves out essential aspects of the argument, and we believe that the reasoning aspects of the argument help us evaluate the validity of the claims in a way that the decomposition structure on its own cannot do. At the present time there is no real agreement on how to perform and structure the reasoning, so we are concerned that in terms of a relatively quick benefit, if we wait until there is consensus on how to perform the argumentation, we will have squandered an opportunity to improve assurance cases in the short term. We propose that, at a minimum, whenever a claim is decomposed into sub-claims or is supported directly by evidence, the assurance case must include a strategy (why and how the claim was decomposed into specific sub-claims), a justification (why the strategy was chosen and is appropriate), and the reasoning (that the claim follows from its premise – the sub-claims or direct evidence). A partial example showing strategy and reasoning is shown in Figure 1.5 in Section 1.3.7.

1.2.4 Aiming for State of the Art

We believe that the items above can be put into general practice now, with a corresponding improvement in the state of the practice for assurance cases. There are two additional items that we see as longer term, that will raise the state of the practice to tomorrow's state of the art. The first of these is to get some consensus on how to perform argumentation in assurance cases in such a way that we can dramatically improve the soundness of the argument, i.e., all the claims are valid and all the premises are true. The second is the evaluation of confidence in the assurance case.

1.2.4.1 Argumentation

Arguments supporting assurance should ideally be formal reasoning in some appropriate formalism defined for the purpose. This is because it is only when the argument is formalized in this way can we be sure that the reasoning is sound and that we have made no mistakes or omitted necessary details. Of course, we may not have to or even want to present the argument completely formally, but use a style similar to that used by mathematics: a rigorous approximation to a completely formal proof, that can, if necessary, be turned into a formal proof.

There then arises the question of what the appropriate formalism looks like. It is commonly accepted that some aspects of such reasoning in assurance cases will not be expressible in conventional logics, like first order logic. The ideas of Toulmin [36] have been used as a departure point for characterizing the kind of reasoning that might be appropriate. Toulmin's argument falls into that group of reasoning known as *inductive reasoning*, in which we cannot provide *"proofs"* in the same way that we can in *deductive reasoning* [35]. Toulmin proposed a scheme for what rules in an argument might look like. That is, he proposed a template that should be general enough to represent the form of any rule used in an argument. So, a logic appropriate for reasoning in assurance cases would be populated with reasoning rules that may conform to Toulmin's scheme.

1.2.4.2 Confidence

One important aspect of Toulmin's scheme is the uncertainty associated with the elements of the rule scheme. In conventional logics, if one accepts the premises of some inference/argument step and one applies a rule to produce a conclusion, then one can have absolute confidence in that conclusion. Confidence is not talked about as there is no uncertainty associated with any of the premises, the applicability of a rule of inference or the resulting conclusion. Toulmin argues that in certain domains, like law, science, and we would claim in assurance cases, all of the three elements above have levels of uncertainty associated with them, which we conventionally refer to as confidence.

Over the past few years, *confidence*, as it applies to assurance cases, has become an important and much discussed topic [4, 7, 9, 10, 13, 14, 43]. What confidence are we, as a community, attempting to define, estimate and even quantify?

The answer is: *confidence in the validity of the top-level claim.* This is clearly the primary concern of the assurance case (if it is not, we have the wrong top-level claim), and we want to be able to evaluate, sensibly and as objectively as possible, how certain we are that the claim is valid. Note that confidence in the top-level claim will typically be governed within the context of the appropriate *safety integrity level*, and its associated *integrity level claim* [23]. To this end, we can try and evaluate our confidence in two items as shown below.

- *Confidence in the argument:* The argument is supposed to be explicit, but even if it is not, the claim, sub-claim decomposition and evidence does describe an argument – it is simply that we have to discover it rather than be told explicitly what it is in the assurance case itself.

- *Confidence influenced by evidence:* The purpose of an item of evidence is to support the parent sub-claim directly connected to the evidence. Confidence is thus related to how well the evidence supports the claim. For example, if the terminal sub-claim is: *"The system behaviour as implemented is adequately*[1] *similar to the documented required behaviour"*, what can we say about our confidence in the assurance case in the two examples of evidence that follow? 1) System Test Reports; and 2) Mathematical Verification Report.[2] Confidence here will be influenced by the coverage criteria used for the tests; the specific test coverage achieved; the number of failed tests at the time of installation; the fact that a mathematical verification was performed; and the number of residual discrepancies that remain at the time of installation.

However, even if we can establish our confidence in the two items above, we have no generally accepted theory or any other means of using this confidence to ascertain the confidence we have in the top-level claim.

1.3 ASSURANCE CASE TEMPLATES

In the discussion that follows, readers who know something about assurance cases may conclude that we simply mean an *assurance case pattern* when we talk about an *assurance case template*. We do not. At least, we do not believe that we do. The real difference between a *pattern* and a *template* is one of completeness. Our observation of assurance case patterns is that they are assurance case *design artifacts* that can be instantiated for a specific situation, and that are used within an overall assurance case. The assurance case template, is a complete assurance case, in which claims are specialized for the specific situation, and the evidence for terminal claims is described together with acceptance criteria for the evidence, and as development progresses the evidence is accumulated and checked against its acceptance criteria. The concepts are, indeed, very similar – but these templates are not the same as current patterns. Patterns could prove to be useful within assurance case templates.

[1] We assume that "adequately" is defined for each specific use of the word, within the assurance case.

[2] In general we expect the evidence to be much finer grained than presented in this example.

1.3.1 What is an Assurance Case Template?

Probably the best way of defining or describing an *assurance case template* is to show an overview example of the process that led us to believe that assurance case templates could be a productive and important step in being able to develop reliable, safe and secure CPS.

To start, let us assume we have successfully developed a product (system) and also documented an effective assurance case for the product. This product is creatively called *"Product 1"*, and the assurance case decomposition structure is shown in Figure 1.2. In the interests of clarity, assumptions, contexts, strategies, etc., are not included in the figure, and the "1" in the top-level claim box simply indicates that this is the top level-claim and assurance case for *Product 1*.

