Journal of Diabetes Science and Technology
Volume 5, Issue 6, November 2011

ORIGINAL ARTICLE

Generic Safety Requirements for Developing Safe Insulin Pump Software

Yi Zhang, Ph.D.,[1] Raoul Jetley, Ph.D.,[1] Paul L. Jones, MS/CE,[1] and Arnab Ray, Ph.D.[2]

Abstract

Background:

The authors previously introduced a highly abstract generic insulin infusion pump (GIIP) model that identified common features and hazards shared by most insulin pumps on the market.

The aim of this article is to extend our previous work on the GIIP model by articulating safety requirements that address the identified GIIP hazards. These safety requirements can be validated by manufacturers, and may ultimately serve as a safety reference for insulin pump software. Together, these two publications can serve as a basis for discussing insulin pump safety in the diabetes community.

Methods:

In our previous work, we established a generic insulin pump architecture that abstracts functions common to many insulin pumps currently on the market and near-future pump designs. We then carried out a preliminary hazard analysis based on this architecture that included consultations with many domain experts. Further consultation with domain experts resulted in the safety requirements used in the modeling work presented in this article.

Results:

Generic safety requirements for the GIIP model are presented, as appropriate, in parameterized format to accommodate clinical practices or specific insulin pump criteria important to safe device performance.

Conclusions:

We believe that there is considerable value in having the diabetes, academic, and manufacturing communities consider and discuss these generic safety requirements. We hope that the communities will extend and revise them, make them more representative and comprehensive, experiment with them, and use them as a means for assessing the safety of insulin pump software designs. One potential use of these requirements is to integrate them into model-based engineering (MBE) software development methods. We believe, based on our experiences, that implementing safety requirements using MBE methods holds promise in reducing design/implementation flaws in insulin pump development and evolutionary processes, therefore improving overall safety of insulin pump software.

J Diabetes Sci Technol 2011;5(6):1403-1419

Author Affiliations: [1]Office of Science and Engineering Laboratories, Center for Device and Radiological Health, U.S. Food and Drug Administration, Silver Spring, Maryland; and [2]Fraunhofer Center for Experimental Software Engineering, College Park, Maryland

Abbreviations: (BG) blood glucose, (GIIP) generic insulin infusion pump, (MBE) model-based engineering, (V&V) verification and validation

Keywords: insulin pump, model-based engineering, safety requirement, software

Corresponding Author: Yi Zhang, Ph.D., Office of Science and Engineering Laboratories, Center for Device and Radiological Health, U.S. Food and Drug Administration, 10903 New Hampshire Avenue, Silver Spring, MD 20993; email address Yi.Zhang2@fda.hhs.gov

Introduction

Insulin pumps have been used for many years by people with diabetes to help achieve rapid, precise, and tight glycemic control. The use of these pumps has proven to be fairly effective in helping people with diabetes to achieve a specified basal-bolus regimen and to establish desired blood sugar levels, contributing to a significant improvement in the quality of life of persons with diabetes.[1] Effective as they are, insulin pumps have been implicated in a significant number of adverse events, as documented in the Food and Drug Administration's Manufacturer and User Facility Device Experience database.[2] The potential for insulin pumps to cause unintended and harmful consequences are rooted in various factors, including latent development and manufacturing errors, use of increasingly complex technologies, differences in individuals' physiology and lifestyle, user errors, poor human-factor design decisions, device mobility, and environmental issues.

Modern insulin pumps depend increasingly on software for new features. Software is increasingly responsible for safety functions such as dosage control, interpreting user input and providing display output, and mitigating certain hazards through alarms and alerts. However, due to complexity, software designs may fail to account for foreseeable operating conditions or contain latent design flaws and code defects, resulting in potential pump failure or patient harm. Therefore, a rigorous hazard analysis and software development process must be carried out and validated before the device can be considered ready for patient use.

Evaluating the safety of insulin pump designs, particularly in the context of software, can be difficult; again, due to complexity. Some of this complexity stems from the diversity of use features, each with their own special risks, and issues associated with mobility and changing environments. Currently, there are no suitable reference standards that establish performance and safety criteria to aid in the evaluation process.

This article presents a core set of safety criteria for a generic insulin infusion pump (GIIP) model.[3] In general, the safety criteria presented here for the GIIP model serve to establish design requirements that will eliminate, protect against, or warn patients with diabetes of potential hazardous situations. The safety criteria presented are not exhaustive. They require additional analysis, in general, and further device-specific analysis, in particular. We envision these criteria being extended and used by different stakeholders in different meaningful ways. For example:

1. The safety criteria can be used to establish a basis for community discussion and lay the foundation for developing insulin pump (software) safety consensus standards.

2. Manufacturers can use these criteria, instantiated with details of their own devices, to determine whether their devices have sufficiently addressed these safety concerns.

3. Regulators might use the criteria as a safety reference in assessing the safety of submitted insulin pump designs.

The safety criteria presented in this article might also be exploited in a model-based engineering (MBE)[4] development process to help ensure the correctness and completeness of any insulin pump designs developed. "Model-Based Engineering is about elevating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system".[5] Model-based engineering produces models as the primary development artifact, enabling automated checking for design errors early in the life-cycle development process. Model-based engineering has been used extensively in high-confidence domains such as aerospace and automotive software engineering.[5,6]

Caveats

Safety criteria, or safety requirements, presented in this article are intended to establish baseline safety criteria for the GIIP model. They should not be considered as exhaustive or mandatory, either for the GIIP model itself or for any insulin pump design. Complying with these requirements does not guarantee that the GIIP model, or any insulin pump design, is acceptably safe and will not cause potential harm to end users.

Manufacturers who enforce these general safety requirements in their products may benefit from checking their products against this independent work. If they

do so, they are responsible for deleting, revising, and supplementing these requirements to accommodate their own safety-related design decisions. Manufacturers bear the responsibility for assuring the acceptability of risk-control measures implemented in their products.

Utility of these safety requirements depends on the ability of manufacturers to instantiate these requirements with design and implementation details specific to their own products. Manufacturers must decide how to examine their products and evaluate their conformance with these requirements.

Background

The GIIP model architecture is briefly summarized here to provide necessary background information. Interested readers can find a more complete description of the GIIP model in our previous GIIP (preliminary) hazard analysis paper.[3]

The GIIP model was first introduced as an abstraction of functions and features commonly found in home-use insulin pumps on the market or likely to be on the market soon. **Figure 1** illustrates the system boundary for the GIIP, which includes the model itself, the user, the infusion set (user/device drug delivery connection), and the environment. Notably, a wireless remote control is excluded from this system boundary.

