Can Product Specific Assurance Case Templates be Used as Medical Device Standards?

Alan Wassyng, Neeraj Kumar Singh, Mischa Geven, Nicholas Proscia,
Hao Wang, Mark Lawford and Tom Maibaum

Abstract—International standards are a key ingredient in the quality assurance of software intensive medical devices. One problem with such standards is that they often describe a life-cycle process that should be used to develop the system, rather than describe acceptance criteria to be applied to the system itself, thus guaranteeing safety directly in terms of the artefact's attributes. In the past few years, the U.S. Food and Drug Administration (FDA) introduced a (strong) recommendation that manufacturers submit an Assurance Case in their submission for approval to market an infusion pump. This reflects a move towards a more product/evidence based approach to certification, compared with the primarily process based certification used in the past. The perceived advantage of an Assurance Case is that it obliges the manufacturer to make an explicit argument regarding the safety/security/reliability of their product, under expected operating conditions. Taking this idea one step further, we explore whether there are benefits to using an Assurance Case Template as a new kind of standard, replacing existing process standards, and we describe some benefits of doing this.

1 Introduction

Recently, the U.S. Food and Drug Administration (FDA) introduced a (strong) recommendation that manufacturers submit an *Assurance Case*[1] that demonstrates the safety, security and reliability of any new infusion pump they want to market [2]. The motivation for this change in practice at the FDA, which previously required compliance with a process based standard, was the alarming rate of infusion pump failures [2], and the harm that this was inflicting on patients. These problems were apparent across a wide segment of manufacturers, and were occurring in spite of the fact that we have a number of international standards governing the manufacture of software intensive medical devices [3], as well as standards related to electrical and mechanical safety. A problem with many international standards of this type is that they make the unfounded assumption that following a life-cycle designed to produce safe, secure and reliable systems, will result in a system that achieves this. Having manufacturers use these "good" processes is, indeed, advantageous. It is simply not true that their use guarantees a "good" product [4]; this is true both in principle and in practice, as noted just above. We need to evaluate the quality of the product before declaring it *fit for purpose*. Of course, we can, and should, make our standards specify acceptance criteria on the product as well as on the process, but we do not have a good track record in this regard. The advantage of an assurance case is that its structure encourages (or should encourage) us to make our reasoning explicit as to why we believe that a system is safe, secure and reliable. The purpose of this paper is to propose that we use a product domain *assurance case template* (described later in this paper – see Section 4.1.3) as a new kind of standard for software intensive medical devices within that product domain. This de-emphasizes how the device is produced in favour of creating an explicit safety argument for it.

1. The FDA refers to it specifically as a *Safety Assurance Case*, emphasizing the safety aspect of the case, and has published an updated version of the Guidance document [1].

2 The Role of Standards

The construction of a standard is a community effort and reflects a state-of-the practice approach to the subject matter of the standard. The purpose of a standard that relates to the development of medical devices is typically to specify requirements on the process used to develop the medical device, or requirements on the medical device itself. A useful standard will include acceptance criteria on the process and/or product that will help to reduce variance in conformance to critical properties of the device, such as safety, security and reliability. In any industrial domain, we can typically find a number of international standards that fulfill this role. In addition, if the domain is regulated, as is the medical device domain, the government appointed regulator in the country in which the device is to be marketed 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.

3 Problems with Standards

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, process and services are fit for their purpose [3].

The use of standards has several potential benefits as described in Section 2. However, there are also problems and limitations with current standards that affect how useful they are in governing the development and certification of software intensive systems. Below, we summarize potential weaknesses in current standards.

Most software related standards are primarily process based: Other than requirements on test results, most current standards (from IEC, ISO and IEEE, for example) place requirements on the development process rather than on the product itself. As a result, compliance with such a standard is not a totally convincing reason for believing that the developed product is safe, secure and reliable; at best it is a statistical guarantee about the class of devices and not at all about a particular device.

Outdated standards: Process based standards encode best practice at the time they were written, and may eventually prevent developers from adopting best practice [5].