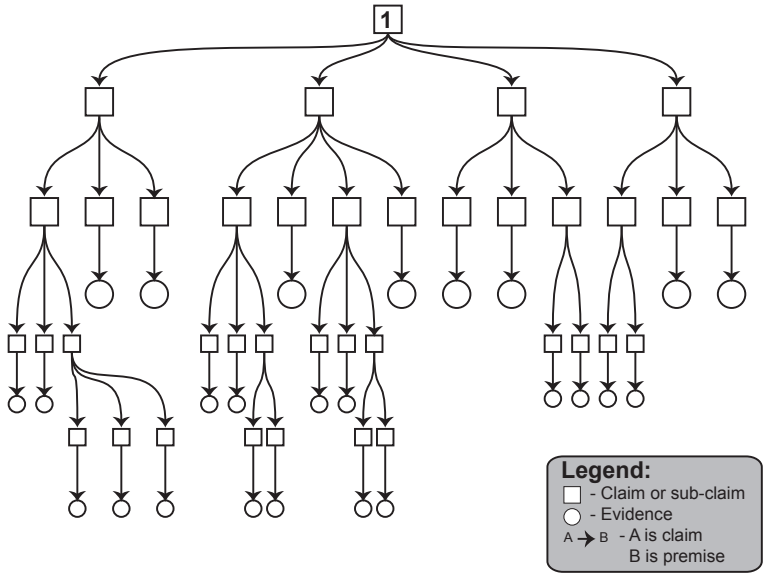

FIGURE 1.2 Assurance Case Structure for Product 1

Now let us further assume that we follow up our success by developing another product, which we call *"Product 2"*. *Product 2* is a product different from *Product 1*, but is related in that it is within the same *product domain* as *Product 1*. For instance, perhaps *Product 1* is an insulin pump, and *Product 2* is an analgesic infusion pump. Or, perhaps *Product 1* is a car and *Product 2* is a mini-van. We again document an assurance case for *Product 2*, and we are so expert at developing assurance cases that the new assurance case differs from that for *Product 1* only where absolutely necessary. The assurance case for *Product 2* is shown in Figure 1.3.

Figure 1.3 also highlights the differences between the two assurance cases

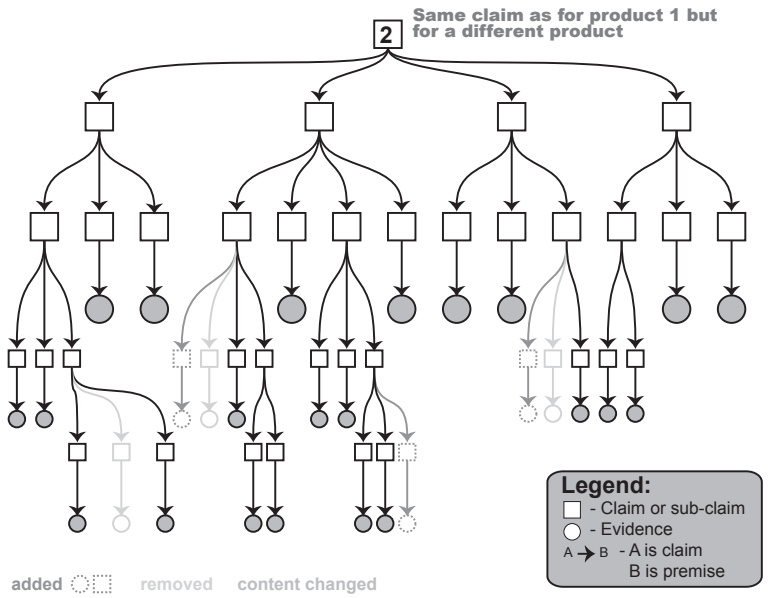

FIGURE 1.3 Assurance Case Structure for Product 2

by explicitly showing which components have been added, removed or modified in the assurance case for *Product 2* as compared with the assurance case for *Product 1*. Now, consider what the figure would look like if we developed *Product 2* before *Product 1* and then highlighted the differences in the assurance case for *Product 1*. It should be clear that components added in the case for *Product 2* would be shown as components removed in the case for *Product 1*. In other words, the difference between "added" and "removed" is one of time – it depends on whether (the assurance case for) *Product 1* is developed before or after (the assurance case for) *Product 2*.

So now we get to the heart of the idea of an assurance case template. We want to develop an assurance case for both *Product 1* and *Product 2* – before we actually develop the products themselves. Of course, since an assurance case must be product/system specific, we cannot actually build a single assurance case that precisely documents assurance for both products. What we can do is build a template that has a structure that will cater for both products, but in which the content of components will need to be different for the two products. If our template has to handle only the two products, *Product 1* and *Product 2*, then a template that may achieve this is shown in Figure 1.4. The idea here is that claims and sub-claims can often be parameterized, similarly as is done in *assurance case argument patterns* [6, 44], so that the claim can reflect the specific product the assurance case deals with. The details are not visible in the figure, but we will discuss this aspect of the template later in this chapter.

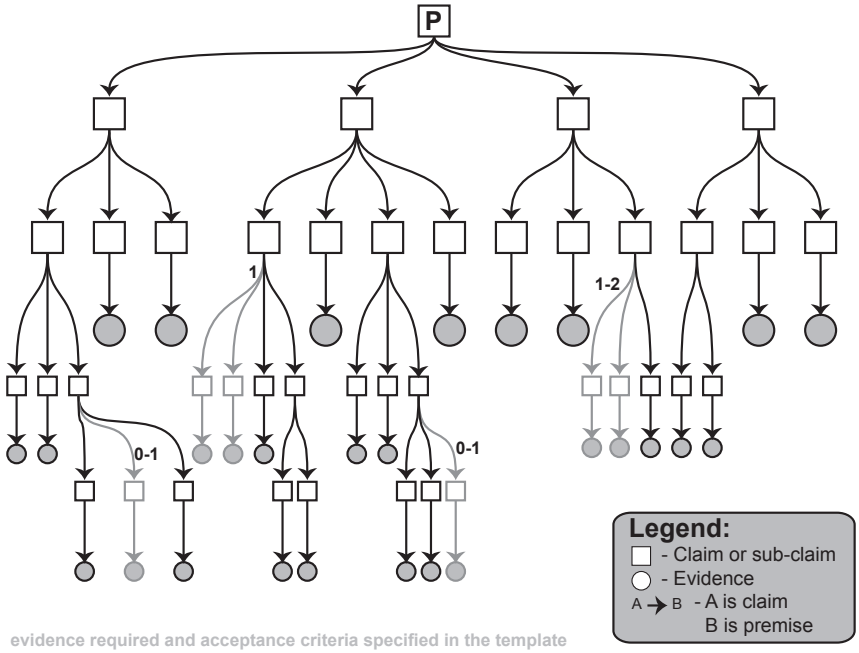

FIGURE 1.4 Assurance Case Template Structure

Figure 1.4 now needs some explanation. There are two specific aspects of the template we discuss at this stage.