From an architectural viewpoint, the GIIP model comprises a number of functional components. At the core of the architecture is a pump controller, an abstract representation of generic insulin pump software. The primary function of the pump controller component is to command the pump delivery mechanism to propel, at a prescribed rate and for a prescribed duration, insulin stored in the drug reservoir to the patient through the drug delivery interface and the infusion set.

The pump controller bears other responsibilities to ensure correct and robust operation of the model. These responsibilities include interacting with the patient through a user interface; recommending appropriate bolus dosages with the help of a bolus calculator and a food database; managing and checking parameters and programs related to insulin administration; alerting the patient when abnormal conditions arise; and logging important data and events during pump use to facilitate clinical use analysis and problem diagnosis.

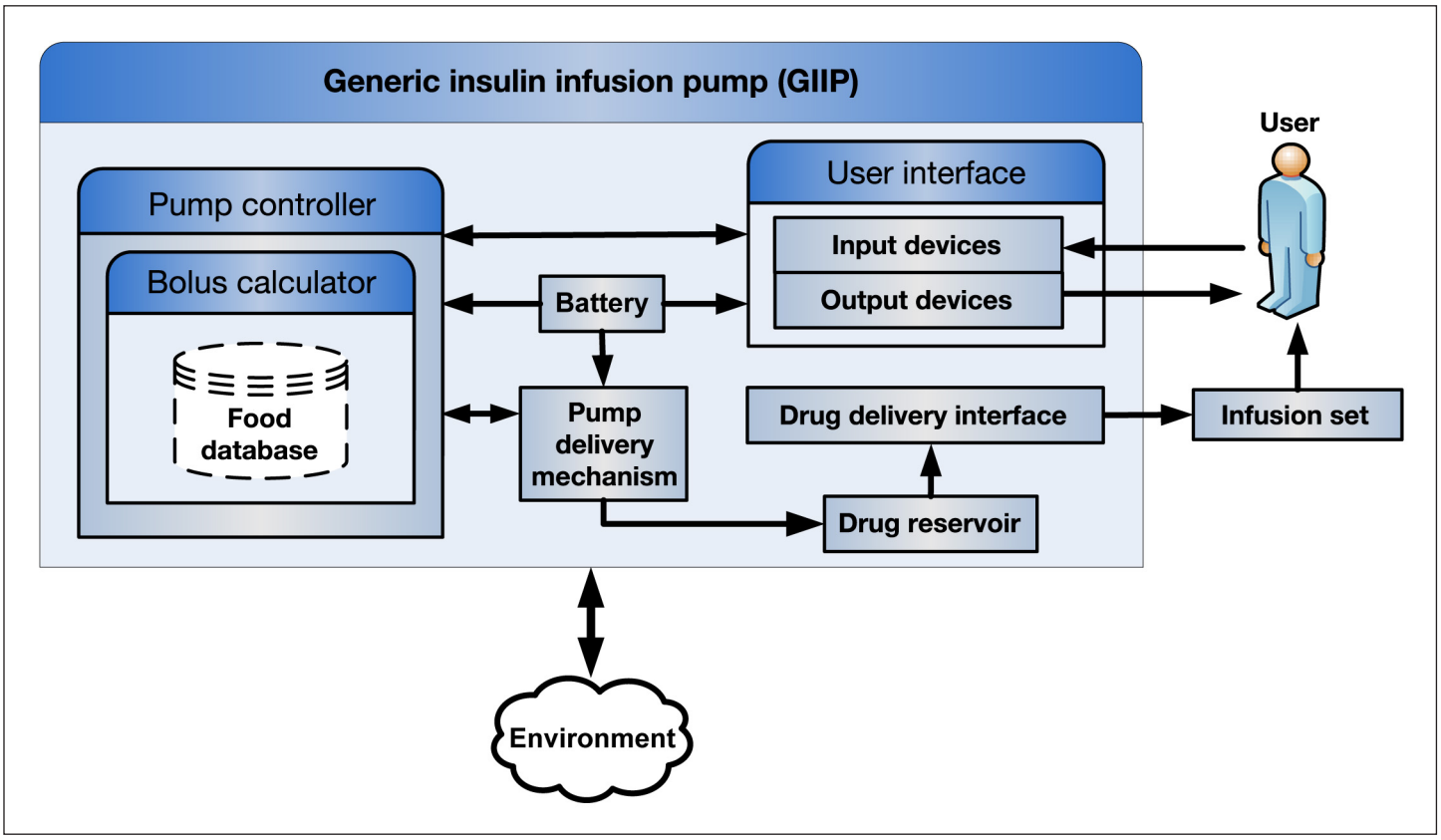

Figure 1. System architecture of generic insulin infusion pump (GIIP).

It should be noted that the GIIP model is intended to capture the common behavior of many insulin pumps, not only modern pumps but also those obsolete ones. Thus, many features pioneered by specific pump manufacturers were intentionally excluded from the model. For example, remote controllers are not included in the GIIP model because some obsolete insulin pumps do not have remote-control devices. However, since more and more modern insulin pumps incorporate the remote-control feature, making it a common feature for insulin pumps, we plan to extend the GIIP system to include remote control in our future work.

The authors have conducted a preliminary hazard analysis for the GIIP model, enumerating typical hazardous situations as well as their potential causes. Detailed results of this analysis can be found in our GIIP hazard analysis paper.[3]

Generic Safety Requirements for GIIP Software

To varying degrees and in various ways, software can be used to mitigate potential insulin pump risks. For example, software can be designed to react to a user command for a correction bolus when unnecessary. In particular, software can issue alerts to the user when he/she tries to command a correction bolus when the blood glucose (BG) level is low, so that the chance of a user getting an inappropriate bolus is reduced. Software can also be used to coordinate functions of various components within the pump to ensure safe and robust operation of the pump. One such example is to use the combination of software and delivery flow sensors to detect and promptly report an inaccurate insulin delivery rate.

There are many circumstances where software is incapable, ineffective, or inefficient in mitigating potential risks. Physiological or biological risks are typical examples. There are also circumstances where software needs to be used in combination with other risk-control measures to mitigate insulin pump risks efficiently. For example, software is often used to detect if the user programs a delivery with incorrect parameters. In contrast, patient training and device labeling are frequently used as risk-control measures to reduce the likelihood that the user makes such mistakes. Thus, use of software detection in conjunction with labeling and patient training can mitigate the risk of incorrect delivery programs to a greater degree than if any of these measures were used alone.

Therefore, an important consideration in insulin pump design is to determine whether and how software can reduce risks. This article focuses on identifying a core set of software-based risk control measures or safety requirements, which are then encapsulated in the GIIP model. Various formal analysis methods can be applied to these requirements to establish minimum safety properties for real-world insulin pumps.