Complex and ambiguous: Many standards are incredibly complex, and are often hundreds of pages in length. Common terminology can have different meaning in different standards, and different terminology may be used for similar concepts [6]. There are usually good reasons for a specific clauses in the standard, but little or no rationale is included, and so the argument as to why non-compliance with any particular clause will lead to a lower quality product is typically also absent.

Checklist based completeness: Standards promote checklist based task completion that does not consider the technical quality of the process or the product – acceptance criteria are typically woefully inadequate or missing. Checklist completeness may be used to simplify the task of quality assurance, but without adequate acceptance criteria it cannot begin to succeed in evaluating the safety, security and reliability of the developed product.

Lack of design and technical detail: The majority of standards address software life-cycle processes, software development activities, and guidelines that aim to develop high-integrity software, without giving sufficient design and technical details, nor any particular software engineering methods to achieve the normative requirements [7].

Excessive documentation: Blind compliance with standards may produce excessive documentation. Lack of rationale and explicit acceptance criteria may lead manufacturers to produce documentation that is either not required or does not contribute to the development of a quality product. They do this to comply with perceived requirements in the standard, or because they are not sure what is required and over-compensate. This consumes resources in both the development and approval processes, and much of it may not contribute to system safety (for instance) at all.

Not adopted by small companies: The majority of small companies perceive existing process standards as being focused at large organizations. Small companies do not have the resources to implement some of the process requirements in standards (consider team independence, for example) [8].

4 A New Kind of Standard

The problems described in Section 3 are not insurmountable, and there are a number of approaches available to us that could be effective. However, it occurred to us that there is an alternative approach that may be more effective. This approach is based on *Assurance Cases*.

4.1 Assurance Cases

An Assurance Case is a modern generalization of the Safety Case [9], and safety cases have been in use for over 50 years, especially in the United Kingdom. An assurance case provides a structure in which the developer of a product makes a claim regarding critical properties of the product (e.g., safety, security, reliability), and then presents an argument that validates that claim. The argument typically uses a decomposition of that claim into sub-claims that are eventually 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 [9]. Figure 1 gives an idea of what an assurance case may look like, represented in GSN. It is taken from the GSN Community Standard website [10]. *Goals* in GSN represent claims; *Solutions* in GSN represent evidence; *Strategies* in GSN are used to explain why and how a claim was decomposed into specific sub-claims; *Justifications* in GSN are used to explain why a strategy was chosen and is appropriate; and so on.

4.1.1 Perceived Benefits of Assurance Cases

There are many benefits of assurance cases expressed by communities that have been using them. These benefits range from the explicit, detailed documentation of the 'case', to the traceability of evidence to a specific claim facilitated by the structure of the 'case'. A major benefit that is often expressed is that the *argument* related to the validity of the claim(s) is made explicit in the assurance case, and that this facilitates determining whether or not the top claim in the structure is valid – or, what *confidence* we have that the claim is valid. A well-structured assurance case can facilitate the identification of gaps between claims, arguments, and evidence in the constructed 'case'.

4.1.2 Problems with Assurance Cases

Every methodology has its problems. Assurance Cases are no different. There are a few important problems we want to highlight.

Argumentation: The most important problem with many current assurance cases is that the argument linking claims to sub-claims and eventually to evidence is simply *not* explicit. We say this because current assurance cases make explicit the *structure of claim decomposition* and the link of evidence to a specific sub-claim. While this structure is useful, this is not the same as making the argument explicit so that we can determine its validity. For example, consider Figure 1 again. *G1* (the top-level claim) is decomposed into two sub-claims, *G2* and *G3*. Usually there would be a *strategy* component that explains why this decomposition was selected. In this case there is no strategy provided. So, let us turn to the decomposition of *G3* into the sub-claims *G7* and *G8*. In this case there is a *strategy* provided, *S2*.