- *Optional paths:* The paths shown in grey in the figure are optional, depending on the specific product for which the assurance case is being instantiated. The numbers next to the optional paths show the *multiplicity* of the paths.
  - *Optional **0-1**:* This is a single path that may or may not be required for a specific product.
  - *Exclusive-Or **1**:* One of the paths (there can be more than 2) must be instantiated for a specific product.
  - *Non-exclusive-Or **1-n**:* One or more of the paths can be instantiated for a specific product.
- *Evidence nodes:* The actual evidence for products will differ from product to product. That is why all the evidence nodes in Figure 1.4 are shaded. If this were not true we would not need to develop an assurance case at all! Since this is a template, the content of the nodes in the template will not be the actual evidence. What will it be? The answer is simple, and reflects one of the benefits of building an assurance case template. Each evidence node must contain:

- A description of the required evidence.
- Acceptance criteria for that item of evidence, i.e., what must be true of that evidence to raise the level of confidence that the critical properties that the system must have are true.

Not surprisingly, the GSN community, in their work on assurance case patterns, has had to deal with exactly this type of optionality. GSN now includes specific notation to deal with what they have termed "Multiplicity" and "Optionality" [15]. A small solid ball together with a number representing the cardinality is used for multiplicity. A small hollow ball is used for optional (in this case 0 or 1). The notation was further extended to use a solid diamond together with the cardinality number, as a general *option* symbol. What seems to be missing from GSN is the *non-exclusive-Or* option. An example of an assurance case template using this notation is provided in Figure 1.6 in a later section of this chapter. We have used the option symbol together with an "at least n" descriptor to describe the *non-exclusive-Or*.

1.3.2 Characteristics of an Assurance Case Template

The essential characteristics of an assurance case template must also include characteristics of state of the art assurance cases. Why state of the art? Primarily, to reap the benefits of a template.

1.3.2.1 Characteristics of state of the art assurance cases

We believe the following characteristics/components are essential for any state of the art assurance case.

- *Explicit assumptions:* An assurance case applies to a particular system, and we know that we cannot claim to assure any critical property in that system without describing precisely what has to be true about the system. Assurance cases have improved the general approach to system assurance by highlighting the fact that assumptions are first class components of such assurance. They are useful/essential because:
  - Reviewers can determine whether or not the assumptions are practical;
  - They help to constrain the system, and without constraints we usually will not be able to achieve the required quality;
  - They provide an easy opportunity to check that the system does not violate any assumptions;
  - They enable us to evaluate changes in the system in order to determine whether or not the assurance case will have to be modified to account for the changes (even if the changes do not violate stated assumptions, we may need to modify the argument component of the assurance case).

- *Explicit context:* An area where current assurance cases do reasonably well is in the documentation of context for the primary system components. This is important since one of the typical entities described in context is the definition of terms. This may seem trite, but becomes crucial the more precise we try to be – and precision/formality is something we strive for in the case of safety-critical systems.

- *Explicit argument:* One of the benefits "claimed" for safety cases and assurance cases is that they make the assurance argument explicit. Typically this is not true in extant examples we have seen. Most assurance cases present an explicit decomposition of the top-level claim(s) into sub-claims, and eventually at the end of each path of sub-claims, there is an evidence node (or nodes) that "supports' the sub-claim it is attached to. The decomposition of sub-claims is not an argument – it does not provide reasoning that uses the premise(s) to support the (sub-)claim (see Figure 1.4 above). Actually, there is no agreement at this time as to what an assurance case argument should be. Some researchers believe we can provide localized arguments as described above. Others believe that there are some properties, safety is the prime example, that are global to the system, and local arguments (the premises "imply" the claim) ignore emergent behaviour. In this case, the argument would have to be global to the assurance case. Over the years there have been a number of attempts at defining how to perform argumentation (reasoning) in assurance cases [18, 34, 42, 41, 5]. The report by Rushby [35], mentioned earlier, presents an excellent discourse on the different types of arguments for assurance cases. At this time, any explicit argument, of whatever level of formality, even if we find out in the future that it has shortcomings, will be a huge improvement. The sad truth is that, in practice, there is almost no attempt to provide explicit reasoning in assurance cases. Since so many successful projects have been completed without it, do we really need it? Our answer is "yes" – especially long term. The size and complexity of CPS is stretching our ability to produce systems/products of sufficient quality and reliability. The complexity and distributed nature of these systems makes it even more difficult to identify emergent behaviours, and protect against hazards that arise because of these emergent behaviours. Evaluating the safety, security and reliability of these systems is assuming more and more importance, whether it be for external certification (licensing and approval), or internal certification (quality control). Making the argument explicit, so we can evaluate the soundness of the argument more easily and accurately, will help us build systems with adequate quality.

- *Explicit strategies:* Although the strategy is not an argument, it is still useful and important. The idea of the strategy is to explain how the parent claim is decomposed into sub-claims and sub-claim paths that explicitly characterize the safety design aspects of the artifact.

- *Explicit justifications:* Justifications are usually regarded as "nice to have", but they are more than that. Whereas the strategy describes the "how" of the decomposition, the justification describes the "why". The structure and rationale for the decomposition is akin to the module guide in Parnas's *Rational Design Process* [30], and this is precisely what the strategies and justifications, together, should deliver.

1.3.2.2 Characteristics Essential for an Assurance Case Template

- *Adequate descriptions of evidence to be provided:* An important difference between an assurance case and an assurance case template is that, in a template, the specific evidence for a system cannot be included, because the specific system is not known at the time the template is constructed. However, we do (should) know what evidence will be required to support the claims we want to make – we know the claims as well. Why should this be true? The simple fact is that any manufacturer of safety-critical systems sets out to build a safe (reliable and secure) system, and an incredible amount of work goes into planning how to build the system so that it will be safe! This planning uses knowledge accumulated by practitioners through the ages and documented in books and research articles in appropriate fields, as well as in corporate memory. It may not have been framed or even recognized as claims and evidence, but it is – or can be viewed as such. So, what we need for an assurance case template is the explicit documentation of the nature of the evidence that is required for each "bottom-level" sub-claim. This description can include alternatives, since it is sometimes possible to support a claim in a number of ways. In fact, it may be possible to provide more than one item of evidence for a single claim – and that certainly helps to build confidence in the assurance case.