We present safety requirements that we identified in **Tables 1–6** in the **Appendix**, where safety requirements in the same table focus on the same aspect of pump operation. One thing worth noting is that the identification of GIIP safety requirements is strictly constrained to the system boundary established for the GIIP model. For example, we impose no safety requirements on remotely controlling the model because such a feature is excluded from the current GIIP model. If manufacturers decide to use remote-control devices in their pumps (many of them already do), they take on the responsibility of developing reasonable safety requirements to assure that their pumps coordinate appropriately with their remote-control devices. Similarly, the remote-control devices must be designed and implemented in a manner that ensures operational safety (which includes security considerations).

Risk-control measures may be implemented in the form of design decisions that eliminate the risk or protective actions and instructions that reduce the risk. This observation provides a basis for developing the GIIP model safety requirements that are enumerated in the **Appendix**, where:

- Certain safety requirements are intended to clarify the ambiguities in scheduling and administration of insulin therapy. One such example is requirement 1.3.5, which prohibits overlapping of normal boluses. Requirements in this category permit the user to monitor and track insulin administration without misunderstandings, reducing the likelihood of the user programming inappropriate insulin delivery plans.
- Safety requirements focusing on event logging (**Appendix Table 4**) enforce the collection of useful diagnostic information with acceptable accuracy and precision when the pump malfunctions.

  Although these requirements do not protect the user from adverse events caused by the pump, they do assist in a root cause analysis of pump malfunctions, which can help prevent similar problems from reoccurring.

- The rest of the safety requirements aim to address foreseeable hazardous situations and their causes identified in the previous GIIP hazard analysis paper.[3] Working alone or together, each requirement is meant to (1) eliminate the occurrence of a particular cause; or (2) provide prompt and precise notification to the user whenever the cause arises during pump operation, so that the user can intervene and eliminate it before any adverse effect is realized.

  For some causes, software can accomplish both goals. For example, in order to eliminate the presence of air-in-line, software will not only reduce the chance of air-in-line by guiding the user to prime the pump correctly, but also notify patients whenever air bubbles are detected in the delivery path.

These requirements can be used in the development of most insulin pumps because the abstractions on which they are based are free of low-level, device-specific implementation details. The requirements are intentionally presented in a flexible format, in order to provide manufacturers some freedom in utilizing these requirements.

Some of these safety requirements carry parameters that allow manufacturers to accommodate arbitrary safety margins. For example, in requirement 1.6.1 (in **Table 1** of the **Appendix**), the pump's sensitivity to air bubbles is measured by the minimum size of air bubbles, which is defined as parameter y in the requirement, that will trigger an air-in-line alarm. The smaller y is, the more sensitive the pump will be to air bubbles. While utilizing this requirement, manufacturers will have the freedom of assigning any values to y, corresponding to their design decisions. However, manufacturers have to ensure that the assigned values comply with clinical performance standards or generally accepted practices, or more generally, are appropriate to assure safety.

We divide the safety requirements into six different categories based on aspects of pump functionality to facilitate crosschecking processes. Each category has its own table in the **Appendix**, as follows:

1. insulin administration
2. user interface
3. alarm system
4. event logging
5. battery management
6. interaction with the environment

Although safety requirements in category 6 are not purely software related, we have included them here to highlight the importance of safety issues related to environmental factors, given the fact that insulin pumps are often used in diverse and dynamic environments. We encourage manufacturers to take these issues into consideration when designing their products.

Discussions—Using MBE Methods in Safety-Critical Environments

The value of safety requirements presented in this article lies in their utility for examining the correctness of real-world insulin pump software designs via the GIIP model. In particular, the resulting safety requirements can be modeled as an independent test framework, against which a real-world insulin pump software design and implementation can be verified. Manufacturers can also adopt other software verification and validation (V&V) techniques, such as model checking, testing, walk-throughs, etc., to check if the software in their products satisfies these safety requirements. However, different V&V techniques provide different degrees of confidence in checking consistency between software and safety requirements. Some safety requirements (e.g., requirements related to human factors) are not particularly amenable to automated checking methods and therefore require other V&V methods, such as clinical or patient-use experiments. Thus, it is up to manufacturers to choose appropriate V&V techniques and to assure that results produced by the chosen techniques are convincing and trustworthy.

Of course, a real-world insulin pump software design can adopt an alternative safety measure rather than the one defined by the GIIP requirements. In such circumstances, the properties of these safety requirements can still be used to determine whether the alternative measure achieves equivalent or better safety than the GIIP model.

Based on previous experience,[7] we believe that integrating safety requirements into a MBE paradigm can help detect and eliminate flaws and defects in insulin pump software designs and implementations. **Figure 2** illustrates potential ways of integrating safety requirements into the MBE software-development lifecycle.

Model-based engineering developers do not, in general, produce the same design artifacts as conventional software-development practices do. Instead, they formalize software design as a series of design models. In particular, developers establish a high-level software design abstraction and then progress through a series of elaboration steps to lower-level, more detailed executable models, from which code is ultimately generated. As mathematical abstractions of software design, design models help to eliminate potential ambiguity and confusion underlying the design. Formal verification can also be applied to design models to mechanically examine all possible behaviors of these models, often detecting subtle error conditions not considered by domain experts and developers or typically found by conventional design review and validation techniques.

In a model-based development process, safety requirements can be used by manufacturers at two different stages:

1. Design verification. After the software design is captured in (preliminary or refined) design models, developers can utilize safety requirements to check

