2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) | 979-8-3503-4479-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSTW60967.2024.00039

A Survey on the Verification and Validation of Artificial Pancreas Software Systems

1st Bara' Nazzal
School of Computing
Queen's University
Kingston, Canada
21bn7@queensu.ca

2nd Manar H. Alalfi
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada
manar.alalfi@torontomu.ca

3rd James R. Cordy
School of Computing
Queen's University
Kingston, Canada
cordy@queensu.ca

***Abstract*—In this paper, we present a comprehensive survey focusing on software safety and security verification and validation, particularly concerning medical devices, with a specific focus on artificial pancreas systems (APS) used in diabetes treatment. Medical device software represents one of the most critical cyber-physical applications, where failures can have immediate life-threatening consequences. Therefore, it is important to insure that these systems meet the specifications insuring their functionality, safety, and correctness. Our survey emphasizes the critical role of verification and validation procedures in achieving that. Furthermore, we explore the application of software analysis and modeling techniques in the certification, validation, and verification processes of the software employed in these systems.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The integration of software into medical devices offers numerous benefits but also introduces risks. Surveys of medical device recalls reveal that software faults frequently contribute to device failures [27], [60], underscoring the critical need for validation and verification of software components. Both the medical and technology communities exhibit significant interest in this area, since ensuring the safety and security of medical devices is paramount. While there are many technologies and techniques proposed to ensure security, safety, and correctness of programs, the extent to which they are utilized in real-life implementations of devices is yet to be determined.

In this survey, we chose to focus on artificial pancreas systems (APS) as an example. APS is a closed-loop system, where the controls system automatically responds to input and adjusts accordingly. APS comprises three main components: a continuous glucose monitor, controller software, and an insulin pump. Widely accessible and vital, APS represents a moderately complex life-critical multi-component medical device. The presence of active open-source projects for the controller software makes APS an ideal subject for research, particularly regarding the application of software verification and validation techniques and their potential use in the certification process.

Notable studies have reviewed security issues facing insulin pumps [49], highlighted the risks and benefits of insulin pumps, and advocated for safety standards [32], and explored challenges posed by such systems [29]. Industry standards for cyber-physical medical devices, including APS, have been developed by organizations such as the IEEE [33], FDA [23], the European Union [28], and NIST [15], [19].

APS solutions are rapidly evolving, with open-source do-it-yourself APS solutions, like OpenAPS [46], AndroidAPS [4], and Loop [39], gaining popularity. Evaluating the safety and security of these evolving solutions from a medical standpoint is of great interest. Recent studies have explored verification monitoring [48], in silico testing [58], and algorithm simulation efficiency [52].

Various techniques are employed for testing and validating insulin pumps, including static analysis tools like CodeSonar [35]. Formal methods and modeling techniques are also widely used, as shown by recent surveys [16], [40]. Formal techniques demonstrate potential for automatic, dependability-driven certification [26], as evidenced by their practical applicability in the development of medical devices. Additionally, formal languages like Tamarin [13], [45] and ProVerif [14] are utilized for verifying security protocols.

Our study aims to investigate the literature concerning the techniques used to analyze and ensure the safety and security of software utilized in APS, with a focus on model-based methods. Safety properties insure that no harm is done to the user, and security properties insure the confidentiality, integrity and availability of the software. Software modeling techniques provide a means to analyze and encompass software states, thereby establishing and maintaining safety and security properties [26]. In the subsequent sections, we will survey and analyze different techniques employed for analyzing such systems and identify research gaps in this critical area.

II. Background

A. Medical Device Software Analysis

Medical devices represent safety-critical cyber-physical systems, where gaps in security and safety could have severe implications for patient health and well-being. The current certification process for medical devices relies heavily on clinical-trial tests, which are susceptible to errors. This stands in contrast to other fields where automatic formal tests are commonplace. Additionally, the emergence of do-it-yourself solutions, such as controller programs in artificial pancreas

979-8-3503-4479-0/24/$31.00 ©2024 IEEE
DOI 10.1109/ICSTW60967.2024.00039

systems (APS), raises concerns as they may not undergo official certification. Addressing these challenges requires research into automated and error-resistant certification processes that are user-friendly and accessible.