- *Acceptance criteria for evidence:* A description of the kind of evidence required for each terminal sub-claim is necessary, but not sufficient. We also need to describe, as precisely as possible, what must be true of the documented evidence to be sufficient to support a specific claim. The fact that this is not routinely required in assurance cases reflects the fact that most assurance cases are developed after the fact (or at best, during development) to document why a particular system satisfies specific criteria. They are not developed to drive development so that those criteria will be satisfied. There are academic papers that exhort practitioners to build the assurance case as early as possible [24, 12]. However, those papers do not discuss acceptance criteria for evidence. Our ideas regarding an assurance case template were prompted by the work of Knight and colleagues. Their idea of *Assurance Based Development (ABD)* [12] was intriguing, and seemed to reflect many of the principles we believed in.

However, this body of work also does not promote the idea of acceptance criteria.

- *Arguments that cope with optional paths:* As we saw in Section 1.3.1 and Figure 1.4, an assurance case template requires arguments that deal only with logical conjunction. An assurance case template requires arguments that can deal both with logical disjunction and logical conjunction. This should not constitute a real problem, but it is worth noting that the situation has to be dealt with.
- *Why explicit arguments will be crucial:* As stated earlier, we strongly believe that explicit arguments are essential if we are to realize the true potential of assurance cases to help improve the safety, security and reliability of complex CPS. Explicit arguments (reasoning) are even more important in the construction of assurance case templates, simply because the template is predictive rather than reflective. We can consider two different cases:
  - What if we want to use evidence that is not explicitly included in the description of evidence in the template? In this situation it may be that the template may have erroneously omitted the evidence we now want to include. Or, it may be that the developers of the template did consider whether or not to include that evidence, but, for whatever reason, decided not to. In both situations, if we have an explicit argument in the template, we will be able to determine the effect of our potential change in evidence will have on the validity of our claims.
  - What if we want to modify a claim or sub-claim? Since our current knowledge does not adequately inform us as to whether or not it is at all possible to perform incremental safety assurance, modifying claims in a way that requires a change in the reasoning/argument appears to be risky. Well, it may not be quite as bad as it seems. First of all, if the original argument is "bottom-up" and global (as opposed to a set of localized arguments, each constrained in scope to a specific set of sub-claims that support the directly connected parent claim), then we should be in a position to evaluate the effect of the change in terms of how it affects the "top-level" claim(s), and what would be necessary to make the argument "sound" again. Secondly, if the argument consists of a set of localized arguments, then we must have decided that these localized arguments are sufficient for our purpose – at least at this time and stage of knowledge in argumentation for assurance cases. Then we should be able to use the localized argument that encompasses the change to evaluate the effect of the change in terms of how it affects the "parent" claim, and what would be necessary to make the argument "sound" again.

1.3.3 Acceptance Criteria and Confidence

The notion of *confidence* in assurance cases was discussed in Section 1.2.4.2.

Why is it, that with all the research on *confidence* in assurance cases, we hear so little about *acceptance criteria*? The answer to this is quite simple, but sometimes hotly refuted. The focus on *confidence* as compared with *acceptance criteria* is primarily due to the perception that assurance cases are developed after the system/product has been built – or, at best, while the system is being built.

Why are we concerned about this? There is significant research effort on confidence related to evidence. We believe that in dealing with this aspect of confidence, our task would be better defined if we could assume that acceptance criteria for evidence have been defined. The more we can reduce uncertainty in this regard, the better off we are going to be. A consensus of expert opinion used to define acceptance criteria would be of definite benefit.

1.3.4 Dealing with Evidence

We have already stated that, in terms of evidence in assurance case templates, the two most important attributes we have to document in the template for each evidence node are a description of the evidence required to support the parent claim, and the acceptance criteria for the evidence. Related to the first of these, we need to discuss the nature of evidence in an assurance case node. It has become prevalent to simply reference documents produced during development as the relevant evidence. We believe that evidence in assurance case nodes should be much more specific. If we include specific sections of a document as part of the reference, then this can still be achieved by referencing development (or other) documents. This is sometimes difficult when done "after the fact", since the section referenced may contain more than the specific item we wish to refer to. In the case of a template, this should be less of a problem since we will know ahead of time that we will want to reference the specific item.

As far as evidence is concerned, one important thing to note is that the evidence and the assurance case argument are strongly linked, and both are essential to successful assurance cases. Researchers at the University of York have a saying regarding this: *"An argument without evidence is unfounded whereas evidence without argument is unexplained."* [8]. You cannot necessarily change one without making changes in the other. This is always true in any assurance case. For an assurance case template, this dependence is both a potential challenge, and extremely useful. It is a challenge in that changing the evidence will incur a re-examination of the argument, and re-examining the argument can be costly in terms of time and effort. One mitigation in this regard is that the evidence is at the bottom of the argument tree, and, our intuition is that a change in evidence is more likely to have local impact

on the argument. The dependence can also be useful in helping developers understand why that particular evidence is necessary.

1.3.5 Product-Domain Specific

The template we envisage will be product-domain specific. It has to be. A major assumption in this work is that there are classes of products/systems for which an assurance case structure (claims and arguments) exists that handles all the products within a class without *too many* differences. The only possible way that this assumption could be valid is if we restrict our assurance case templates to product-domain specific systems. We do not have proof yet that this is possible. We do have some preliminary results from consideration of infusion pumps – and also preliminary ideas with respect to the automotive domain.

1.3.6 Different Development Processes

In Section 1.3.1, and in Figures 1.2, 1.3 and 1.4, we presented motivation for assurance case templates by describing how we could handle differences caused by product/system differences. There is another kind of difference that we have not yet discussed – differences in the development process. Many differences in the development process can be handled without any change in the structure of the assurance case. For many years, we have argued that even when we promote "product-focused" certification, there is always an assumption of an "idealized development process", as described in [39]. Typically, it is these elements of an idealized process that make their way into an assurance case. However, there is one situation that is definitely different, and may result in requiring a slightly different assurance case structure. If one system is developed using a traditional development life-cycle (especially as regards the software), and another system is developed using a model-driven engineering approach [22], then we can expect differences in the assurance case structure. In addition, the use of two substantially different development processes can/will occur even within a product-domain. Thus, an assurance cases template must be able to handle these differences, and we believe that a template with optional paths as shown in Figure 1.4 will be able to handle them well.

