Insulin Pump Software Certification

Yihai Chen[1,⋆], Mark Lawford[2,⋆⋆], Hao Wang[2,⋆⋆⋆], and Alan Wassyng[2,⋆⋆]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
[2] McMaster Centre for Software Certification
McMaster University, Hamilton, Ontario, Canada

Abstract. The insulin pump is a safety-critical embedded medical device used for treatment of type 1 and insulin treated type 2 diabetes. Malfunction of the insulin pump will endanger the user's life. All countries impose some regulation on the sale and use of medical devices. The purpose of such regulation is to protect the public by imposing standards of *safety* for medical devices, including insulin pumps. The regulator in the USA, the USA Food and Drug Administration (FDA), actually goes further, and includes *efficacy* in the regulatory requirement. Until recently, regulatory approval was dependent on *process based* guidance. However, this has proven to be inadequate in some (most) cases where the device depends on software for its safe and effective operation, and the FDA recently changed its approval process for infusion pumps (including insulin pumps), so that the production of an assurance case that demonstrates that the device is safe and effective is now a strongly suggested regulatory requirement. However the current regulatory guidance does not recommend any particular software development methodology, and does not include definitive guidance on the evaluation component of the certification process. In this paper, we briefly review the related USA regulatory standards for insulin pumps, highlight development and certification challenges, briefly discuss attributes of a safe, secure and dependable insulin pump, and propose an effective certification process for insulin pumps.

Keywords: insulin pump, safety critical system, software certification, standards compliance.

1 Introduction

Diabetes mellitus is one of the major noncommunicable diseases (NCDs) facing modern society today. Type 1 diabetes results from the inability of the pancreas to create the insulin required to constrain the blood glucose levels in the body. Type 1 diabetes is fatal unless insulin can be introduced into the bloodstream. Type 2 diabetes is also characterized by high blood glucose levels, but in this case the lack of insulin in the body is not absolute. Type 2 diabetes can be treated through other means, but sometimes

⋆ Supported by National Natural Science Foundation of China (NSFC) under grant No. 61170044 and China Scholarship Council.
⋆⋆ Partially supported by IBM SOSCIP Project, and Ontario Research Fund - Research Excellence.
⋆⋆⋆ Supported by IBM Canada R&D Centre and Southern Ontario Smart Computing Innovation Platform (SOSCIP) project.

J. Gibbons and W. MacCaull (Eds.): FHIES 2013, LNCS 8315, pp. 87–106, 2014.
© Springer-Verlag Berlin Heidelberg 2014

the use of insulin is necessary. Diabetes is directly responsible for 3.5% of NCD deaths. Type 1 and insulin treated type 2 diabetes patients must inject insulin daily for their survival. Historically, this has been achieved by the patient injecting insulin at particular times in the day, typically at meal time.

For some years now, an alternative has been available. Continuous subcutaneous insulin infusion (CSII) has been successfully used to treat type 1 and insulin treated type 2 diabetes patients. This subcutaneous infusion of insulin is achieved through the use of an *insulin pump*.

An insulin pump is a pager sized electronic device that continuously delivers insulin using a catheter. The first reported insulin pump system was developed by Dr. Arnold Kadish in the early 1960s. Since then people have explored using CSII therapy to treat diabetes. The Diabetes Control and Complication Trial [28] in 1993, showed that sustained lowering of blood glucose slows diabetes complications. The insulin pump can imitate physiological insulin secretion, and a recent study showed that it results in significant improvement in glycated hemoglobin levels as compared with injection therapy [4]. These research findings and improvements in the design and function of insulin pumps motivated their increased usage in recent years. More than 300,000 patients around the world use insulin pumps today [26].

An insulin infusion pump is a safety-critical software-intensive medical device (SMD). Flaws in the pump software can cause serious injury or even loss of life. Until recently, regulatory approval of SMDs was dependent on *process based* guidance, e.g., in Europe, IEC 62304 regulates the development processes of medical device software. Process based guidance has proven to be inadequate: The USA Food and Drug Administration (FDA)[1] received nearly 17,000 insulin pump-related adverse-event reports from Oct. 1, 2006 to Sept. 30, 2009 [8] and 41 of the 310 death reports were associated with blood-sugar levels being too high or too low, suggesting the device may not have been working properly.

The FDA recently changed its approval process for infusion pumps, so that the production of an *assurance case* that demonstrates that the device is safe and effective is now a *recommended* regulatory requirement in the USA. However the current regulatory guidance does not recommend any particular software development methodology, and does not include definitive guidance on the evaluation component of the certification process. In this paper, we briefly review the related USA regulatory standards for insulin pumps, discuss development and certification challenges for such medical devices, and propose an effective certification process for insulin pumps.

The remainder of this paper is organized as follows: section 2 introduces the domain knowledge of an insulin pump system and the challenges for development and certification. Section 3 gives an overview of related regulatory requirements and standards for insulin pumps. Section 4 discusses required quality attributes, and Section 5 presents a suggested certification process based on our understanding of those software attributes and assurance cases. The final section provides conclusions.

[1] In this paper, we use the FDA as representative of a government regulatory agency for medical devices because we are more familiar with their regulatory standards and guidelines and their certifying practices, and, more importantly, FDA approval is an important benchmark for marketing medical devices globally.

2 The Insulin Pump System

2.1 A Generic Insulin Pump

Typically, in modelling such devices, and in discussion related to their development and certification, we need to be specific regarding the features and components of the devices. This paper discusses a *generic* insulin pump, which is a pump that is not marketed and not manufactured. The reason we have done this is so that we are able to describe a pump that is typical of pumps on the market, and exhibits behaviour representative of most pumps that we know about. The idea is to remove this from the baggage that is often associated with analysis of an existing, physical pump, manufactured by a specific company. The paper is concerned with principles, rather than with the certification of a specific pump.

Throughout the remainder of this paper, when we talk about the insulin pump, we are talking about the *generic insulin pump*. The structure of such a pump is presented in the following section.

2.2 Insulin Pump Structure

The basic components of an insulin pump include (see Figure 1):

- a user interface;
- the controller;
- the pumping mechanism;
- the insulin reservoir; and
- wireless input/output.