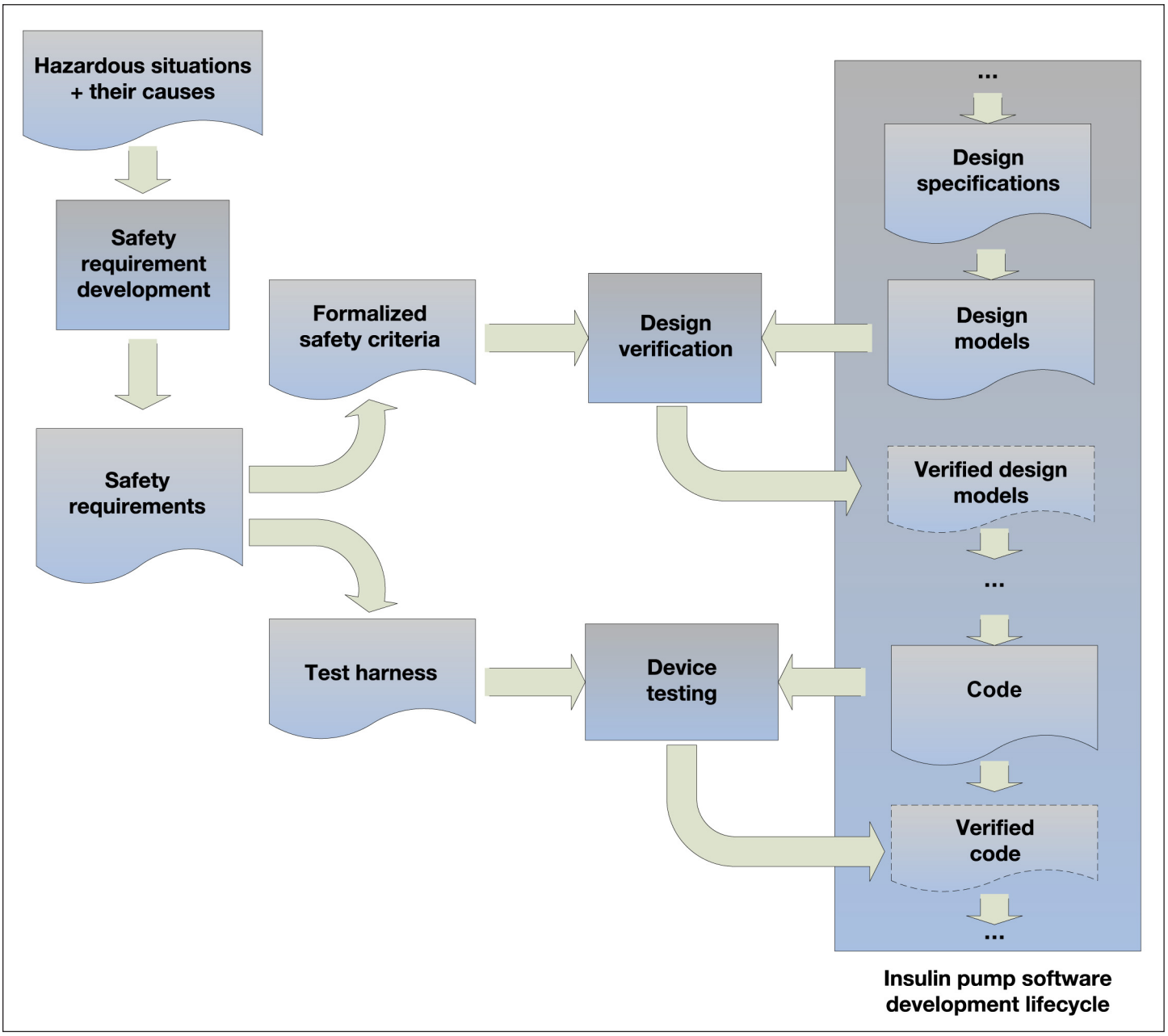

Figure 2. Integrating safety requirements into software development lifecycle.

the behavior of these models to ensure that they do not violate any requirements. As a result, flaws existing in the design can be filtered out before models are translated into a final implementation.

To utilize safety requirements at this stage, developers can first formalize them into logical or mathematical criteria (e.g., temporal logic[8] formulae or monitoring models) and then seek assistance from formal verification techniques, such as model checking[9] and instrumentation-based verification,[10] to conduct thorough checking of the design models against formalized criteria.

However, not all safety requirements can be formalized; in fact, safety requirements may demonstrate a great diversity in their characteristics (e.g., some requirements are qualitative and some others are quantitative). This makes it impossible to formalize all safety requirements, especially those qualitative ones, into a computer-verifiable style. For those requirements that cannot be formalized, conventional V&V techniques other than formal verification can be used to assure that the software design satisfies them.

It should also be noted that the safety requirements presented in this article are derived based on an abstract model. If manufacturers are willing to apply these requirements to evaluate the software design of their products, it is more beneficial to formalize these requirements—if they can be formalized—after all related design details have been articulated.

2. Implementation verification. After a device design has been implemented, safety requirements can be used to check if the software faithfully implements the design. Here, the device design serves as a kind of safety reference standard because it has been proven safe, with respect to safety requirements, at the first stage.

   Developers can translate safety requirements into explicit test cases, and then apply the test cases to their software to examine whether the software produces the expected output. Unexpected output may indicate that the code implementation deviates from the original design. Developers can also turn safety requirements into safety checks (or assertions in software engineering terminology), and place these checks into the software, so that execution of the software will terminate if the assertions are violated.

Notably, the MBE process can also be used in a corrective action process. For example, design and implementation changes for corrective actions can be verified against the safety model to establish the fact that prior safety properties were not compromised in the process.

Conclusion

A minimal set of safety requirements for a GIIP model has been presented as a step toward establishing an open-source insulin pump-safety reference model that can be helpful in improving the safety and effectiveness of insulin pumps. The requirements presented earlier intend to provide a means for establishing that the GIIP model performs correctly and unambiguously to mitigate some potential, foreseeable real-world risks.

It would be valuable if the diabetes and academic communities and manufacturers would consider and discuss these generic safety requirements for insulin pump software, to extend and revise them, to make them more representative and comprehensive, to experiment with them, and to use them as means for assessing the safety of insulin pump software designs.

We hope that this work will help to reveal flaws in insulin pump software design and hence improve the overall safety of the products. We encourage manufacturers to consider these safety requirements in their insulin pump software development and evolutionary processes.

Acknowledgments:

ASHVINS Group Technology Professionals, Lynn Hilt, Thomas Love and Alin Andea, Miami, Florida; David C. Klonoff, M.D., FACP, Medical Director, Diabetes Research Institute, Mills-Peninsula Health Services, San Mateo, California; Lt Col Mark W. True, M.D., FACP, FACE, Director, Diabetes Center of Excellence, Lackland Air Force Base, San Antonio, Texas; Nugget Burkhart, B.S.N., M.A., NP, BC-ADM, CDE, Diabetes Care Manager, Department of Medicine, Kaiser Permanente Medical Center, San Francisco, California; Meaghan Devlin, R.N., Staff Nurse, Massachusetts General Hospital, Boston, Massachusetts; Tamara James, R.N., CDE, Clinical Resource Nurse III, UC Davis Medical Center, Sacramento, California; Irina Nayberg, R.N., B.S.N., CDE, Clinical Research Coordinator, Mills-Peninsula Health Services, San Mateo, California; Gloria Yee, R.N., CDE, Principal Diabetes Instructor, Diabetes Teaching Center, UC San Francisco, San Francisco, California.