1.3.7 Suggested Structure

Researchers have been trying to work out how to achieve incremental assurance for complex systems, preferably using assurance cases. The difficulty here is that some properties, like safety, are global to the system, and emergent behaviours have implications for the safety of the system. We do not know how to isolate portions of the system in such a way that each portion being "proven" to be safe would imply that the entire system is safe. We do not even know if

this is possible. We surmise that it may not be possible without revisiting the argument in the assurance case. At the same time, we cannot afford for that to be the final answer, since we will not be able to redo complete assurance case arguments every time we make changes to a safety-critical CPS.

In some ways, software engineering research has always grappled with problems like this. Not always concerning safety, and not always with regard to global properties. Throughout the past 50 years we have had a number of challenges in our pursuit to build more dependable software intensive systems. One distinct challenge was how to cope with change in software intensive systems. In the 1970s, Parnas suggested that *"information hiding"* [29] would help us cope with anticipated changes, and that we could then build systems in which dependability would not be degraded when we made these anticipated changes. Information hiding helps us structure software designs so that they are robust with respect to anticipated changes. It is time we put serious effort into determining how to design assurance case structures that are robust with respect to change. This seems particularly apropos with regard to assurance case templates. We will use the top-level decomposition of an assurance case for a medical device, as shown in Figure 1.5, to start this discussion.

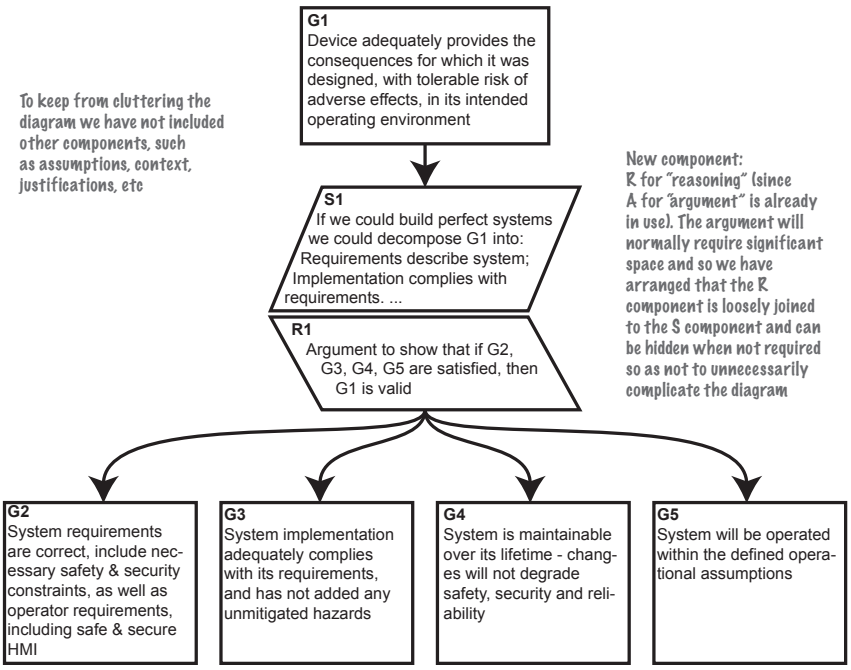

FIGURE 1.5 Robust Top Level of the Assurance Case Template (modified from [38])

The first thing we notice (probably) about the claims G2, G3, G4 and

G5 in Figure 1.5 is that they should not have to be changed no matter what system/product the assurance case template is developed for. Not only is this level robust with respect to change, it also reflects our experience of how we have successfully planned safety-critical systems over more than twenty years. One principle that has occurred to us for designing robust structures for assurance cases, and is supported by what we see here, is to move product specific components of a case as low down in the structure as possible. This is an immediate difference from what we see in many current assurance cases. In most cases we have looked at, the assurance case structure is based very heavily on hazard mitigation, with the hazard analysis providing pertinent evidence. This not only violates the principle we have just espoused, it seems faulty for a number of other reasons.

- *Completeness:* It is impossible to "prove" that any hazard analysis is complete with respect to the hazards in the system. There are many aspects of assurance in which we cannot prove completeness. The way we deal with this is usually two-fold: do as good a job as we possibly can; and have more than one way to achieve a critical outcome. Basing the structure of the assurance case on the structure of the hazard mitigation makes this more difficult. One of the touted benefits of a hazard analysis [27] is that it avoids *confirmation bias*. Ironically, structuring an assurance case primarily by documenting mitigation or elimination of hazards, seems to inject an aspect of confirmation bias in the assurance case.

- *Iterative nature of hazard analysis:* As we progress through the development life-cycle we introduce more hazards, and the simple act of mitigating a hazard may introduce even more new hazards. Since we intend to use the assurance case template to drive development, it is important to be able to convey the iterative aspect of hazard analysis and mitigation.

In our suggested structure, the system hazard analysis is included in the claim-evidence path leading off from sub-claim G2, and it is not the only means whereby we assure safety and security. The actual wording of the sub-claim in G3, *"... and has not added any unmitigated hazards"*, has been worded that way explicitly to help convey the idea that any hazards introduced in the development process must be mitigated. The way in which the usual claim is typically phrased, *"all hazards have been mitigated"* does not convey the same imperative to the developers. Finally the product specific aspects of the hazard analysis are much lower down in the structure.

Another principle that we believe facilitates building assurance case structures that are robust with respect to change is to try and make claim-argument-evidence paths as independent as possible. "Independence" in this context means that, at each "level" of the structure, for each claim at that level, there are no argument specific dependencies between the lower paths that support that claim. This is clearly difficult to do, sometimes not possible,

and maybe difficult to know when you have really achieved that independence. However, we believe it is worthwhile keeping this in mind while deciding on the structure for the assurance case.

These are very basic and obvious ideas for improving the robustness of the structure of an assurance case. What we are lacking right now is something equivalent to the elegance and effectiveness of information hiding as applied to software designs.

1.3.8 Example: Infusion Pumps

In order to illustrate the different kinds of optional paths we foresee for assurance case templates, we have three brief examples drawn from our experience with insulin pumps. The three cases we identified earlier (Section 1.3.1) were:

- *Optional **0-1**:* Some insulin pumps include a built-in light that helps the user read the pump's screen in the dark. This has impact on the assurance case in a number of ways, including access to data from the pump, and battery usage. These paths in the assurance case would not exist at all if the pump did not have this feature.