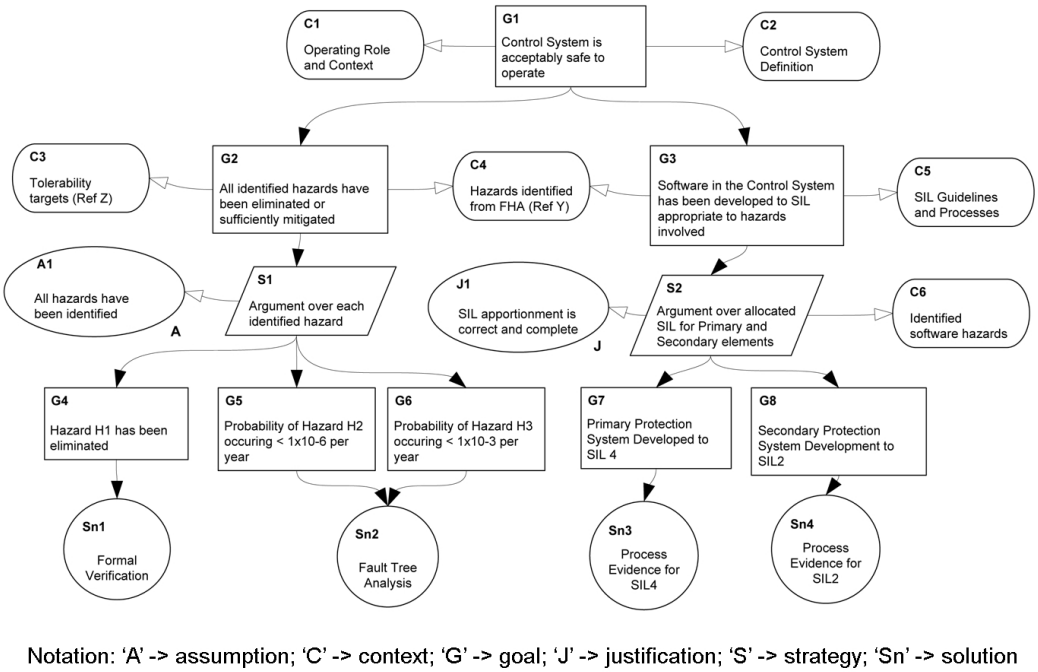

Fig. 1. GSN Community Standard Example [10]

Also, *S2* simply puts into words the same information we can glean from the arrows and the descriptions of *G7* and *G8*. Of course, this example is only being used to describe how the GSN components may be used to document an assurance case, and perhaps real assurance cases are not like this at all. Unfortunately, many are. However, we cannot cite publications to support our view. We have seen real, industrial assurance cases that lack this explicit reasoning – but, like many such cases, they are proprietary and we cannot cite them. More importantly, even if the *strategy* were better worded, at best it would describe a rationale for decomposing *G3* into *G7* and *G8* – it would not provide an *argument* as to why, if *G7* and *G8* are both proven to be valid, it would follow that *G3* must be true. This is what is missing. We should point out that this is not an inherent flaw in assurance cases, it is simply reflective of current practice.

One-off structures: Another problem with assurance cases is that if every assurance case is structured differently, it will be extremely difficult for both developers and regulators to build experience in detecting flaws in assurance case structure/arguments in their domain [11]. This is especially true for regulators of medical devices, such as the FDA who have to evaluate thousands of submissions every year.

Top-level decomposition: Safety Cases have often been structured by showing how all (known) hazards have been eliminated or mitigated. We are convinced that this is not the best strategy for structuring assurance cases, even when we are assuring safety. Our opinion is that we have to show that the system (medical device in our case) "produce[s] the consequences for which it was designed" [12]. This is important because if the system does not deliver on the behaviour its users expect, those users often find work-arounds, which can seriously impact the safety we thought we had assured. For instance, we have heard discussion on the use of radiation machines, where, in the case of obese patients, a protective cone has been removed because otherwise the patient could not be treated at all. Such behaviour is often considered to be *unlikely* during the development of the hazard analysis, and may not have been mitigated adequately. In addition, the system has to be as safe and secure as we can make it, in the face of unknown hazards as well.