Given that medical devices often comprise multiple components, the risk of vulnerabilities increases significantly. Attacks or errors could occur at any layer or within any component, necessitating rigorous testing procedures to ensure comprehensive coverage. Verification ensures the correct implementation of requirements, while validation confirms that the specified requirements are met. Testing methods, including open-box and black-box testing, offer distinct advantages and disadvantages. Open-box testing and static analysis provide extensive coverage but require access to internal functions and are prone to false positives. Conversely, dynamic and black-box testing may not offer full coverage. In safety-critical devices, achieving comprehensive test coverage is paramount. Formal methods, which involve the mathematical development and verification of software to ensure adherence to requirements, offer a promising avenue for improving testing processes in medical devices. Transitioning to automatic testing methodologies could be facilitated by leveraging formal methods in the development of medical device software.

B. Artificial Pancreas Systems (APS)

This survey focuses on one medical software system, an APS, which helps patients with diabetes to control Blood Glucose (BG) levels. Glucose is necessary for humans to function, but it has to be maintained in the body in a safe range, as both low blood sugar and high blood sugar would result in unwanted and potentially dangerous symptoms. The safe target might vary from person to person depending on their conditions, but a typical target according to the Centers for Disease Control and Prevention is as follows [24]: 80 to 130 milligrams of glucose to each decilitre of blood (mg/dL) and less than 180 mg/dL, two hours after a meal.

The amount of glucose in the body changes through the day and rises after meals. Blood glucose levels are regulated by insulin, which is a hormone produced by the pancreas. The body produces more insulin when the blood glucose rises after a meal, and the insulin lowers the glucose back to the safe value. In diabetes patients, the body does not regulate blood sugar as it should be. For example, this could happen if pancreas does not produce insulin, which would be classified as type 1 diabetes. A type 1 diabetes patient would require a medical device to provide the insulin. Such devices are known as APS.

An APS is comprised of three components: a blood glucose sensor, known as a continuous glucose monitor (CGM), a controller, and a insulin pump. The system could be an open loop or a closed loop. After detecting the blood sugar level, in an open loop the controller algorithm would tell the user to manually inject the insulin, while a closed loop does not require user input and the controller can automatically provide commands to the insulin pump to provide the appropriate amount of insulin. A closed loop system is critical, because the device has full automatic control, so it requires more rigorous verification to assure that it functions correctly and the design of the system should prioritize safety. For example, a system that has no limits on how much insulin it pumps is prone to over dosage.

There are many available APS, including do-it-yourself open-source solutions, such as OpenAPS [46] and AndroidAPS [4] which is derived from it, as well as Loop [39]. We can examine OpenAPS security claims and its basic functionality. The default functionality uses basal injection, this means that small dosages, also called boluses, of insulin are delivered through multiple injections through the day. To increase safety, OpenAPS relies on temporary basal commands instead of issuing insulin directly. This allows it to modify the basal rate as needed for a specific period of time. Per OpenAPS' design documents [47] and explained by Toffanin et al. [58], the the basic operations are as follows: The controller periodically checks data from the CGM and queries the pump for its settings and activity. Based on the input and user information; these include the current carbohydrate intake, carbohydrate-to-insulin ratio, current glycemia, target glycemia, the insulin sensitivity factor (ISF). It also calculates the amount of carbohydrates from previous meals, as well as the Insulin on Board (IOB) which is calculated.

The IOB represents the amount of insulin currently in the body. This could be from previous insulin boluses or from basal changes. This is calculated based on the time that passed after each bolus, the insulin peak time and the insulin's duration of action. Furthermore, if there has been no recent bolus, it uses the CGM reading to calculate the eventual blood glucose. This done using the ISF and IOB.

With these calculated, OpenAPS can operate in different ways, depending on the configuration. For example, a safety procedure is followed if the blood sugar is below threshold, where the controller enters glucose suspend mode; this withholds insulin infusion until blood glucose recovers. Otherwise, the basic operation of the algorithm, called oref0, alters temporary basal to correct the glycemia values. For example, if the Blood Glucose (BG) is rising but the calculated eventual BG is below target, or if BG is falling and the eventual BG is above target, the controller cancel temporary basals. Otherwise, if based on the calculations, the required temporary basal is more than the existing one, a new higher one is issued, or if it is lower, a new low one is issued.