- *Exclusive-Or **1**:* Some pumps use a standard Luer connector for their infusion sets, while others do not. The pumps definitely need to use some sort of connector for the infusion set, and there are different pros and cons depending on what connector is used. Use of a Luer connector is not mandatory. The assurance case has paths that depend on the connector since it affects both the safety and effectiveness of the delivery of insulin to the patient. However, these paths are different because of the different pros and cons of the connectors. The template therefore will include a number of paths, depending on the number of connectors likely to be used in commercial pumps. Only one of those paths will apply for a particular instantiation of the template. (In the following section we will discuss how to deal with the situation where a new connector becomes available that is not included in the current assurance case template.)

- *Non-exclusive-Or **1-n**:* Some pumps allow you to input glucose readings directly from an associated meter, or to input those readings manually. The assurance case then has to handle the situation where both options are present in a single pump, as well as the situations where only one of the options is available for a specific pump. Note that this is not a situation where redundancy is being used to increase reliability. The pump that allows both modes of input does so for ease of use – only one mode is used at any one time. The arguments for safe, secure and effective are different in the two modes. An easy difference to note is that there is (probably) less chance of a security problem arising when the manual mode is used.

1.4 ASSURANCE CASE TEMPLATES AS STANDARDS

There are two major reasons we considered using assurance case templates as development standards. The first reason was that it seemed to be an obvious extension of using an assurance case template to drive development of a system, and then use the resulting assurance case as a basis for certification. The second reason was the research over the past few years in which multiple efforts have been made to represent existing standards by assurance cases [20, 19].

While exploring assurance case templates and their potential to replace standards, we learned about a talk given by Knight [26], in which he outlined the use of an assurance case as an alternative to the avionics standard DO-178B. This reinforced our belief that we are on the right track. Our assurance case template is different from the *fit for purpose assurance case patterns* used in ABD. It does have much in common with Knight's ideas expressed in [26].

The assurance cases we build are targeted at assuring the safety, security and reliability of the complete system, not only the software, for instance. There are likely to be aspects of existing standards that do not currently fit easily within the type of assurance case template we envisage. Since at this stage we do not know what they are, we do not know whether or not they are really needed. Our intent is to explore how they can be incorporated. In the meantime, we need to realize that the assurance case template may need to be supplemented by "extra" information currently included in various standards.

1.4.1 Using an Assurance Case Template as a Standard

If we have done our work effectively, using the assurance case template as a standard should be reasonably straightforward – most of the time. The template will naturally dictate certain aspects of the development process, and system development together with the instantiation of the template proceed in tandem. As instantiation proceeds, the resulting assurance case documents the specific system being built. As usual, project planning determines how to comply with relevant standards, including how to comply with the relevant assurance case template. During this period we may find that the template does not cover a situation that we want to include. In such a situation we can modify the assurance case template we will use for our specific development. It is important though, that we carefully document how we have deviated from the community developed template, just as we would currently note any deviations from any other standard. Figure 1.6 show how we can achieve this, by using notation that highlights entities that have been modified, and entities that are completely new, i.e., were not included in the template at all.

Figure 1.6 also shows how the optional paths included in Figure 1.4 are now described using GSN, as discussed in Section 1.3.1. For example, the two optional paths are now described by including small hollow circles in the arrows. Also, the exclusive-Or path is shown with a solid diamond, and a "1

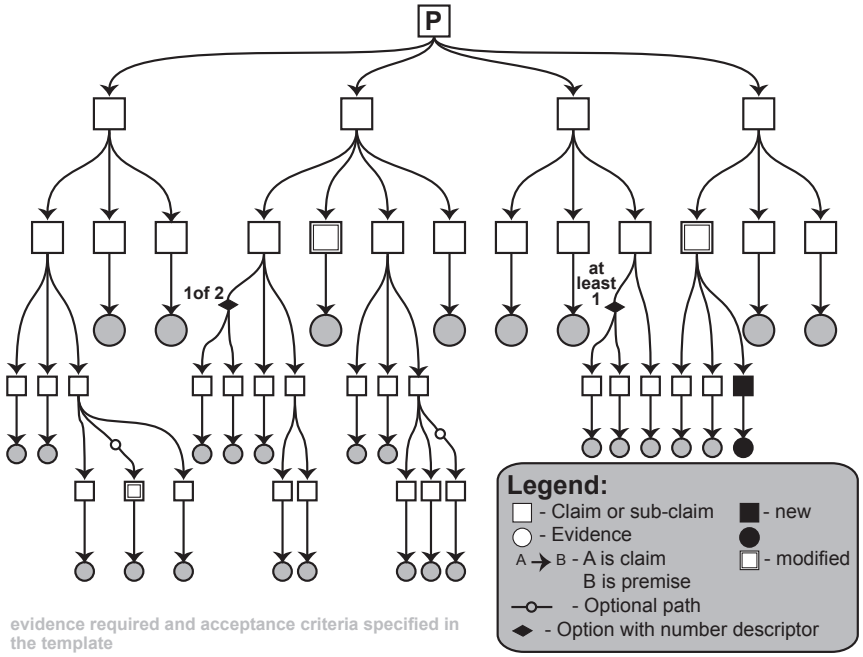

FIGURE 1.6 Describing Variation in the Assurance Case Template

of 2" descriptor, while the non-exclusive-Or is shown with the solid diamond and an "at least 1" descriptor (our modification of the GSN "option").

1.4.2 Problems in Constructing Assurance Case Templates

There is work needed to really *demonstrate* that we can construct effective assurance case templates for CPS.

- *Avoiding a check box mentality:* It is crucial that the template cannot be treated in any way as something where we mindlessly "fill in the blanks", or simply note that we have provided what was specified in the template. Claims, sub-claims, assumptions, contexts, strategies, justifications and reasoning must be worded, as much as possible, in a way that makes the developer seriously consider each claim and how it relates to their specific situation. The evidence nodes in the template do not contain evidence as such until the developers document the relevant evidence. What is available in the node is the nature of the evidence required, and its acceptance criteria. The format used for the acceptance criteria should be such that it requires thoughtful interaction on the part of the developer in ascertaining whether or not the specific evidence generated

meets the acceptance criteria and related argument, and the developed should have to document why it is sufficient.