References:

1. The Diabetes Control and Complications Trial Research Group. The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. N Engl J Med. 1993;329(14):977–86.
2. FDA MAUDE database. Available from: *http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/PostmarketRequirements/ReportingAdverseEvents/ucm127891.htm*.
3. Zhang Y, Jones PL, Jetley R. A hazard analysis for a generic insulin infusion pump. J Diabetes Sci Technol. 2010;4(2):263–83.
4. Kampfner RR. Model-based development of computer-based information systems. Workshop on Engineering of Computer-Based Systems (ECBS); 1997 Mar 24–28; Monterey, California. p. 354.
5. Estefan JA. Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE MBSE Initiative, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California.
6. Wall SD. Model-based engineering design for space missions. Proceedings of the 2004 IEEE Aerospace Conference; 2004 Mar 6–13; Big Sky, Montana. p. 3907–15.
7. Jetley R, Jones P. Safety requirements based analysis of infusion pump software. Proceedings of the IEEE Real Time Symposium; 2007 Dec; Tuscon, Arizona.
8. Pnueli A. The temporal logic of programs. Proceedings of 18th Annual Symposium on Foundations of Computer Science (FOCS 1977); Providence, Rhode Island. p. 46–57.
9. Clarke EM Jr., Grumberg O, Peled DA. Model Checking. Cambridge, MA; The MIT Press; 1999.
10. Cleaveland R, Smolka SA, Sims ST. An instrumentation-based approach to controller model validation. In: Model-Driven Development of Reliable Automotive Services. Berlin: Springer-Verlag; 2008. p. 84–97.

Appendix

This Appendix lists a minimum set of insulin-pump safety requirements developed for the GIIP model in tabular format. To facilitate tracking of these requirements, each requirement is assigned with a unique ID number and grouped into a table with other requirements that focus on the same aspect of pump operation. In these safety requirement tables, except **Table 4** (event logging-related requirements), a column called Causes to mitigate is introduced to document (the index numbers of) causes of hazardous situations that each safety requirement intends to mitigate.

Among the listed requirements, there are certain exceptions that are not mapped to any particular cause of hazardous situations. In fact, these requirements, as mentioned earlier, are defined to either clear up the ambiguities in pump operation or to provide some protective means to ensure safety even under pump malfunctions (such as requirement 1.4.7). Therefore, these requirements can participate in mitigating any causes that may result in the corresponding hazardous situations.

It should also be noted that, if a cause of hazardous situations is mitigated by a safety requirement with multiple subrequirements (such as requirement 1.4), it is actually mitigated by all of the subrequirements together.

Table 1.
Requirements on Insulin Administration

Req. ID	Requirement Specification	Causes to Mitigate
1.1 Infusion control		
1.1.1	The pump shall suspend all active basal delivery and stop any active bolus during a pump prime or refill. It shall prohibit any insulin administration during the priming process and resume the suspended basal delivery, either a basal profile or a temporary basal, after the prime or refill is successfully completed.	2.14, 8.10.10
1.1.2	The average flow rate in any continuous *x*-minute period shall remain accurate within ±*y*% of the programmed rate.	2.11, 2.12
1.1.3	If the pump allows administering multiple types of insulin, changing drug types and concentrations shall stop any active infusion, remind the user to validate the basal profiles and related parameters, and force the reservoir time and volume to be recomputed.	
1.2 Basal programming and administration		
1.2.1	The pump shall allow the user to program a basal profile with a set of basal rates, ranging from 0.05 to *x* units/hour in 0.05 units/hour increments. For each basal rate in the profile, the user shall define the duration of the particular rate, and the duration shall be set in *y* minute increments. Durations of all basal rates shall not overlap with each other, and shall together cover 24 hours of a day.	
1.2.2	The pump shall allow the user to set at least two basal profiles at the same time, and require the user to activate no more than one profile at any single point in time.	3.9
1.2.3	The pump shall notify the user when a basal profile is activated, and shall administer basal insulin according to the profile immediately after activation.	
1.2.4	The pump shall allow the user to temporarily override the current basal delivery with a temporary basal without changing existing basal profiles, provided that no normal bolus or other temporary basal is in progress. The user shall be required to specify the duration and rate of the temporary basal being programmed.	
1.2.5	The programmed infusion rate of a temporary basal shall not exceed *x* units/hour and the duration of a temporary basal shall not exceed *y* hours.	
1.2.6	The pump shall start to administer a temporary basal immediately after the user confirms it, and resume the previously active basal profile after the temporary basal is finished.	
1.2.7	The pump shall allow the user to stop a temporary basal while it is being administered. When the user chooses to stop a temporary basal, the pump shall either resume the active basal profile prior to the temporary basal or require the user to activate a predefined basal profile.	
1.2.8	If the currently activated basal profile or the currently ongoing temporary basal has been paused for more than *x* minutes, it shall signal an audible alarm every *y* minutes up to *z* hours.	

Continued →

Table 1. Continued

Req. ID	Requirement Specification	Causes to Mitigate
1.3 Bolus calculation and administration		
1.3.1	The pump shall allow the user to set the maximum dosage limit for every normal or extended bolus. For each bolus whose dosage exceeds the limit, the pump shall prompt the user to either confirm this bolus or cancel it.	3.1.2
1.3.2	The pump shall allow the user to define the dosage of a normal bolus in no coarser than *x* units increments.	3.1.2
1.3.3	The pump shall start a valid normal bolus immediately after it is programmed, and deliver it at the highest rate that satisfies requirement 1.3.4.	
1.3.4	The combined flow rate (basal rate + normal bolus rate + extended bolus rate) shall be limited by the maximum flow rate at which the pump can function correctly.	2.7, 2.12
1.3.5	The pump shall not allow a normal bolus to start when another normal bolus is in progress. If the user requests a normal bolus when another normal bolus is in progress, the pump shall issue an alert and deny the request.	2.12, 2.14
1.3.6	The pump shall start a valid extended bolus at the time the user specifies. The extended bolus delivery shall be distributed evenly over its duration.	2.7, 2.12
1.3.7	The user shall be able to stop an active normal or extended bolus. When the user stops a bolus, the pump shall display the amount of insulin that has been delivered for the bolus.	3.1.1–3
1.3.8	If the user changes correction factors, insulin-to-carbohydrate ratios, or target BG levels, the pump shall stop any bolus delivery being administered. If the user changes the system date/time, the pump shall prompt the user to either stop or continue the current bolus administration.	3.1.2, 3.1.3
Requirements 1.3.9–17 are applicable only if the pump recommends correction boluses		
1.3.9	The pump shall allow the user to program either a single correction factor or a set of correction factors to describe his/her sensitivity to insulin over the time of day. Each correction factor shall be defined in the range of *x* mg/dl to *y* mg/dl, in *z* mg/dl increments. If the program allows the user to define a set of correction factors, it shall prompt the user to define the duration for each correction factor in *u*-minutes increments. Durations of correction factors shall not overlap each other and shall cumulatively cover 24 hours of a day.	
1.3.10	The pump shall use the correction factor currently in effect to calculate a correction bolus. At the same time, it shall display the factor to the user through its user interface.	3.1.2
1.3.11	The pump shall allow the user to configure the duration of insulin activity from *x* to *y* hours in *z*-hour increments.	3.1.2, 3.1.3
1.3.12	The pump shall report to the user the BG reading, as well as its input time or the time elapsed since the reading that the pump uses to calculate recommended dosages of correction boluses. The pump shall allow the user to confirm the reading or replace it with a new one.	3.1.2
1.3.13	The pump shall allow the user to define different target BG levels for different periods of the day. If any target BG level that the user inputs is out of the range *x* to *y* mg/dl, the pump shall ask the user to confirm or cancel it.	3.1.2
1.3.14	If the pump does not support reverse correction, it shall not recommend a correction bolus if the user's current BG reading is lower than his/her current target BG level.	3.1.2
1.3.15	The pump shall allow the user to view and modify the dosage of a recommended bolus and to configure the distribution of the bolus between normal and/or extended boluses.	3.1.1–3
1.3.16	If an extended bolus is being delivered while a correction bolus is recommended, the remaining amount of the extended bolus (that is used to correct abnormal BG levels) shall be added to the calculated unabsorbed insulin amount.	3.1.2–3
1.3.17	The amount of unabsorbed insulin shall be retainable after the user changes the date and time in the pump.	3.1.2–3