Note that the infusion set, the complete tubing system to connect an insulin pump to the pump user, is assumed to be outside of the system boundary for the purpose of this paper. This affects any modelling of physical links and the hazards analysis, but has no other effect. We have excluded discussion related to it simply to reduce the complexity of the system dealt with in the paper. Similarly, although the battery is shown in Figure 1, we are not going to include battery behaviour in our analysis other than to recognize that power may be on or off. The last item missing from Figure 1, is the environment. For the sake of simplicity in the presentation of this paper, we assume that the environment, which is everything outside of the system, relates almost entirely to the user – whether it be infusion of insulin into the user's body, or data focused interaction with the pump. In actual practice, of course, we would include the hardware, all relevant aspects of the environment, and their interaction with the software, in the development of convincing proof that the insulin pump is safe and secure.

The remainder of this section will provide a brief description of the components of the insulin pump system.

The *user interface* includes the capability to input data, and also to choose actions. In addition, the pump is able to display information to the user. In our case, the input and output are kept very simple and not specific as to whether they are textual or audible.

The *controller* is the digital control unit that we are able to program. Again, this is assumed to be as general as feasible. We are not interested (in this paper) in the different

faults that are introduced by digital computers compared with field programmable gate arrays, e.g., The (pump) controller can be programmed to administer basal (periodic) or bolus (extra) insulin according to a patient's request.

The *pumping mechanism* is the hardware pump that moves the insulin from the reservoir into the user's body (through the infusion set). It is also generic in that we do not distinguish between the different types of pumps.

The *insulin reservoir* holds the insulin cartridges. The reservoir needs to have the capability of signalling when the remaining amount of insulin is low (below a threshold), or empty.

Some of the latest insulin pumps can communicate *wirelessly* with a remote computer, a continuous glucose monitor (CSM), or even the cloud [17].

The pump that we are considering is referred to as an *"open loop" system* in that it does not use any form of automated feedback to determine the amount of insulin to administer. Compared with open loop insulin pumps, a *closed-loop system* or "artificial pancreas" can monitor glucose levels 24/7, and automatically delivers an appropriate dose of insulin without a patient's intervention. These pumps are just appearing, or are about to appear, on the market. They are much more complex than the open loop pumps, and since this complexity does not contribute to the principles and approaches we want to present, and because we have not yet managed to develop even the simpler open loop pumps to the appropriate level of dependability, we have excluded them from further discussion in this paper.

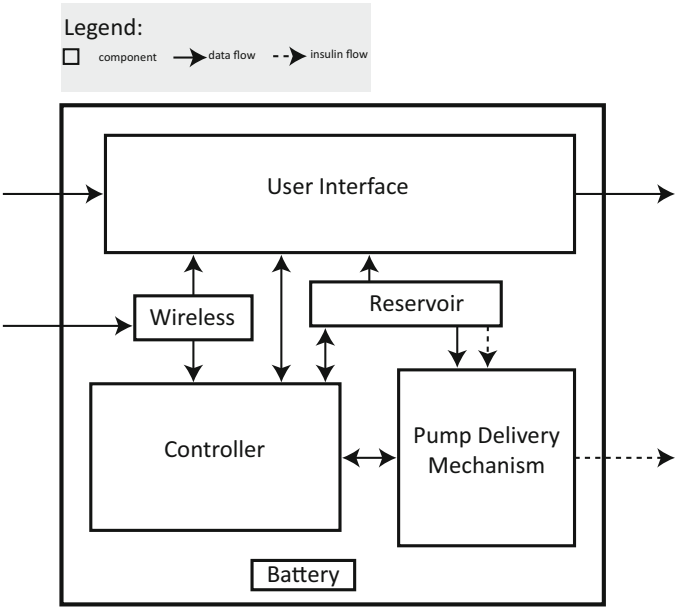

Fig. 1. Structure of a Generic Insulin Pump

2.3 Challenges for Development

Developers of medical devices face many challenges, and software intensive systems certainly add to these. The following challenges have to be dealt with if we are to be able to produce dependably safe and secure, software dependent insulin pumps.

- *Medical software is subject to government laws and regulations*
  In the USA, the FDA requires the manufacturers to submit a premarket notification[2] [510(k)] [14]. The submission process requires the insulin pump to be in compliance with guidelines including *General Principles of Software Validation* [12] and *Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices* [13], etc. More discussion on regulatory requirements can be found in Section 3.
- *Process based regulation – yet there is no recommended software development life cycle*
  In spite of the perceived burdensome regulation imposed on medical device manufacturers, in terms of the software embedded in the device, the regulation and referenced international standards do not mandate the usage of a specific software development life cycle.
- *Safety*
  This is a safety critical device and there are a large number of safety requirements. Below are examples of some safety concerns related directly to the components of the insulin pump.
  - The user interface is critical in that the user cannot be assumed to be as skilled as a nurse would be in interfacing with other types of infusion pumps
  - The pump delivery mechanism must not allow free-flow, and must not allow air bubbles to form in the liquid insulin
  - The controller must deliver 24/7 service outside of routine maintenance, and so must be proven to be deadlock free, and robust in the event of hardware failures
- *Security*
  Modern insulin pumps have wireless capabilities (see Figure 1) that bring additional security challenges for insulin pump software development. It has been reported that insulin pumps are vulnerable to hackers [2,7]. This security problem leads directly to a safety concern. Some research work has been done to address the problem, but it is far from being solved [19,27].

2.4 Challenges for Certification

The goal of certification is to systematically determine, based on the principles of science, engineering and measurement theory, whether an artifact satisfies accepted, well defined and measurable criteria [15]. Maibaum and Wassyng pointed out that process oriented software process quality does not necessarily translate into good software product quality [21]. The certification of insulin pump software poses some additional challenges:

[2] A stricter premarket approval (PMA) process is required for closed-loop insulin pumps.