Newer versions of the controller include more algorithms that are used for different profiles and scenarios. These algorithms include the Advanced Meal Assist (AMA) and SuperMicroBolus (SMB). AMA allows for a quicker adjustment to a higher temporary basal after a meal, while SMB, which is in the newer oref1 system, is not based on temporary basal rates, instead rapidly issues smaller boluses, which allows for quicker insulin delivery. This could also be adjusted for a full, half, or no meal bolus. Each one of these configurations would also require its own safety checks.

TABLE I
KEY WORDS USED

Device	Validation	Standards
Medical device	Security	FDA
Medical software	Verification	IEC
Artificial pancreas	Certification	ISO
Infusion pumps	Modeling	EU
Closed loop	Model checking	Standards
APS	Formal methods	Recalls
OpenAPS	Software analysis	
AndroidAPS	White box	
Loop	Reverse Engineering	
Insulin Delivery	Testing	

III. SURVEY AND PAPERS SELECTION

A. Papers Selection

For paper collection in our survey we used Web of Science and Google Scholar for our initial search. We supplemented our initial search by using digital libraries that cover computer science topics, such as IEEE Xplore, ACM, and Springer, and medical digital libraries such as The National Library of Medicine.

To search for papers related to our targeted area of application, artificial pancreas systems and medical devices in general, we used a combination of words related to the targeted devices, keywords relating to validation and testing, and words related to standards and regulations. The keywords are shown in Table I. The query strings combined the keywords in the device column with the validation and standards columns; for example *Artificial pancreas security*, *Artificial pancreas verification*, *Artificial pancreas FDA*.

We also consulted previous surveys in the relevant fields, such as John Majikes et al. [40] and Silvia Bonfanti [16], and expanded on them by including recent research, with a focus on on APS validation, verification and certification. We also considered a wider array of possible techniques, even if we focus on modeling and formal methods.

Papers that covered the relevant topic, the security and safety of artificial pancreas software and their testing and validation were included. Due to the nature of the topic, we have also included papers on other medical devices than APS if the technology is relevant and could be applied to APS as well. We have included research on modeling and validation techniques on Android, since APS controller software, such as AndroidAPS, use that platform.

On the other hand, we have excluded research that was not focused on the software, such as the research on the hardware or user interaction with the system. We have also excluded research on devices other than APS that did not use technologies that are relevant or could be generalized, for example research on the robotic and motion controls of surgical medical devices.

Web of Science results for the search term *Artificial Pancreas* returns 3,603 and sorting them by date, as shown in Figure 1, shows the existing and continued interest in these devices. The majority of the research is from pharmacology and medical research and when filtered to include only Web of Science categories related to computing, we get 286 results. However, when searching for *Artificial Pancreas Security*, we get a total of only 20 results. Additionally, when searching for *Insulin Delivery Security*, which is more broader term, we only get 33 results, the majority which are not computer-science focused. Searching for research related to artificial pancreas safety returns a higher count, with 439 results, and searching for artificial pancreas validation gives us 133 results. Many of these are medical papers or are not specifically on software validation.

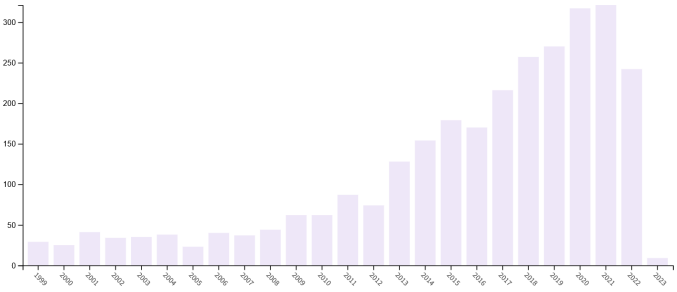

Fig. 1. World of Science Results for Artificial Pancreas

We commence our investigation by examining research specifically concerning the security of artificial pancreas systems, complemented by studies on insulin delivery systems. Out of the 20 papers reviewed, we identified 1 paper utilizing formal methods for validation, 5 focusing on anomaly detection, and one on code generation. Additionally, other papers cover topics such as encryption implementation, communication protocol technologies, and the utilization of fingerprints for access control. Furthermore, 4 papers explore non-technical aspects, including ethical, social, and user experience considerations, while 2 offer general overviews of APS. Lastly, 3 papers were found to be irrelevant to our study. Figure 2 illustrates these findings.