Safety should be designed-in: Almost all researchers in the safety community know that safety is designed into a system right from the start. Proponents of safety/assurance cases have made it clear that an assurance case is not something that is documented after the system is built to be presented to a regulator [13]. However, for whatever reason, that message does not seem to be universally recognized. Too many assurance cases seem to be developed solely to convince a regulator that the system is safe and secure, and are thus documented after system development has been completed, or perhaps, while the system is being developed. We are not able to cite specific examples, but can simply say that other researchers have made the same observation [14], [15].

4.1.3 Suggested Solutions

Argumentation: We need to provide both the strategy (a top-down explanation of why the claim decomposition was chosen), and the argument (bottom-up rigorous reasoning that demonstrates that the combined effect of the decomposed claims implies that the parent claim is valid). Strategies in current assurance cases seem quite superficial and often say little more than we can see from the graph structure itself. The strategy should provide the insight that led to

the decomposition of a (sub-)claim, and this should include decisions made to support the bottom-up argument we believe is essential. This bottom-up argument will require expertise in argumentation that is currently not wide-spread – in fact, other than researchers who are still divided as to how to structure these arguments (this level of knowledge would not be common in any engineering community), many users of GSN seem unaware that there is a problem. The combination of strategy and argument helps us build much more robust 'cases'. Interestingly, the original plans for GSN and earlier work on GSN such as [16] are very much in the spirit of what we have said here. There was explicit realization that the bottom-up argument is also required. As mentioned earlier, this is not a deficiency in assurance cases – but it is a pervasive problem in many current assurance cases. A standardized template, as described below, could help in this regard.

Standardized template: A number of solutions to limiting the variety of assurance cases have been proposed. Two of these are closely related – *safety case patterns* [9], and *assurance case templates* [11]. Safety case patterns were originally described as "A means of documenting and reusing successful safety argument structures" [9]. The idea here is that an assurance case could be composed primarily of well-known (decomposition) patterns. This is reminiscent of *design patterns* [17], that are widely used in software design. An extension of this idea is an assurance case template, which is an almost complete assurance case structure that can be determined before development starts, in which missing details are provided during development, and some elements may be modified during development. We believe this works well within a specific product domain. For instance, we may use one assurance case template for infusion pumps, and a different one for radiation treatment machines. The assurance case template approach includes a complete, documented argument, examples of evidence and associated acceptance criteria that must accompany the 'case'. The argument and evidence would continue to drive development even when the template has a placeholder for details relevant to the specific device. This would enable modifications to the 'case', since the effect of the modification would be known and the argument then could be reconstructed. An assurance case template for insulin pumps was described briefly in [18]. Without templates or patterns, developers will still reuse components from previously constructed assurance cases. However, this reuse has the same problems associated with it as those described in [19], that is:

- *artefacts being reused inappropriately;*
- *reuse occurring in an ad-hoc fashion;*
- *loss of knowledge;*
- *lack of consistency/process maturity;*
- *lack of traceability;*

and to this we can add – the underlying argument may not be understood well enough.

Top-level decomposition: We decided early in our work that we should not structure the decomposition of the assurance case claims by mitigation of all identified hazards. This, then raises the question as to how we do structure it. The answer seemed straightforward. We should do what we have been doing for years in terms of how we plan our current projects. This leads to the top-level structure shown in Figure 2. This figure cannot show enough detail because of length constraints on the paper. S1 and R1 would be more detailed, and relevant assumptions, justifications and context would help in understanding the thoroughness of the approach, as well as lay the foundations for the appropriate arguments. The claims and evidence related to mitigation of hazards are now embedded lower down in the structure. For example: the claim that system level hazards have been identified and mitigated is in a sub-claim path below *G2*, and the claim that there are no unmitigated hazards (at all) forms a sub-claim path below *G3*. If we are intent on implementing an iterative hazard analysis, this seems an appropriate and effective way to do it.