- *Lack of clear definition of evidence and how to evaluate it [15]*
  What evidence would convince the FDA that the insulin pump is safe, secure and dependable? In general, the FDA provides very little definitive guidance on what evidence to provide. This results in huge unpredictability for the manufacturers, because they are not sure how to document or structure the evidence they have produced. Of course, in many cases they have not even developed the evidence that the FDA may be expecting. In addition to this, different staff at the FDA may have different lists of 'essentials'. In the case of the insulin pump, in particular, what will convince the regulator? Do they want/need to see that the manufacturer has defined the smallest air bubble that can be detected in the pump, and that bubbles over a threshold size can be prevented from being injected into the user?
- *Do not provide any guidance on how to evaluate assurance cases and hazard analysis results*
  Currently relevant standards and guidance recommend assurance cases and talk about mitigating hazards, but do not necessarily provide guidance on how to demonstrate that all hazards have been mitigated. So, the FDA recommends that assurance cases should be submitted – but it does not explicitly require that assurance cases should have a structure that argues over all identified hazards [14].
- *What do the FDA need to know about the battery?*
  The battery must be safe, i.e., it must not spontaneously combust. It needs to provide sufficient power and must provide *n* hours of continuous use. There should be accurate warnings to the user when battery life is low. What evidence should be presented to the FDA to support all of these 'claims'?
- *Integrating with third-party components*
  Insulin pump software usually integrates with third-party software components, called *Software Of Unknown Provenance* (SOUP) components in IEC 62304. For example, the insulin pump software may interface to WiFi radio software developed by another company. SOUP components impede certification efforts.
- *Insulin pump systems are becoming part of Medical Cyber-Physical Systems*
  A modern insulin pump system is not a stand-alone device anymore. It is connected with a continuous glucose management system, blood glucose monitor, and other associated devices and health information systems. These interconnections make the certification more difficult and more challenging.
- *Minimizing certification time and effort*
  Under the Medical Device User Fee Act (MDUFA), the FDA is under pressure to finish 510(k) reviews quicker than they do now. Failure to follow guidance document(s) or recognized standards, inadequate software documentation will delay the review process [10].

3 Overview of Regulatory Requirements for Insulin Pumps (Software Focus)

Many years ago, the FDA included a way of grandfathering devices onto the market with the so-called "510(k) Guideline". Its introduction was supposed to have been temporary, and was made at the time the U.S. started to regulate medical devices. It has

never been removed. Infusion pumps in general have been remarkably prone to error, and so the FDA has published new guidance on the 510(k) process, the *infusion pump premarket notification 510(k) guidance* (FDA 510K Guideline) [14]. The guideline requires that the manufacturer show *substantial equivalence* with an existing device on the market, and show that no new hazards are introduced in the device submitted for approval. The guideline also encourages the manufacturers to take advantage of any recognized software standards and provide statements or declarations of conformity as described in the FDA guidance *Use of Standards in Substantial Equivalence Determinations* [11]. In addition, in reaction to the poor dependability and safety record of infusion pumps, the new guideline recommends that manufacturers demonstrate substantial equivalence by using an *assurance case* to structure the claim [14]. In Section 5.2, we discuss assurance cases in detail and present a partial assurance case template as part of our proposed certification process.

Figure 2 shows international standards relevant to the development of medical device software. The IEC 62304 standard provides a framework of life cycle processes and requirements for each life cycle process. The standard requires that the manufacturer use a *quality management system*, for which ISO 13485 is recommended, and a *risk management process* complying with ISO 14971. In particular, IEC 62304 addresses the usage of SOUP.

Risk management is undoubtedly vital to the development and certification of insulin pumps, but the hazard analysis methods recommended by ISO 14971 do not reflect more recent established methods for software-intensive systems like the STAMP-based Analysis[3] (STPA) [20]. More importantly, the European Committee for Standardization (CEN) identified[4] all content deviations of ISO 14971 compared with the requirements of EU Directives 93/42/EEC. One deviation is that:

> ISO 14971 ... contains the concept of reducing risks "as low as reasonably practicable"...(while) Directive 93/42/EEC and various particular Essential Requirements require risks to be reduced "as far as possible" without there being room for *economic considerations*.

We can see that the majority of related standards are concerned with the development process of medical device software, with the exception of IEC 60601-1[5]. However, IEC 60601-1 is for general medical devices, so it does not cover all aspects of medical device software. The FDA 510K Guideline recommends the insulin pump meet IEC 60601-1 as to the alarms/warnings and environmental safety requirements

4 The Quality Attributes of Insulin Pump Software

This section discusses some of the quality attributes important to insulin pump software. It is not a comprehensive list, but covers several key elements that can be used as

[3] STAMP stands for *Systems-Theoretic Accident Model and Processes*, an accident model by the same MIT group

[4] when the ISO 14971:2007, Corrected version 2007-10-01 was taken over as a European Standard EN ISO 14971:2012

[5] The other product standard – IEC 61010-1 – is for electrical equipment for measurement, control and laboratory use, which is not relevant to insulin pumps.

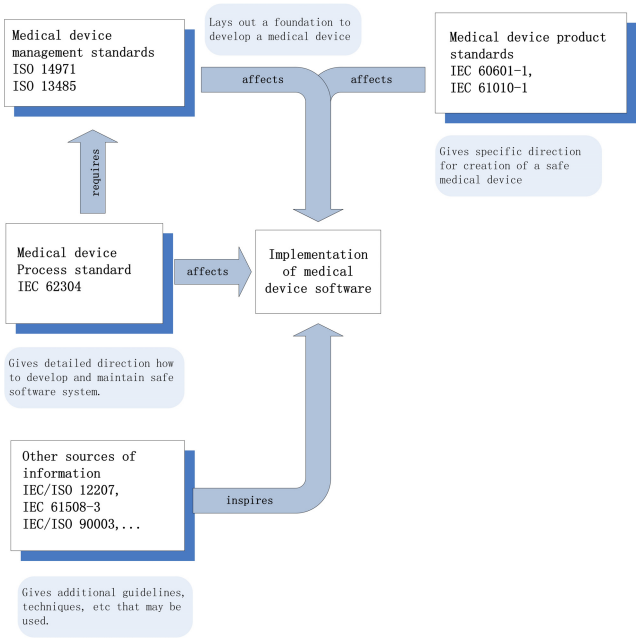

Fig. 2. Relationship of key MEDICAL DEVICE standards to IEC 62304 [18]

acceptance criteria to evaluate insulin pump software. We divide the quality attributes of the insulin pump into three sets: functional attributes, 'design' attributes ('design' in the conceptual sense, which includes requirements specifications as well), and software development process attributes. We have listed functional and design attributes in the remainder of this section. We have not included process attributes in order to save space, and also because process attributes form the mainstay of current certification processes and we do not have anything significantly new to say on the subject.