Continued →

Table 1. Continued

Req. ID	Requirement Specification	Causes to Mitigate
Requirements 1.3.18–22 are applicable only if the pump recommends food boluses		
1.3.18	The pump shall allow the user to program either a single or a set of insulin-to-carbohydrate ratios (food factors) in the range from *x* to *y* g/unit in increments of *z* g/unit. If the pump allows the user to define a set of food factors, it shall prompt the user to define a time segment with *u*-minute increments for each food factor. Time segments of all food factors shall not overlap each other and shall cover 24 hours of the day.	3.1.2
1.3.19	If the pump incorporates a food database to support the calculation of intake carbohydrates, the information contained in the database shall either be verified and approved by qualified nutritionists or be configured and confirmed by the user.	3.1.1
1.3.20	The pump shall use the food factor currently in effect to calculate a food bolus. The pump shall display the factor currently in effect through the user interface.	3.1.1
1.3.21	While calculating a food bolus for a meal, the pump shall require the user to configure (w/o using a food database described in requirement 1.3.20) the number of digestible carbohydrates or all types of ingredients that are related to deciding food-bolus dosage and their amounts projected for the meal intake.	3.1.1
1.3.22	The pump shall allow the user to view and modify the dosage of a food bolus that it suggests and to configure the distribution of the bolus between normal and/or extended boluses.	3.1.1
1.4 Drug reservoir		2.2, 2.4, 2.8, 2.11–15, 3.1.2–3, 3.7
1.4.1	The calculation of the remaining reservoir volume shall be accurate to ±*x* μL.	
1.4.2	The reservoir volume remaining shall be recomputed after the pump is primed.	
1.4.3	The reservoir volume remaining shall be updated after each pump stroke by subtracting the amount of insulin delivered during the stroke.	
1.4.4	The reservoir volume remaining shall be recalculated at the start and end of every basal profile segment, every temporary basal, and every (normal or extended) bolus.	
1.4.5	If the insulin remaining in the drug reservoir is less than *x* units (within a tolerance of ±*y* μL) and an infusion is in progress, a low reservoir alert shall be issued.	2.10, 4.3.7, 4.6.5
1.4.6	If the insulin remaining in the drug reservoir is 0 units (within a tolerance of ±x μL) and an infusion is in progress, an empty reservoir alarm shall be issued.	2.9
1.4.7	The pump shall monitor the insulin (bolus and basal) delivery in progress. When the actual volume delivered differs from the expected delivery by more than *x*%, the pump shall signal an alarm and stop the delivery.	
1.5 Occlusion (requirements 1.5.1–1.5.5 are only applicable if the pump includes tubing as part of its drug delivery interface)		
1.5.1	The pump shall have an occlusion sensor.	2.6, 2.11 2.14, 4.3.7
1.5.2	An occlusion alarm shall be triggered if the pump senses an upstream (insulin-supply side) occlusion.	
1.5.3	An occlusion alarm shall be triggered if the pump senses a downstream (patient side) occlusion.	
1.5.4	The occlusion sensor shall trigger an occlusion alarm whenever the actual flow rate is less than the programmed rate by at least *x*% for *y* seconds due to occlusion. Note that this requirement does not necessarily imply that the occlusion sensor should measure the actual flow rate.	
1.5.5	When an occlusion occurs, the pump shall stop flow and alarm within a maximum delay time of x seconds.	
1.6 Air in line		
1.6.1	An air-in-line alarm shall be triggered within a maximum delay time of *x* seconds if air bubbles larger than *y* μL are detected, and all insulin administrations shall be stopped.	2.1, 4.3.7
1.7 Reverse flow		
1.7.1	During normal use and single fault conditions of the pump, continuous reverse delivery shall not be possible. A single fault condition refers to a situation where a single abnormal external condition arises or one protection means against an adverse health consequence is defective.	2.3

Continued →

Table 1. Continued

Req. ID	Requirement Specification	Causes to Mitigate
1.8 Pump suspension		
1.8.1	When the option to suspend the pump is selected, the current pump stroke shall be completed prior to suspending the pump.	
1.8.2	When the pump is in suspension mode, insulin deliveries shall be prohibited. Any incomplete bolus delivery shall be stopped and shall not be resumed after the suspension.	2.14
1.8.3	If the suspension occurs due to a fault condition, the pump shall be stopped immediately without completing the current pump stroke.	
1.8.4	If the pump has been put in a non-delivery mode for more than x minutes, it shall signal an audible alarm for every *x* minutes up to *y* hours.	8.10.10
1.8.5	When the pump resumes from suspension, calculations shall be performed to synchronize insulin used and remaining reservoir volume.	
1.9 Data integrity		
1.9.1	The user's programming of any basal or bolus shall not take effect until the user has input all required parameters and has reviewed and confirmed the input parameters and programming results.	3.1.1, 8.9.2–6, 8.10.8
1.9.2	The pump shall be protected from operating with corrupted critical data. Critical data includes at least the following: • basal profiles; • temporary basal duration and rate; • the maximum bolus dosage and rate; • normal bolus dosage; • extended bolus duration and rate; • insulin-to-carbohydrate ratios and their effective periods; • insulin correction factors and their effective periods; • food database; • target BG level profiles and their effective periods; • BG readings; • records of previous boluses; • concentration and activity duration of currently loaded insulin; and • duration and time period of recent suspension. The detection of critical data corruption shall stop all active infusion and signal a data corruption alarm.	3.1.2, 3.1.3, 3.7, 3.8