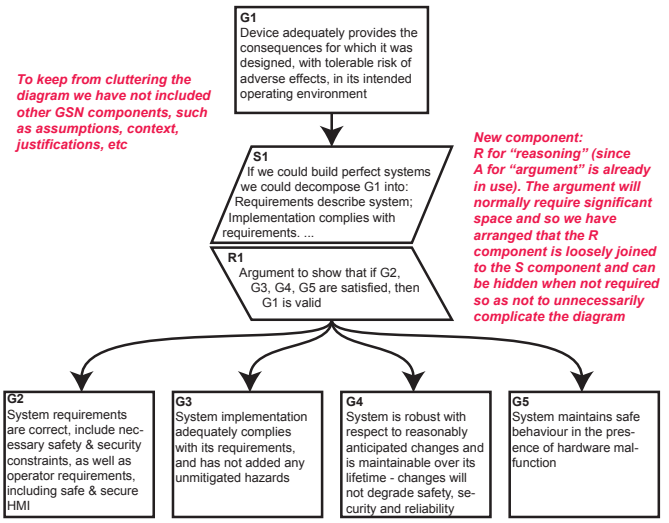

Fig. 2. Top-level Claim Decomposition

Planning a safe and secure system that is fit for purpose: Anyone who has developed a safety-critical software intensive system knows that the system is designed to be safe and secure when development is planned (as well as deliver functionality that will serve the needs of its users). It is not built on the hope that it will be safe. We use our knowledge and experience of what did or did not work for previous, similar systems. We plan a development process, and put in place checks and balances to monitor the quality of our product and process as we proceed. Some of this knowledge is captured in the standards we have to comply with. There is also knowledge in the corporate memory, and in process documents developed by the company. An assurance case template is an excellent way of documenting this knowledge. We discuss how to do this, as well as the benefits of using an assurance case template, in more detail in Section 4.2.

4.2 Assurance Case Templates as Standards

Once we had decided to use an assurance case template to drive development and both specify and monitor compliance with acceptance criteria on product and process, it occurred to us that we should *consider* replacing existing standards by such a template. We acknowledge that there is much work to be done to validate the feasibility and

efficacy of this approach. However, this paper is a first step in reporting on initial work we have done in exploring this option, and to exhort others to consider it, discuss it, and to build example templates.

In addition to assurance case researchers exhorting developers to build the assurance case early in the development cycle, our interest in using the assurance case, in the way we describe below, was inspired by the work of John Knight and colleagues on *Assurance Based Development (ABD)* [20]. After working on using assurance case templates and exploring their potential to replace standards, we were informed of a talk given by John Knight [21], in which he outlined the use of an assurance case as an alternative to the avionics standard DO-178B.

Most of our work is involved in how to build and evaluate the software in software intensive systems. However, we realize that there is more to these systems than software – and there are a variety of standards that apply to the manufacture of a medical device. The assurance cases we build are assurance cases for the safety, security and reliability of the complete system. It may be that there are aspects of existing standards that do not currently fit easily within the type of assurance case template we envisage. Our intent is to explore how they can be incorporated. In the meantime, the assurance case template may need to be supplemented by such standards.

With reference to Section 3, we briefly indicate how an assurance case template based standard would help alleviate some of the problems with standards that we identified.
Not primarily process based: It should be clear from the top-level decomposition (Figure 2) of the assurance case template we designed that the template is not primarily process based. Confidence in the argument will be dependent on the specific product evidence obtained and the degree to which the acceptance criteria are met. The template will not jettison all process related requirements. Even in product focused certification approaches, a process, sometimes just phases in an idealized process, is important in obtaining relevant evidence and overall confidence in the quality of the product [4].
Takes longer to become outdated: Assurance case template based standards may need updating less often than process based standards. Process based standards are updated when the state of the practice changes as regards how we believe we can achieve the quality goals relevant for that domain. The actual goals, evidence and acceptance criteria, if we actually identified them, may not have changed. An assurance case template is more focused on the evidence, argument and acceptance criteria, and as such, will be less likely to need to be changed as frequently.
Coping with complexity in the document: This new standard will be complex, in that it will be large and more detailed than many existing standards. It should be a lot easier to make it unambiguous in that it will be more internally consistent. The structure of the template makes this easier to achieve than in existing standards, that do not have the same explicit argument structure. Good assurance case tools that take advantage of the structure will make it easier to negotiate such a standard.
Less prone to checklist compliance: We have to be cognizant of the fact that a template always opens the door for 'mindless compliance' – similar to the checklist based completeness problem we observed in current standards compliance. However, the integral assumptions, evidence requirements and acceptance criteria accompanying the assurance case should mitigate against this. In addition, the template will not contain details for the specific product; these have to be obtained during the development of the product, and then documented in the template.
Appropriate technical guidance: With more explicit direction than in current standards, together with (again) evidence requirements and acceptance criteria, developers will know what is expected of them, and will not need to produce unnecessary documentation in the hope that it strengthens their regulatory case.
May be easier for smaller companies to adopt: Explicit direction will also make it possible for small companies to comply with standards more easily than they can now. However, the technical requirements may be more stringent and require better trained people.