4.1 Functional Quality Attributes

Safety. The safety challenge was described briefly in Section 2.3. What evidence do we look for in this regard, and use in the certification? Brief list: global safety and liveness properties satisfied; hazards mitigated (includes fail-safe behaviour); completeness checks satisfied; Human machine interface (HMI) testing report. Figure 3 provides an excerpt from a table detailing the system hazards and proposed mitigations.

Security. The security challenge was also described briefly in Section 2.3. Brief list of evidence: hazards mitigated; completeness checks satisfied; freedom from coding defects.

Availability. This was also discussed in Section 2.3, and straddles the safety attribute. Evidence to consider includes: global safety and liveness properties satisfied; fail-safe hazards mitigated; completeness checks satisfied; freedom from coding defects.

Hazard	Contributing Factor	Mitigation
Failure in the pump causes the flow rate to be incorrect	Backflow through the pump	Use a pump which prevents flow when not being powered by the system
	Free flow through the pump	
	Efficiency fatigue over time	Require user to replace or perform maintenance after given period of time
	Cracking/damage to the pump	Use a pump that is composed of resilient materials
	Blockage	Pressure sensing included in the pump
	Not enough power provided	Regulate the power source and ensure the correct voltage outputs from system to pump
	Too much power provided	
Failure in reservoir provides incorrect volume of insulin	Cracking/damage	Use a reservoir that is composed of resilient materials
	Empty reservoir	Include volume sensing in the reservoir to alert when low
	Blockage	Pressure sensing included to ensure insulin is exiting the reservoir
Failure in the delivery line provides incorrect flow rate	Cracking/damage	Use a delivery line that is composed of resilient materials
	Line not attached to the user	Medical professional provides training on how to set up the infusion set
	Blockage	Pressure sensing included to ensure insulin flowing
	Air in line	Force user to prime the line before use and include sensing capabilities for air gaps

Fig. 3. Hazards and mitigations corresponding to infusion rate not matching requested value [29]

Usability. ([34]) Straddles safety and availability and is also discussed in Section 2.3. Evidence is primarily HMI related validation test reports.

Maintainability. The lifespan of insulin pumps may be years, requiring updates and safety/security enhancements as more usage data becomes available, and the science of insulin pumps further develops. Evidence to consider includes: requirements-traceability forward and backward; *information hiding* modularization (see 4.2); a documented uses hierarchy.

4.2 Design Attributes

Requirements Consistent, Complete and Unambiguous. If this is not true, then the software design loses its context. Evidence includes: completeness checks satisfied; verification reports that include consistency, completeness, etc.

No Dead Code. 'Dead code' affects both safety and security, and many regulatory domains have statements to the effect that it must be avoided. The question is what evidence do we need to show that it has been avoided? The most effective evidence in our opinion is a code (formal) verification report that demonstrates that the code faithfully implements the software design.

Code Free from Common Defects. Reports from two or more static code analysis tools are excellent evidence in this regard. We suggest two or more since different tools tend to find different errors [5].

Information Hiding. The objective of information hiding is to identify requirements or design decisions that are likely to change and to encapsulate the essence of what would change in a single module/class. This results in excellent modularity and clearly facilitates maintainability [23,24]. Evidence of effective information hiding includes lists of requirements and design decisions that are likely to change in the future, and traceability of those 'secrets' down into the software design level.

Design Facilitates Verification. Verification means confirmation through provision of objective evidence that specified requirements have been fulfilled [18]. The IEC 62304 [18] guided process requires planning for software verification. Borrowing from the nuclear domain, the design output "should facilitate the establishment of verification criteria and the performance of analyses, reviews, or tests to determine whether those criteria have been met"[22]. Evidence of this should be visible in project planning procedures, and in the design itself.

One task the certifier must then perform is to understand what quality attributes are used in claims in the assurance case, and evaluate the evidence associated with them. In the following section we describe the certification process.

5 A Suggested Certification Process

We have briefly described the current regulatory (certification) process for medical devices in the USA (Section 3). It is not remarkably different, we believe, in most countries that have equivalent oversight of medical devices. We also briefly described the major challenges faced by manufacturers of medical devices (Section 2.3), as well as certifiers of medical devices (Section 2.4). In this section we present an overview of a certification process for medical devices, focused on the evaluation of the software attributes in those devices.

It is important to note that we are considering the certification of a medical device that is safety critical – the insulin pump saves lives, but can also take them.

There is an old maxim that the certification of software intensive systems should depend on a tripod – *people, process and product*. Currently, the regulatory regime in most domains that deal with software intensive safety critical systems is predominantly process based. As we have seen, the certification of medical devices is no different. Our contention is that people (the developers and certifiers) and process are undoubtedly important, but the certification process should be as *product-focused* as possible. This section presents ways in which we think we can achieve this.

5.1 People and Process

Before we describe the product-focused aspects of the certification process, we deal with the essentials that relate to people and process.

People. The developers of the insulin pump must be competent in a variety of domains in order to develop a safe, secure, effective and dependable pump. In terms of software expertise, the developers need to be able to document knowledge and experience in the crucial aspects of safety critical software development. If the company/division is too small to be able to perform some of the activities, they will need to contract others to help them. Their expertise will also need to be documented. We are not certain that the FDA or other medical device regulators currently conduct checks of this nature – but they should. In particular, development teams must be able to document relevant expertise in: hazard analysis; software requirements elicitation and specification; software design; coding; hardware interfacing; security; testing; configuration management; and assurance cases. The acceptance criteria by which the regulator can evaluate this knowledge and expertise is currently problematic. The body of knowledge in these areas is not universally accepted. However, accreditation by various professional bodies, degrees in relevant disciplines, and so on, are all useful in this regard.