Table 2.
Requirements on User Interface

Req. ID	Requirement Specification	Causes to Mitigate
2.1 Resistance to tampering and accidents		
2.1.1	The pump shall provide a locking option that, once selected, shall allow only the user and authorized personnel to unlock and access the pump status and user records and statistics.	8.10.1, 8.10.9
2.1.2	To avoid accidental tampering, the pump shall not allow or shall require the user's confirmation to: • activate a basal profile while another one is active; • change an active basal profile; • change an active temporary basal; • change an active normal bolus; or • change an active extended bolus.	2.15, 8.10.1, 8.10.9
2.1.3	The pump shall provide protection measures, such as password protection, to assure that unauthorized personnel cannot tamper with data critical to insulin administration. Data critical to insulin administration is defined in requirement 1.9.2.	3.8, 9.6–7
2.2 User input		
2.2.1	If the pump is in a state in which user input is required, e.g., setting time and date, setting drug type, and concentration after reloading the drug reservoir, the pump shall issue periodic alerts/indications every *x* minutes until the required input is provided.	1.16, 3.15, 8.9.1
2.2.2	Clearing, changing or resetting the pump settings shall require the user's confirmation.	3.2, 3.17, 8.10.1
2.2.3	Setting and changing the concentration and activity duration of the currently loaded insulin shall require the user's confirmation.	8.9.1, 8.10.1
2.2.4	If the user has not interacted with the pump for *x* minutes while programming a basal profile, a temporary basal, or a normal/extended bolus, the pump shall signal a notification and discard all parameters the user has entered.	8.10.3–5
2.3 Keypad		
2.3.1	The pump shall generate a stuck key alarm whenever a key is held down for a minimum of *x* minutes.	4.3.4, 8.10.2
2.3.2	A key that is depressed shall not be identified as a distinct key press for less than *x* milliseconds.	4.3.3, 8.10.2
2.4 Information display		
2.4.1	The pump shall display sufficient information to the user during its normal operation to assist the user in monitoring pump operation. The information displayed shall include at least: the currently active basal profile, its latest update time and date, and the current basal rate (if applicable); the programmed rate and remaining time of any active temporary basal (if applicable); a visual indication that a normal bolus is in progress (if applicable); the rate and remaining time of an active extended bolus (if applicable); a visual indication of the remaining battery life; and current time and date programmed into the pump.	3.9, 8.10.3–5

Table 3.
Requirements on Alarm, Alert, Warning, and Reminder

Req. ID	Requirement Specification	Causes to Mitigate
3.1 Alarms		
3.1.1	The pump and its accessories shall be designed to maintain a failsafe state in the presence of a single fault condition that results in the inability of the pump to ensure the integrity of the pump's operation. When in a failsafe state, the pump shall neither deliver insulin nor generate energy or substances that could affect the user's safety.	2.14, 3.1.1–3, 3.3, 3.4, 3.10, 3.13, 4.2.3, 6.1–2, 8.8, 8.10.12, 9.3
3.1.2	An alarm condition shall be indicated through both auditory/tactile and visual signals.	2.15, 4.3.8–9
3.1.3	Alarms should clearly indicate the specific condition causing the alarm.	3.6
3.1.4	The pump shall allow the user to choose either audible or vibration mode for alarms. If the pump is in vibration mode and the user does not acknowledge an alarm for more than *x* minutes, the pump shall automatically transit to audible mode and signal an audible alarm.	4.3.9
3.1.5	The pump shall continue notifying the user every *x* minutes while an alarm remains unacknowledged and not overridden by alarms with higher priorities.	2.15, 4.3.5, 4.3.8–9, 8.8, 8.10.12, 9.3
3.1.6	Audible alarm signals shall be in the range of *x* dBA to *y* dBA.	4.3.5–6, 8.8, 8.10.12, 9.3
3.2 Alarm, warning, and reminder		
3.2.1	The pump shall signal audible reminders when no food bolus has been requested by the user within 2 hours after normal meal hours.	8.10.7
3.2.2	The pump shall remind the user to rotate infusion sites if it has been attached to the user at the same site for more than *x* days.	7.4
3.2.3	For a disposable insulin pump, it shall signal an expiration reminder no later than *x* hours before its normal use expires and shall keep signaling expiration reminders every *y* minutes until the user stops using the pump.	7.4
3.2.4	The pump shall advise the user to disconnect the infusion set from the patient prior to a prime process.	2.14
3.2.5	When the user inputs a BG reading, target BG level, insulin-to-carbohydrate ratio, or correction factor that is out of manufacture- or user-defined ranges, the pump shall generate a warning and require the user to confirm or change the input.	8.9.4–5
3.2.6	Any change of delivery modes in the pump shall be accompanied with auditory, visual, or tactile feedbacks.	8.10.4–5, 8.10.9
3.2.7	The pump shall issue a warning whenever there is a failure in event logging or log retrieving.	3.11
3.3 Safety checks		3.7, 3.8
3.3.1	The pump shall have a mechanism that checks the correctness and accuracy of the real-time clock (RTC) of the pump once every *x* minutes. Any problem detected in the check shall cause the pump to signal an RTC error alarm and stop the ongoing insulin administration.	4.6.1
3.3.2	Whenever data is loaded from the nonvolatile memory (e.g., ROM, EPROM, EEPROM, etc.) of the pump to its volatile memory (e.g., RAM, MRAM, FLASH memory, etc.), the integrity of the data shall be checked and ensured, i.e., the data loaded into the volatile memory shall be identical to that in the nonvolatile memory.	4.1.2
3.3.3	Whenever data is written from the volatile memory of the pump to its nonvolatile memory, the integrity of the data shall be checked and ensured, i.e., the data written into the nonvolatile memory shall be identical to that in the volatile memory.	4.1.3
3.3.4	A system failure alarm shall be issued if any of the safety checks fail.	4.1.1–3, 4.6.1
3.3.5	When a pump suspension command is issued, the pump mechanism shall be checked within *x* milliseconds to verify that the pump has stopped. If the pump has not stopped, power to the pump shall be interrupted via redundant circuitry and a system failure alarm shall be issued.	2.14