- *Coping with omissions in the template:* No matter how well constructed our template is, there will be omissions and "errors". Related to the point above, we need to develop a mechanism for recognizing "holes" in the argument structure for a specific situation that was not predicted.
- *Essential items that do not fit within an assurance case template:* There are likely to be items that are currently included in standards that do not have an obvious place in an assurance case. We do not know yet what these are, we have simply speculated that they exist. We need to find out what they are, and how we can handle them.
- *Open systems:* A much more serious problem to deal with is that of constructing an assurance case template for an *"open system"*, i.e. a system that is constructed by combining existing systems (perhaps with other systems/components specially constructed for the new system. Such Systems of Systems pose unique and extremely difficult challenges for dependability, safety and security. The current concept of an assurance case template will not cope adequately with such systems. This does not mean that we will not be able to extend the concept in the future. It is simply an acknowledgement that right now we do not know how to achieve this. This is not surprising. We believe that assuring dependability, safety and security of such open systems is not adequately dealt with in general at this time.

1.4.3 Benefits of Using an Assurance Case Template as a Standard

It is one thing to say we could replace some existing standards by assurance case templates it is another to explain why we should do so. Without significant benefit, it would be pointless to attempt this. This section presents the major perceived benefits.

- *Explicit, consistent and understandable guidance:* An assurance case template provides guidance to both the developer and certifier in a way that is much more structured than is a textual standard. Yes, the template is likely to be complex, and large. However, so are modern standards. The structure does have some inherent benefits that help us contain and control the complexity.
  - It is easier to make the standard internally consistent.
  - Given both the strategies used to decompose claims, the arguments that explain why the premises support the claims, what evidence is required and the acceptance criteria for the evidence, users of the template understand how items depend on each other, what

evidence to document, the rationale for evidence, and what should be true of the evidence to satisfy the claims.
  - There is a consistency to the way in which these items are presented to the user. This aids understandability, and it also aids completeness since it is more difficult to ignore/forget items.
  - The need to confirm and/or enter assumptions and contexts also helps to develop a deep understanding of the system. The fact that these are supposed to hold consistently across the product-domain makes it easier for developers and certifiers to understand whether anything is "out of the ordinary" for the system at hand.

- *An opportunity for incremental safety:* In Section 1.3.7 we discussed incremental safety, and pointed out that it may not be possible – and that we could not afford for that to be true. We may have an opportunity with an assurance case template to make some inroads in this regard. We have built the template so that the fact that the argument is explicit (to reinforce the strategy), that the description of the evidence to support a claim is explicit, and that the acceptance criteria for the evidence is explicit, makes it possible to create an assurance case that will eventually apply to a variety of systems. If this is achievable, then we will have created a methodology for incremental assurance – if the modified system still fits within the assumptions and contexts that govern the assurance case template.

- *Easier to develop expertise in evaluating safety, security and reliability:* One of the first things that occurred to us when we first started evaluating assurance cases for regulated systems, was that if every assurance case has a one-off structure, then it will be extremely difficult for regulators to develop expertise in evaluating assurance cases submitted to them [40]. A product-domain specific assurance case template will help to alleviate this problem.

- *Valid arguments built by experts:* The argument in the assurance case template is the state-of-the-art component that will help us tame complexity in our modern CPS, and thus make it possible to develop and certify adequately safe, secure and reliable systems. For this to happen, the argument has to be both valid and sound. Unfortunately, the argument is one of the more difficult aspects in the creation of assurance cases, and the number of people with the expertise to do this is rather limited. The fact that the assurance case template we envisage will be created by a community of experts is of crucial importance. Amongst those experts, we can make sure that some of them have the requisite expertise in argumentation. Hopefully, gradually, the number of people with this expertise will grow.

- *Avoid confirmation bias:* The very fact that the assurance case template is developed prior to development of a system, should definitely

help in avoiding confirmation bias – a potential problem postulated by Leveson [27]. In this regard, there is a tremendous difference between developing the assurance case, including acceptance criteria on evidence, before developing the system, and developing the assurance case after or during development of the system – and then justifying the acceptability of the evidence.

- *Publicly available examples of good assurance cases:* assurance case templates, developed by a community of experts will provide something we do not have at all at this time – examples readily available so that more and more practitioners can develop a thorough understanding of, and facility with, assurance cases.

1.5 CONCLUSION

We propose that we use a product/system-domain specific assurance case template as a standard for the development and certification of products/systems within that product domain. Assurance cases have been growing in acceptability as an excellent way of determining the confidence we have that a system fulfills the need for which it was intended, and possess the critical properties it is required to posses. The assurance case template we envisage is the structure of a complete assurance case with (optional) sub-claims and requirements on generated evidence. The template is to be produced, much like a standard, through consensus of experts in the relevant domain. Such an approach promises to deliver significant benefits, as itemized in sections above. Although it seems that our discussion in this chapter is largely independent of CPS, one of the guiding motivations of this work is to be able to cope with the complex, multi-disciplinary, connected nature of today's and tomorrow's CPS. Developing and certifying such systems, so that they are dependably safe, secure and reliable, seems to call out for a significant change in the way we guide people through the development and certification of these systems. We believe that a concerted effort in understanding how to produce effective assurance case templates for this purpose will be one effective step in meeting the challenges of the (near) future. Luckily, they should be just as useful for less complex systems as well.

Bibliography

[1] Shuib Basri and Rory V. O'Connor. Understanding the perception of very small software companies towards the adoption of process standards. In Andreas Riel, Rory O'Connor, Serge Tichkiewitch, and Richard Messnarz, editors, *Systems, Software and Services Process Improvement*, volume 99 of *Communications in Computer and Information Science*, pages 153–164. Springer Berlin Heidelberg, 2010.

[2] Peter Bishop and Robin Bloomfield. A methodology for safety case development. In *Industrial Perspectives of Safety-Critical Systems*, pages 194–203. Springer, 1998.

[3] Robin Bloomfield and Peter Bishop. Safety and assurance cases: Past, present and possible future - an Adelard perspective. In Chris Dale and Tom Anderson, editors, *Making Systems Safer, Proceedings of the Eighteenth Safety-Critical Systems Symposium, Bristol, UK*, London, 2010. Springer London.

[4] Robin E Bloomfield, Bev Littlewood, and David Wright. Confidence: its role in dependability cases for risk assessment. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, pages 338–346. IEEE, 2007.

[5] V Cassano and TSE Maibaum. The definition and assessment of a safety argument. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 180–185. IEEE, 2014.

[6] Ewen Denney and Ganesh Pai. A formal basis for safety case patterns. In *Computer Safety, Reliability, and Security*, pages 21–32. Springer, 2013.

[7] Ewen Denney, Ganesh Pai, and Ibrahim Habli. Towards measurement of confidence in safety cases. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 380–383. IEEE, 2011.