Development Process and Tools. Safe, secure and dependable software needs to be developed using a development process approved by the regulator. This is often achieved by showing compliance with a process standard, such as IEC 62304. Alternative standards usually exist within the relevant regulatory framework. The FDA, for example, has process guidance in their set of regulatory guidelines [11]. Any tool support for the process should not represent a potential "single point of failure" that either introduces an error or result in an error going undetected. In many standards such reliance upon a tool requires that the tool itself be developed to the same level of rigour as the system under development. Thus the process and supporting tools must support each other to eliminate any potential "single point failures".

5.2 Product

As indicated in Sections 1 and 3, the FDA now recommends the submission of an assurance case for insulin pumps. An assurance case should have substantial product evidence to support the claims and arguments included in the case, and so is one way in which we can focus our certification regime more on the product under scrutiny, rather than on the process used to build it.

A Brief Introduction to Assurance Cases. An assurance case (originally developed as safety cases) is a structured document that presents a claim about the product and also demonstrates the validity of the claim through a series of connected arguments, sub-claims and evidence [6].

Hawkins et al. [16] recently compared the two approaches to certification of software safety: *prescriptive certification*[6] and assurance cases. They argue that the two approaches are complementary and could lead to "a better solution than either approach on its own".

[6] Some standards prescribe specific processes and techniques that must be followed.

Assurance Case Template. We believe that the quality of the evaluation of the assurance case by a regulator (or certification authority) is just as important as the quality of the development of the assurance case by the manufacturer. This would seem to contradict our belief (shared by most proponents of assurance cases) that the act of developing the assurance case is more beneficial than the resulting assurance case, but it does not. For the manufacturer, the act of developing the assurance case forces the development team to consider gaps and the validity of their claims, continually. This assumes, of course, that the assurance case is developed with honest intent, and that it drives development, rather than serve as documentation after the fact. Sincerity and skilled work are not sufficient though. We are probably all familiar with the effect of reading and critiquing our own work ("own work" here includes other members of the team or members of another team within the same company). We need an objective check on this, and it makes sense that the relevant regulators (or certifying authority) are the ones to do it. Not only is it within their mandate, but they are (almost always) aware of issues that have plagued other manufacturers in the same domain. Thus, their evaluation of the assurance case is also vital.

Now that we have established the importance of the regulatory evaluation of the assurance case, we need to examine aspects of the development and evaluation of the assurance case that are likely to affect the quality of the evaluation. This has only recently generated interest. One of our concerns for some time now [32], has been that if each assurance case submitted to the FDA is a *one-off* example, then the FDA is not likely to build sufficient expertise in evaluating these assurance cases, and they are likely to struggle to find subtle flaws in these cases in the time that they have available for the evaluation. It is mainly for this reason, that we think that frameworks/patterns/templates for assurance cases within an application domain, make excellent sense. *Sufficiency* of the safety argument is also of interest, and there have been a few different approaches in this regard. One recent approach is described in [3]. There are a number of assurance case frameworks that have been suggested for medical device certification [1,30,33]. We would go a little further, and suggest that a reasonably prescriptive template be provided to insulin pump manufacturers, and that the manufacturers be given guidelines on how (and why) to use the template. This introduces some problems – political and technical. The political problems all stem from the fact that *prescription* seems to be a dirty word in software. However, we need to come to terms with the fact that most branches of engineering are quite prescriptive and conservative in their approved approaches to the development of safety critical systems, that they do this for good reasons, and that they often do manage to keep just behind the curve of innovation, so that there is progress as new methods pass the stage from *radical* to *normal* design. We should also realize that there are good ways and bad ways of mandating prescriptive approaches. The technical problems relate to having to know more about our domain than we think we do. Our belief is that we do know enough to get started. This paper presents one approach to getting started on more prescriptive approaches to assurance cases for insulin pumps.

Most assurance cases that we have seen have used a hazards analysis to drive and structure the assurance case. We do not believe that this is the best way to structure the case. It is difficult to determine and show that any hazard analysis is complete, in the sense that all relevant hazards have been identified. If we use the structure of the hazard

analysis to structure the assurance case, we are then going to find it very difficult to argue that the assurance case is sound. We could, of course, add a claim that deals with the case that not all hazards were identified, but that seems to lead into having to then show the argument we could have used if the hazard analysis was not the driver of the structure. It seems clear to us then, that hazard analyses should be included at lower levels of the arguments. Our suggested template has three claims at the top level:

- The requirements accurately and consistently specify the behaviour of the insulin pump, such that the pump, if built in compliance with these requirements, will maintain the user's insulin levels so that they are within a safe range for the user
- The pump is built so that its behaviour is compliant with the behaviour specified in the requirements, within specified tolerances
- It will be possible to maintain and operate the pump over its projected lifetime without adversely affecting the safety, security and effectiveness of the pump

A fourth top level claim, *fail-safe*, is a possibility. If fail-safe behaviour is included explicitly in the requirements specification, then we have an option to deal with evidence related to fail-safe behaviour in the verification claim regarding compliance with requirements. We could also separate it out as a separate claim. If details of the fail-safe behaviour are not explicitly included in the requirements, then we definitely need a fourth top level claim regarding fail-safe behaviour.

The real idea behind an assurance case template is to provide guidance to both manufacturers and regulators of the assurance case in a way that directs the manufacturers to develop arguments that are important in that domain, but are not so detailed as to make it a mindless exercise performed solely to convince the regulators.

An idea of what this may look like is presented in Figure 4, in which we have shown the claims and sub-claims for just one of the three top claims, since it is not possible to show the other two claims in the space available. The claims are shown but the strategies, context, and other types of possible nodes are not shown, again because of space limitations. Also, because they are not really relevant to the point we are making. The evidence at the end of the argument chain is blank here, but in an actual template we would include a description of acceptable evidence for each of these paths. We would also use context nodes to describe the rationale for the choice of claims, sub-claims, strategies, etc.