4.2.1 Community Development

To use an assurance case template as a standard, we have to develop the template in much the same way we do conventional standards. We need contributions from all stakeholders – industry, academia, and regulatory bodies. Each assurance case template has to reflect the expectations of the product domain, and in particular, must result in compliance with the requirements imposed by the appropriate integrity level. It must also result in compliance with the principle of *As Low As Reasonably Practicable (ALARP)*, with regard to the residual risk in the product. There are real benefits in having the template developed as a community effort. Templates evolved through community development should benefit from input from a diverse group of experts. Members of industry can provide valuable input on practical development processes, members of academia can propose enhancements driven by their research to be tested in practice, and members of regulatory agencies can use the vast data collected from adverse events to further enhance them. Flaws found in a template can be corrected in a timely way to prevent multiple occurrences of similar mistakes. Every developer in the domain would use the latest approved template. The principal benefits are thus similar to those that apply to community development of standards, namely that communal expertise can be more than the expertise of a chosen few, and that industry is more likely to 'buy-in' to the effort required to comply as well as recognize the benefits of compliance. Specifically for assurance case templates, there is important and much needed consensus to obtain as to what evidence to produce and the acceptance criteria to be used.

4.2.2 Further Benefits

One of the ideas behind using the template as a standard is that the template will provide better guidance to developers than a traditional standard could. One of the major differences in using an assurance case template as a standard is that, currently, standards actually represent minimum requirements on the development process (and some aspects of the product). We envisage the template as much more definitive, as well as explicitly including

evidence and acceptance criteria to satisfy compliance with ALARP.

A crucial assumption in all the research and development related to assurance cases is that they provide a structure that, with the information contained within the structure, presents a compelling demonstration that the system of interest possesses the properties and attributes claimed in the assurance case. This is certainly true when the assurance case is well-structured (the argument at the base of the assurance case is valid) with relevant evidence tied to specific sub-claims.

An assurance case template describes lifecycle artefacts that provide the evidence needed to support sub-claims of the system. It also describes acceptance criteria on that evidence. Once the developer fills in the details relevant to the specific system they are building, examination of this evidence helps the regulator build confidence in the validity of the assurance case that is presented. The use of a community developed assurance case template will help in three important ways:

- the argument on which the 'case' is based should be solid since people with the requisite skill and knowledge to build and confirm the argument will be members of the team;
- most developers will use the template without much modification, filling-in details in appropriate places. This will allow regulators to build expertise in evaluating the cases submitted to them.
- Regulators, like the FDA, are loathe to prescribe in any detail *how* manufacturers should manufacture the device. An assurance case template fits this kind of regime splendidly since the template typically details much more *what* – evidence and acceptance criteria – than it does *how* to attain the evidence or *how* to achieve the acceptance criteria.

In earlier sections (see for example Section 4.1.2) we noted that most industrial assurance cases today are proprietary. This is a great pity since manufacturers cannot see examples of good (and bad) assurance cases that would help them build their own expertise. One additional benefit of a community assurance case template is that it would not be proprietary. It would be available to all (although judging by current practice it may be expensive), and would serve not only as a standard in the traditional sense, but would also serve as an example of a *good* assurance case.