[8] George Despotou, Sean White, Tim Kelly, and Mark Ryan. Introducing safety cases for health IT. In *Proceedings of the 4th International Workshop on Software Engineering in Health Care*, SEHC '12, pages 44–50, Piscataway, NJ, USA, 2012. IEEE Press.

[9] John Goodenough, Charles B Weinstock, and Ari Z Klein. Toward a theory of assurance case confidence. 2012.

[10] John B Goodenough, Charles B Weinstock, and Ari Z Klein. Eliminative induction: a basis for arguing system confidence. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 1161–1164. IEEE Press, 2013.

[11] P.J. Graydon, J.C. Knight, and E.A. Strunk. Assurance based development of critical systems. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 347–357, June 2007.

[12] P.J. Graydon, J.C. Knight, and E.A. Strunk. Assurance based development of critical systems. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 347–357, June 2007.

[13] S Grigorova and TSE Maibaum. Argument evaluation in the context of assurance case confidence modeling. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 485–490. IEEE, 2014.

[14] Silviya Grigorova and TSE Maibaum. Taking a page from the law books: Considering evidence weight in evaluating assurance case confidence. In *Software Reliability Engineering Workshops (ISSREW), 2013 IEEE International Symposium on*, pages 387–390. IEEE, 2013.

[15] GSN Community. GSN community standard, 2011.

[16] Ibrahim Habli and Tim Kelly. Process and product certification arguments: Getting the balance right. *ACM SIGBED Review*, 3(4):1–8, 2006.

[17] John Hatcliff, Alan Wassyng, Tim Kelly, Cyrille Comar, and Paul Jones. Certifiably safe software-dependent systems: challenges and directions. In *Proceedings of the on Future of Software Engineering*, pages 182–200. ACM, 2014.

[18] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. A new approach to creating clear safety arguments. In *Advances in systems safety*, pages 3–23. Springer, 2011.

[19] Ashlie B Hocking, Joseph Knight, M Anthony Aiello, and Shinichi Shiraishi. Arguing software compliance with iso 26262. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pages 226–231. IEEE, 2014.

[20] C Michael Holloway. Making the implicit explicit: Towards an assurance case for do-178c. In *Proceedings of the 31st International System Safety Conference (ISSC)*, 2013.

[21] Michaela Huhn and Axel Zechner. Arguing for software quality in an IEC 62304 compliant development process. In *Leveraging Applications of Formal Methods, Verification, and Validation - 4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Crete, Greece, October 18-21, 2010, Proceedings, Part II*, pages 296–311, 2010.

[22] Eunkyoung Jee, Insup Lee, and Oleg Sokolsky. Assurance cases in model-driven development of the pacemaker software. In *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 343–356. Springer, 2010.

[23] Paul Joannou and Alan Wassyng. Understanding integrity level concepts. *Computer*, (11):99–101, 2014.

[24] T. P. Kelly and J. A. McDermid. A systematic approach to safety case maintenance. *Reliability Engineering & System Safety*, 71(3):271–284, March 2001.

[25] Tim Kelly. *Arguing Safety – A Systematic Approach to Managing Safety Cases*. PhD thesis, University of York, September 1998.

[26] John Knight. Advances in software technology since 1992. In *National Software and Airborne Electronic Hardware Conference*, FAA 2008, 2008.

[27] Nancy Leveson. The use of safety cases in certification and regulation. *Journal of System Safety*, 47(6):1–5, 2011.

[28] T. Maibaum and A. Wassyng. A product-focused approach to software certification. *Computer*, 41(2):91–93, Feb 2008.

[29] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[30] David Lorge Parnas and Paul C Clements. A rational design process: How and why to fake it. *Software Engineering, IEEE Transactions on*, (2):251–257, 1986.

[31] David J Rinehart, John C Knight, Jonathan Rowanhill, and Dependable Computing. *Current Practices in Constructing and Evaluating Assurance Cases With Applications to Aviation*. 2015.

[32] RTCA. Software Considerations in Airborne Systems and Equipment Certification. *RTCA Standard DO-178B*, 1992.

[33] RTCA. Software Considerations in Airborne Systems and Equipment Certification. *RTCA Standard DO-178C*, 2012.

[34] John Rushby. Formalism in safety cases. In *Making Systems Safer*, pages 3–17. Springer, 2010.

[35] John Rushby. Understanding and evaluating assurance cases. (SRI-CSL-15-01), 2015.

[36] Stephen E Toulmin. *The uses of argument*. Cambridge University Press, 2003.

[37] U.S. Food and Drug Administration. Infusion Pumps Total Product Life Cycle: Guidance for Industry and FDA Staff, December 2014. OMB Control Number: 0910-0766.

[38] A. Wassyng, N. Singh, M. Geven, N. Proscia, H. Wang, M. Lawford, and T. Maibaum. Can product specific assurance case templates be used as medical device standards? *Design & Test, IEEE*, 32(5):1–11, October 2015.

[39] Alan Wassyng, Tom Maibaum, and Mark Lawford. On software certification: We need product-focused approaches. In *Foundations of Computer Software. Future Trends and Techniques for Development*, pages 250–274. Springer Berlin Heidelberg, 2010.

[40] Alan Wassyng, Tom Maibaum, Mark Lawford, and Hans Bherer. Software certification: Is there a case against safety cases? In Radu Calinescu and Ethan Jackson, editors, *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, volume 6662 of *Lecture Notes in Computer Science*, pages 206–227. Springer Berlin Heidelberg, 2011.

[41] Rob Weaver, Georgios Despotou, Tim Kelly, and John McDermid. Combining software evidence: arguments and assurance. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–7. ACM, 2005.

[42] Rob Weaver, Jane Fenn, and Tim Kelly. A pragmatic approach to reasoning about the assurance of safety arguments. In *Proceedings of the 8th Australian workshop on Safety critical systems and software-Volume 33*, pages 57–67. Australian Computer Society, Inc., 2003.

[43] Charles B Weinstock, John B Goodenough, and Ari Z Klein. Measuring assurance case confidence using baconian probabilities. In *Proceedings of the 1st International Workshop on Assurance Cases for Software-Intensive Systems*, pages 7–11. IEEE Press, 2013.

[44] Seiichi Yamamoto and Yutaka Matsuno. An evaluation of argument patterns to reduce pitfalls of applying assurance case. In *Assurance Cases for Software-Intensive Systems (ASSURE), 2013 1st International Workshop on*, pages 12–17. IEEE, 2013.