So, how does this help us? It actually helps us in a number of ways.

First of all, for the regulator/certification authority:

- It conveys to the manufacturers, in a very explicit way, the type of argument and supporting evidence required in order to obtain approval for marketing an insulin pump;
- The main structure of the assurance case is pre-determined (not the specific content of a node in many cases), and the regulator should have (must have) done sufficient analysis to determine that arguments based on this structure, with relevant content in the nodes, should produce adequate arguments;
- The consistency of the structure will allow the regulator to develop expertise in what content leads to adequate assurance cases, and will be able to audit submissions to determine if there is something important lacking, whether the specific evidence presented does not adequately support a specific argument, etc.

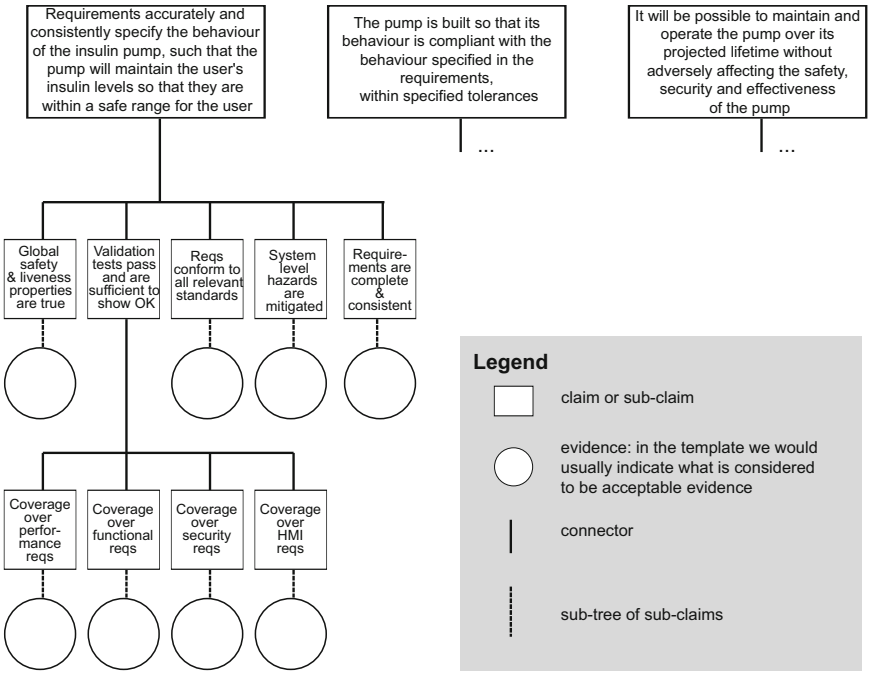

Fig. 4. Example Extract from an Assurance Case Template

Secondly, for the manufacturer:

- There should be much better predictability of the regulatory evaluation process, and this is one of the major concerns of manufacturers;
- It provides guidance based on the regulator's experience and knowledge of problems that are common in a specific domain;
- Manufacturers need to develop products knowing that they will be safe, not prove that they are safe after development. Templates like this can help direct development so that the product will be safe, secure and dependable.

We already presented an overview in graphical form of the top level claim *"The requirements accurately and consistently specify the behaviour of the insulin pump, such that the pump, if built in compliance with these requirements, will maintain the user's insulin levels so that they are within a safe range for the user"* in Figure 4. Now, as another example, we show a template in text form[7] that supports the claim that *"The pump is built so that its behaviour is compliant with the behaviour specified in the requirements, within specified tolerances"*. Possible sub-claims are as follows:

[7] Although most assurance case research uses graphical representations of the case, we are not yet convinced that this is effective for large, practical assurance cases. Tabular text form for cases may be more appropriate in that we may be able to design the layout so that it facilitates views of the 'big picture' more effectively than graphical layouts can.

- The pump's behaviour is compliant with the behaviour specified in the requirements, within specified tolerances
  - All requirements in the system requirements spec (REQ) are present and equivalent in the system level design (DGN)
    - *Evidence:* DGN review report that shows all requirements from REQ are present in DGN
    - *Evidence:* DGN verification report shows equivalence of requirements in REQ with their representation in DGN [may be omitted if model and notation in REQ and DGN are the same]
  - All behaviour specified in DGN that is not in REQ has been justified, and DGN is correct, complete, unambiguous and consistent
    - *Evidence:* DGN review report documents this justification
  - All software related behaviour in DGN is represented equivalently in the software requirements (SRS), and
  - All hardware related behaviour in DGN is described in M-I or O-C mappings (as in the 4 variable model [25], and transfer events [31])
    - The software design (SDD) correctly includes all behaviour in the SRS
      - *Evidence:* SDD verification report shows equivalence of requirements in SRS with their representation in the SDD, within tolerance
    - All behaviour specified in the SDD that is not in the SRS has been justified, and the SDD is correct, complete, unambiguous and consistent
      - *Evidence:* SDD review report shows justification
      - *Evidence:* SDD verification reports demonstrates SDD is complete, unambiguous and consistent
  - All behaviour in the SDD is equivalently implemented in code
    - *Evidence:* Code verification report shows equivalence of behaviour in code with its representation in the SDD
    - *Evidence:* Test reports demonstrate that no tests fail
  - All behaviour implemented in code and not in the SDD has been justified
    - *Evidence:* Code review report shows justification
  - Code is complete, unambiguous and *"free from coding defects"*
    - *Evidence:* Code verification report shows complete and unambiguous
    - *Evidence:* Reports from two independent static code analysis tools show code is *"free from coding defects"*

There are clearly some claims missing from this template – they are missing simply because we do not want to complicate the claim/argument structure for the purpose of this illustration. In particular, all the claims related to hazard analysis are missing. Similarly, fail-safe items are also ignored. However, there is enough detail here to see that there is some obvious prescription implicit in this template.