4.2.3 Evidence of Benefits

FDA Approved infusion pumps have resulted in patient deaths due to buffer overflows in their software [22] despite the fact that such errors are easily detectable using commercially available static analysis tools. In its latest guidance, the FDA now recommends that, to help demonstrate software safety, a manufacturer should provide a static analysis of all software in the infusion pump system [1]. This effectively becomes evidence for, amongst other things, demonstrating that the manufacturer has a high degree of confidence that the system will be free from runtime errors such as buffer overflows. By making explicit the recommended claim (the software is free from common runtime errors), argument (because static analysis found no unresolved anomalies), and evidence (results of the static analysis), the FDA is providing clarity on known hazards and their mitigations, while not preventing a manufacturer from innovating in how they deal with the issue by, for example, using a strongly typed language and certified compiler. In fact, the assurance case presents the capability for the manufacturer to build *confidence* in their *defence-in-depth* approach by showing that the use of a static analysis tool was augmented by the use of the strongly typed language and certified compiler.

5 Conclusion

We have proposed that we use a product domain assurance case template as a standard for the development and licensing of medical devices within that product domain. Assurance cases are growing in acceptability as an excellent methodology for determining the confidence we have that a medical device is safe, secure and effective. Taking this one step further, if we develop an assurance case template, an almost complete assurance case with placeholders for yet to be determined sub-claims and details of generated evidence, then even at this early stage, there seems to be sufficient reason to believe that eventually we will be able to use such a template instead of current process based standards to guide both development and regulatory evaluation of a medical device.

Acknowledgments

The authors recognize the support of the Ontario Research Fund – Research Excellence, IBM, and the Southern Ontario Smart Computing Innovation Platform. We thank Paul Joannou for his many difficult to deal with questions, as well as John Knight, Tim Kelly, John McDermid and John Rushby for all their work as well as discussion that has guided our own research. We also wish to thank the anonymous reviewers who challenged us to substantiate claims and improve the paper in numerous ways. They succeeded in making us think more deeply about certain aspects of the work, and contributed significantly to improving our original version.

References

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

[2] ——, *Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions*, Food and Drug Administration Std., April 2010.

[3] IEC 62304, *Medical Device Software – Software Life Cycle Processes*, International Electrotechnical Commission Std., May 2006.

[4] T. Maibaum and A. Wassyng, "A product-focused approach to software certification," *Computer*, vol. 41, no. 2, pp. 91–93, Feb 2008.

[5] R. Bloomfield and P. Bishop, "Safety and assurance cases: Past, present and possible future - an Adelard perspective," in *Making Systems Safer, Proceedings of the Eighteenth Safety-Critical Systems Symposium, Bristol, UK*, C. Dale and T. Anderson, Eds. London: Springer London, 2010. [Online]. Available: http://www.springerlink.com/index/10.1007/978-1-84996-086-1

[6] P. Joannou and A. Wassyng, "Understanding integrity level concepts," *Computer*, no. 11, pp. 99–101, 2014.

[7] M. Huhn and A. 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*, 2010, pp. 296–311. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16561-0_30

[8] S. Basri and R. V. O'Connor, "Understanding the perception of very small software companies towards the adoption of process standards," in *Systems, Software and Services Process Improvement*, ser. Communications in Computer and Information Science, A. Riel, R. O'Connor, S. Tichkiewitch, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2010, vol. 99, pp. 153–164. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15666-3_14

[9] T. Kelly, "Arguing safety – a systematic approach to managing safety cases," Ph.D. dissertation, University of York, September 1998.

[10] GSN Community, *GSN Community Standard*, Std., Rev. Ver. 1, 2011. [Online]. Available: http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf

[11] A. Wassyng, T. Maibaum, M. Lawford, and H. Bherer, "Software certification: Is there a case against safety cases?" in *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, ser. Lecture Notes in Computer Science, R. Calinescu and E. Jackson, Eds. Springer Berlin Heidelberg, 2011, vol. 6662, pp. 206–227. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21292-5_12

[12] N. R. Council, D. Jackson, and M. Thomas, *Software for Dependable Systems: Sufficient Evidence?* Washington, DC, USA: National Academy Press, 2007.