Continued →

Table 3. Continued

Req. ID	Requirement Specification	Causes to Mitigate
3.4 Power-on self-test		
3.4.1	Upon being powered on, the pump shall undergo a power-on self-test (POST), which should include tests as specified in 3.4.3.	2.16, 3.7, 3.8, 4.1.1–3, 4.3.1–2, 4.3.5–6, 4.3.8–9, 4.5.3, 4.6.1, 6.5
3.4.2	The system shall perform a POST for all subassemblies without degrading normal operation.	
3.4.3	The POST shall include at least the following tests: • CPU test • nonvolatile memory test • volatile memory test • battery test • keypad test (or other input device test) • display test • watchdog test • RTC test • speaker/vibrator test (if applicable)	
3.4.4	Any failure of a test step during POST shall abort the remaining test steps and generate the appropriate alarm for the failure, and transition to a known safe state.	
3.4.5	The pump shall wait in a known safe state during the POST process, i.e., the pump shall deliver no insulin, other substances, or energy during POST.	
3.4.6	Software shall be initialized to appropriate values.	3.15
3.5 Watchdog		
3.5.1	The pump shall have a watchdog, or equivalent safety mechanisms, which are capable of detecting unrecoverable software failures that prevent the pump from meeting its expected runtime performance.	2.15, 3.3, 3.4, 4.2, 4.5.1–2, 6.1–2
3.5.2	When unrecoverable software failures that prevent the pump from meeting its expected runtime performance are detected, the watchdog or equivalent safety mechanisms implemented in the pump shall trigger the pump to enter into a failsafe state (see the definition in requirement 3.1.1) within *x* seconds.	
Abbreviations list: (POST) power-on self-test, (RTC) real-time clock		

Table 4.
Event Logging

Req. ID	Requirement Specification
4.1	The pump shall maintain an electronic log to record each user event.
4.2	When the user overrides a suggested bolus, the pump shall maintain an electronic log to record the original dosage of the suggested bolus and the final dosage that the user selects.
4.3	The pump shall maintain an electronic log to record each fault condition, and the associated alarm and/or alert issued.
4.4	The pump shall maintain electronic records of the user's BG readings for the previous *x* days.
4.5	The pump shall maintain electronic records of the user's daily basal and bolus dosages for the previous *x* days.
4.6	The pump shall maintain electronic records of the last *x* boluses, administered completely or incompletely. Each bolus record shall at least include the administered dosage and duration of the bolus.
4.7	Each log entry shall be stamped with a corresponding date/time value.
4.8	Information logged shall be retained for at least *x* days.

Table 5.
Requirements on Battery Management

Req. ID	Requirement Specification	Causes to Mitigate
5.1 Battery voltage		
5.1.1	The pump shall be designed to use batteries as its only power source.	
5.1.2	The pump battery voltage shall be measured prior to each pump motor movement.	6.5.6
5.1.3	The amount of battery life remaining shall be calculated as a function of the active battery voltage.	6.5.1–3
5.1.4	The pump shall signal an empty battery alarm and stop delivery when the amount of estimated battery life remaining is less than *x* minutes. Note that *x* should be instantiated with an appropriate value, so that the pump can guarantee to stop any insulin administration and power off safely within *x* minutes.	6.5.1
5.1.5	The pump shall signal a low battery alert when the amount of estimated battery life remaining is less than *x* minutes. This alert shall occur periodically until the battery is replaced with a good battery. Note that *x* should be instantiated with an appropriate value, so that the user can respond to the low battery alert (e.g., replacing the battery) within *x* minutes.	6.5.2–3
5.1.6	The pump shall signal a bad battery alert and stop delivery if the amount of battery life remaining is unpredictable.	6.5.5
5.2 Battery and contact impedance		
5.2.1	The battery and contact impedance shall be measured prior to or during each pump motor movement.	6.5.7
5.2.2	The pump shall initiate a high battery/contact impedance alert when the measured impedance is greater than *x* Ω. This alert shall occur periodically until the contacts are cleaned or the battery is replaced with a good battery.	
5.3 Battery replacement		
5.3.1	When the battery is removed, a cyclic redundancy check (CRC) value shall be calculated for the pump settings in battery-backed memory. When the battery is replaced, a CRC value shall be recalculated and compared with the CRC calculated at battery removal. The pump shall notify the user and restore to default factory settings if the two CRC values do not match.	4.1.3
5.3.2	When the pump battery is replaced, the pump internal timer shall be checked against the pump real-time clock. The pump shall prompt the user to reset the date and time whenever the discrepancy between these two timers is greater than *x* minutes.	5.3.2
5.4 Auto-off and power-saving mode		
5.4.1	If the user has not interacted with the pump for *x* hours, the pump shall stop all basal and bolus administrations and signal audible alarms. Note that this feature can be either mandatory or user configurable.	8.2, 8.8, 8.10.12
5.4.2	The pump shall transition into power-saving mode if no user action has been detected within *x* minutes and no alarm is active. All basal/bolus administrations shall proceed as scheduled and shall not be affected by the transition.	6.5
5.4.3	The pump shall transition out of power-saving mode when a user event is detected, the time to deliver a basal or bolus dose arrives, or an alarm/alert/reminder condition occurs.	6.5
5.5 Patient leakage current		
5.5.1	If patient leakage current greater than *x* μA is detected, the pump shall issue an alarm.	6.4

Table 6.
Requirements on Interacting with External Environment

Req. ID	Requirement Specification	Causes to Mitigate
6.1 Operational conditions		
6.1.1	The pump shall be able to operate as intended within a temperature range of *x* °C to *y* °C.	2.2, 2.11, 2.14, 9.1
6.1.2	If the pump becomes overheated to more than *x* °C, the pump shall signal a pump overheated alarm.	2.11, 5.4
6.1.3	The pump should be able to withstand and operate as intended under atmospheric pressure ranging from *x* to *y* mm Hg.	2.11, 5.3, 9.2
6.1.4	The pump should be able to operate as intended at relative humidity ranging from *x*% to *y*% (noncondensing).	2.11, 5.2, 6.1–2
6.2 Electromagnetic compatibility		
6.2.1	The pump shall be able to operate as intended without alarm in the electromagnetic environments of intended use without causing interference in other equipment. The pump shall comply with CISPR 11 Group 1 Class B and/or FCC Class B emissions limits. The pump shall be immune to 25 kV air discharge (minimum) when tested according to IEC 61000-4-2. The pump shall be immune to 20 V/m radiated RF, minimum; amplitude modulated 80% at 1 kHz from 80 MHz to 2.5 GHz, when tested according to IEC 61000-4-3.	2.11, 6.2, 9.3