1. Required documents tell us something about the process – system level requirements, system level design, software requirements, software design, system level design review report, software requirements review report, software design review report, software design (mathematical) verification, code verification, testing (lots of it), static analysis;
2. Mathematical verification is required;

3. The above point implies that the software requirements and software design must be mathematically specified;
4. More subtly, it gives the manufacturer the option to form the SRS directly from the DGN without rewriting the behaviour in any way.

c_InfuFlRt

Inputs:

Name	Description	Initial Value	Reference
M_BolAmt	A request for an amount of bolus	N/A	–
f_BasProf	The current basal profile in the system	N/A	–
c_SysOp$_{-1}$	The previous value of the system operation indicator	NoOp	4.3
c_BolInProg$_{-1}$	The previous value of the bolus administration notification	NoBolInProg	4.4
c_BadDelivNotif$_{-1}$	The previous value of the bad delivery notification	NoBadDelivNotif	4.7

Output:

Name	Description
c_InfuFlRt	The flow rate of the insulin to be delivered

Function Table:

<table>
<tr><td colspan="4"></td><td>Result</td></tr>
<tr><td colspan="4">Condition</td><td>c_InfuFlRt</td></tr>
<tr><td rowspan="3">c_SysOp$_{-1}$ = Op</td><td rowspan="3">c_BadDelivNotif$_{-1}$ $\neq$ MaxDose</td><td rowspan="2">c_BolInProg$_{-1}$ = BolInProg</td><td>f_BasProf + M_BolAmt * Δt $\leq$ k_MaxTotFlRt</td><td>f_BasProf(t$_{now}$) + M_BolAmt * Δt</td></tr>
<tr><td>f_BasProf + M_BolAmt * Δt $>$ k_MaxTotFlRt</td><td>k_MaxTotFlRt</td></tr>
<tr><td colspan="2">c_BolInProg$_{-1}$ = NoBolInProg</td><td>f_BasProf(t$_{now}$)</td></tr>
<tr><td colspan="3">c_BadDelivNotif$_{-1}$ = MaxDose</td><td>0</td></tr>
<tr><td colspan="4">c_SysOp$_{-1}$ = NoOp</td><td>0</td></tr>
</table>

Fig. 5. Example tabular requirements specification from [29]

This paper is about insulin pumps, yet if we examine the suggested items in the template that we have presented, other than some very high-level claims we cannot actually see anything specific to insulin pumps – it is reasonably general. This is an excellent demonstration of why such a template is useful. It is possible to structure the assurance case in a way that reviewers/certifiers build expertise in what should be presented in particular sections, and it will still **not** be a mindless *fill-in the blanks* exercise. The reason we can *claim* this, is that the domain and problem specific claims, arguments and evidence will be necessary and visible at lower levels. They are essential

to the overall argument. The fact that a template is used to say what should be proved, in no way diminishes the intellectual burden on the developers to provide the appropriate, problem specific sub-claims, arguments and evidence.

For example, in order to support the sub-claim "Requirements are complete & consistent" from the template in Figure 4, once the system's monitored and controlled variables have been identified, the developers may choose to use tabular specifications for the SRS such as the one in Figure 5. Part of the evidence to support the claim that the requirements are complete and consistent would be that that the value of every controlled variable is determined by a single table (or composition of tables) and each table is complete (no missing input cases) and disjoint (no ambiguities). Figure 5 defines the behaviour of the infusion flow rate which is given by the flow rate output to the pump based on the current basal profile and any requested bolus. Due to space limitation we refer the reader to [29] for a more detailed description of the requirement. The important thing to note from the example is that it is easy to inspect that the table is complete and disjoint. Further, these properties of the table can be easily formally verified by using a tool such as [9].

Another low level sub-claim in the assurance case may be that no insulin will be delivered when the maximum dosage has already been reached. The evidence to support this could be a reference to appropriate test cases, as well as a reference to the tabular expression in Figure 5, in which we see that when $\text{c\_BadDelivNotif}_{-1} = \text{MaxDose}$, the value of c_InfuFlRt is 0.

Evaluation of the Assurance Case. Although the assurance case template is used by the manufacturer, it is presented in this section since it is developed by the regulator. The regulator's task is not finished though - the regulator has to evaluate the submitted assurance case.

There are a number of ways to achieve this, and some of these have been presented in the literature already [3,33]:

- The regulator audits one or more *slices* of the assurance case. A 'slice' may be defined by following a path of claim, sub-claim, evidence. It is likely that the regulator will know that specific claims are problematic for the insulin pump (or whatever other device is being evaluated), and will want to audit those claim-slices. If the audit is close to perfect, it is likely that the regulator will simply then go through a check-list of items, such as the safety requirements documented by the manufacturer. If the audit uncovers problems, the regulator may immediately halt the evaluation and inform the manufacturer, or they may start looking for more detailed evidence that the submission is poor so as to have an overwhelming case for denial.
- Another way of evaluating the assurance case would be to go through the entire assurance case, comparing the submission with the regulator's model case (documented or un-documented).
- Yet another process that could be followed is for the regulator to make an *approval assurance case* (or a *denial assurance case*). This would document the claims for approval (or denial) in a claim, sub-claim, evidence structure. Regulators do this now, but it is implicit. Many people believe that assurance cases are effective simply

because they force us to make our arguments explicit, and this should work equally well for evaluation as it does for development.

All of these involve *confidence* as a major component of the decision process. Some of the literature already cited deals with this, and it is a growing research topic in assurance cases. However, we (personally) do not have enough evidence yet to draw conclusions about how confidence should be presented (as a separate case, for example), or how it should be evaluated.

Our current preference for an assurance case evaluation process would be the audit or the approval (denial) case – or a combination of the two.

6 Conclusion

In this paper we described some of the challenges in developing and certifying a generic insulin infusion pump. We then outlined ways in which to address these challenges, including a partial assurance case template, justifying a product focused approach to the certification evidence and evaluation, and taking into account one of the main certification challenges – reducing the variation in submitted assurance case structure. An important point is that a prescriptive template provides structure that is useful and appropriate in multiple domains and for multiple specific applications, and that, if carefully constructed, it should not lead to mindless completion of the template, since the lower levels of the template require specific claims and evidence that depend on the specific application. There is still important work to be done in identifying the form of evidence required, how to evaluate the evidence (how to 'measure' the degree to which specific attributes have been achieved), and how to evaluate the assurance case itself. The important aspect of *confidence* in assurance cases will be central to some of this research.