[13] T. P. Kelly and J. A. McDermid, "A systematic approach to safety case maintenance," *Reliability Engineering & System Safety*, vol. 71, no. 3, pp. 271–284, Mar. 2001. [Online]. Available: http://dx.doi.org/10.1016/s0951-8320(00)00079-x

[14] T. Kelly, "Are safety cases working," *Safety Critical Systems Club Newsletter*, vol. 17, no. 2, pp. 31–33, 2008.

[15] C.-L. Lin and W. Shen, "Generation of assurance cases for medical devices," in *Computer and Information Science*, ser. Studies in Computational Intelligence, R. Lee, Ed. Springer International Publishing, 2015, vol. 566, pp. 127–140. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10509-3_10

[16] R. Weaver, J. Fenn, and T. 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*, ser. SCS '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 57–67. [Online]. Available: http://dl.acm.org/citation.cfm?id=1082051.1082056

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[18] Y. Chen, M. Lawford, H. Wang, and A. Wassyng, "Insulin Pump Software Certification," in *FHIES 2013: 3rd International Symposium on Foundations of Health Information Engineering and Systems*, ser. LNCS, vol. 8315. Springer, 2013, pp. 87–106.

[19] T. P. Kelly and J. A. McDermid, "Safety case construction and reuse using patterns," in *SafeComp 97*. Springer, 1997, pp. 55–69.

[20] E. A. Strunk and J. C. Knight, "The essential synthesis of problem frames and assurance cases," *Expert Systems*, vol. 25, no. 1, pp. 9–27, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1468-0394.2008.00452.x

[21] J. Knight, "Advances in software technology since 1992," in *National Software and Airborne Electronic Hardware Conference*, ser. FAA 2008, 2008. [Online]. Available: http://www.cs.virginia.edu/%7Ejck/publications/FAA.SW.AEH.2008.PDF

[22] U. Food and D. Administration, "Baxter healthcare Pte. Ltd. colleague 3 cxe volumetric infusion pump 80frn," MAUDE Adverse Event Report Catalog Number 2M9163, 2007, http://www.accessdata.fda.gov/SCRIPTs/cdrh/cfdocs/cfMAUDE/Detail.cfm?MDRFOI__ID=914443.

Alan Wassyng is Director of the McMaster Centre for Software Certification, and one of its founders. He is an Associate Professor in the Department of Computing and Software at McMaster University, Canada, and has been involved in development and certification of safety-critical software intensive systems for more than 25 years.

Neeraj Kumar Singh received his Ph.D in Computer Science from Henri Poincaré University, Nancy 1 (now the University of Lorraine), France, in 2011. He is a postdoctoral researcher in McMaster University's Centre for Software Certification. His research interests include formal methods, software engineering, software and system certification, and cyber-physical systems.

Mischa Geven is a Masters candidate in the Department of Computing and Software at McMaster University, currently developing an assurance case template for generic insulin infusion pumps as part of an M.A.Sc. thesis on the development and certification of such devices.

Nicholas Proscia is a Masters candidate in the Department of Computing and Software, McMaster University, currently completing an M.A.Sc. thesis on improving the software development process.

Hao Wang is an Associate Professor at Aalesund University College, Norway. He was recently a Research Scientist at IBM Canada R&D Centre, seconded to the McMaster Centre for Software Certification specifically to collaborate on research on the generic insulin infusion pump.

Mark Lawford is the Associate Director of the McMaster Centre for Software Certification, and one of its founders. He is a Professor in the Department of Computing and Software, and has more than 15 years experience integrating formal techniques with practical industrial software engineering applied to safety-critical real-time control systems.

Tom Maibaum received his Ph.D., in 1974 from University of London. He has held academic positions at University of Waterloo, Imperial College, and King's College, London. He holds a Canada Research Chair in Foundations of Software Engineering at McMaster University, and is a founder of its Centre for Software Certification.