References

1. Ankrum, T.S., Kromholz, A.H.: Structured Assurance Cases: Three Common Standards. In: HASE 2005: 9th IEEE International Symposium on High-Assurance Systems Engineering, pp. 99–108 (2005)
2. Associated Press: Insulin Pumps Vulnerable to Hacking, `http://www.foxnews.com/tech/2011/08/04/insulin-pumps-vulnerable-to-hacking/`
3. Ayoub, A., Chang, J., Sokolsky, O., Lee, I.: Assessing the Overall Sufficiency of Safety Arguments. In: SSS 2013: 21st Safety-critical Systems Symposium. LNCS. Springer (2013)
4. Bergenstal, R.M., Tamborlane, W.V., Ahmann, A., Buse, J.B., Dailey, G., Davis, S.N., Joyce, C., Peoples, T., Perkins, B.A., Welsh, J.B., et al.: Effectiveness of Sensor-augmented Insulin-pump Therapy in Type 1 Diabetes. New England Journal of Medicine 363(4), 311–320 (2010)
5. Black, P.E.: Samate and Evaluating Static Analysis Tools. Ada User Journal 28(3), 184–188 (2007)
6. Bloomfield, R., Bishop, P.: Safety and Assurance Cases: Past, Present and Possible Future–an Adelard Perspective. In: Making Systems Safer, pp. 51–67. Springer (2010)
7. Carollo, K.: Can Your Insulin Pump Be Hacked?, `http://abcnews.go.com/blogs/health/2012/newline04/10/can-your-insulin-pump-be-hacked/`

8. Dooren, J.C.: FDA Sees Increasing Number Of Insulin Pump Problems, http://online.wsj.com/article/SB10001424052748703862704575099961829258070.html
9. Eles, C., Lawford, M.: A Tabular Expression Toolbox for Matlab/Simulink. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 494–499. Springer, Heidelberg (2011)
10. FDA: Analysis of Premarket Review Times Under the 510(k) Program, http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm263385.htm
11. FDA: Use of Standards in Substantial Equivalence Determinations, http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm073752.htm
12. FDA: Guidance – General Principles of Software Validation (2002)
13. FDA: Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices (2005)
14. FDA: Guidance – Total Product Life Cycle: Infusion Pump-Premarket Notification Submissions [510(k)] Submissions (2010)
15. Hatcliff, J., Heimdahl, M., Lawford, M., Maibaum, T., Wassyng, A., Wurden, F.: A software certification consortium and its top 9 hurdles. Electronic Notes in Theoretical Computer Science 238(4), 11–17 (2009)
16. Hawkins, R., Habli, I., Kelly, T., McDermid, J.: Assurance Cases and Prescriptive Software Safety Certification: A Comparative Study. Safety Science 59, 55–71 (2013)
17. Horowitz, B.T.: Cellnovo's Cloud System Monitors Diabetes in Real Time, http://www.eweek.com/c/a/Health-Care-IT/Cellnovos-Cloud-System-Monitors-Diabetes-in-Real-Time-520914/
18. International Electrotechnical Commission: IEC 62304: 2006 Medical Device Software–Software Life Cycle Processes (2006)
19. Klonoff, D.C., Paul, N.R., Kohno, T.: A Review of the Security of Insulin Pump Infusion Systems. Journal of Diabetes Science and Technology 5(6) (2011)
20. Leveson, N.: Engineering a Safer World: Applying Systems Thinking to Safety. MIT press (2012)
21. Maibaum, T., Wassyng, A.: A Product-Focused Approach to Software Certification. Computer 41(2), 91–93 (2008)
22. NRC: Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems, http://pbadupws.nrc.gov/docs/ML0525/ML052500547.pdf
23. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM 15(12), 1053–1058 (1972)
24. Parnas, D.L., Clements, P.C., Weiss, D.M.: The Modular Structure of Complex Systems. In: 7th International Conference on Software Engineering, pp. 408–417. IEEE (1984)
25. Parnas, D.L., Madey, J.: Functional documents for computer systems. Science of Computer programming 25(1), 41–61 (1995)
26. Potti, L.G., Haines, S.T.: Continuous subcutaneous insulin infusion therapy: a primer on insulin pumps. Journal of the American Pharmacists Association 49(1), e1–e17 (2009)
27. Raghunathan, A., Jha, N.K.: Hijacking an Insulin Pump: Security Attacks and Defenses for a Diabetes Therapy System. In: IEEE 13th International Conference on e-Health Networking, Applications and Services, pp. 150–156. IEEE (2011)
28. Siebert, C.: Diabetes control and complications trial (DCCT): Results of the feasibility study and design of the full-scale clinical trial. Controlled Clinical Trials 7 (1986)
29. Stribbell, J.: Model Based Design of a Generic Insulin Infusion Pump. M.Eng. Report, McMaster University (2013)

30. Sujan, M.-A., Koornneef, F., Voges, U.: Goal-Based Safety Cases for Medical Devices: Opportunities and Challenges. In: Saglietti, F., Oster, N. (eds.) SAFECOMP 2007. LNCS, vol. 4680, pp. 14–27. Springer, Heidelberg (2007)
31. Wassyng, A., Lawford, M.: Lessons Learned from a Successful Implementation of Formal Methods in an Industrial Project. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 133–153. Springer, Heidelberg (2003)
32. Wassyng, A., Maibaum, T., Lawford, M., Bherer, H.: Software Certification: Is There a Case against Safety Cases? In: Calinescu, R., Jackson, E. (eds.) Monterey Workshop 2010. LNCS, vol. 6662, pp. 206–227. Springer, Heidelberg (2011)
33. Weinstock, C.B., Goodenough, J.B.: Towards an Assurance Case Practice for Medical Devices. Tech. rep., DTIC Document (2009)
34. Zhang, Y., Jones, P.L., Klonoff, D.C.: Second insulin pump safety meeting: summary report. Journal of Diabetes Science and Technology 4(2), 488 (2010)