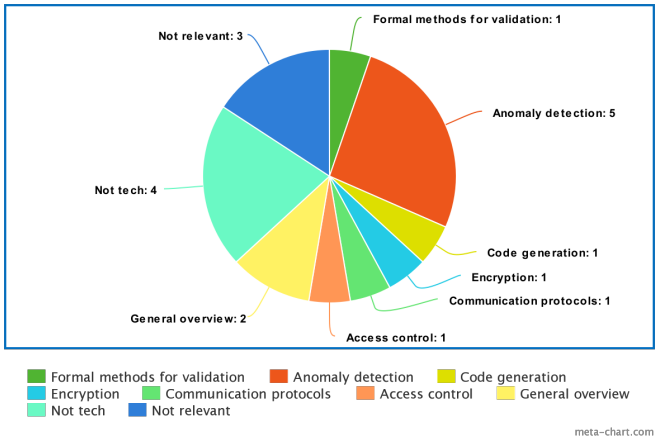

Fig. 2. Categorizing Web of Science Results for Artificial Pancreas Security

B. Papers Categorization

In the following Tables II, III, we have categorized the core related literature. We categorized them first based on whether

TABLE II
A CATEGORIZATION OF PREVIOUS LITERATURE

Group	Development/Analysis	Level(Use case/Subject)	Focus	Goal Group	Target Device	Target Properties
Alur [3]	Development	Requirements and design	Modeling	Correctness	CARA control system	WRAIR Documents
Arney [7]	Development	Abstract System (Generic PCA)	Modeling	Safety	Infusion Pump	Author defined (Provided as examples)
Banerjee [11]	Development	Abstract System	Modeling	Correctness	General (Examples: infusion pump control algorithm, chemotherapeutic Infusion)	ISO 60601 (mentioned)
Babamir [10]	Development	Abstract System	Modeling, Verfication	Safety	General (Examples: Infusion Pump)	Author defined (based on their model's state combinations)
Neeraj Singh [54]	Development	Requirements and design	Modeling, Verfication	Safety	Modular system	Author defined
Silva [53]	Development	Clinical Scenarios	Modeling	Safety, Functionality	Modular system (Medical scenarios)	Author defined
Xin Chen [17]	Development	Abstract System (Multi-basal control system)	Modeling	Correctness, Safety	Controller (Insulin Infusion)	Author defined (BGL not too low, not too high, within euglycemic range)
Thiagarajan [57]	Development	Abstract System (Open PCA Pump)	Modeling	Safety (Risk Management)	Infusion Pump (PCA Pump)	ISO 14971
Mauro [44]	Development	Code Generation	Code Generation	Safety, Functionality	User interface	N/A
Masci [42]	Development	Code Generation (GPCA UI)	Code Generation	Safety, Functionality	User interface (GPCA pump)	GPCA safety requirements
Alshalalfah [2]	Existing System	Concrete System (PID, MB Algorithms)	Security Analysis	Security, Safety	Controller (APS)	Author defined (Attack Scenarios and Two Safety Properties)
Vega-Hernandez [59]	Existing System	Abstract System	Fault Detection	Security	APS	Author defined (Sub- and over-dosing scenarios)
Aliabadi [1]	Existing System	Analysis Tool	Specification Mining	Security	General (Examples: smart meters,smart meters, smart artificial pancreas and unmanned aerial vehicles)	Author defined (Attack models based on priveous literature)
Aliabadi [51]	Existing System	Analysis Tool	Intrusion Detection	Security	General (Examples: smart meters,smart meters, smart artificial pancreas)	Specification mining (by ARTINALI++)
Astillo [9]	Existing System	Analysis Tool	Misbehavior Detection	Security	APS (IMD Enabled)	Specification derivation (by the authors)
Astillo [8]	Existing System	Analysis Tool	Misbehavior Detection	Security	APS (IMD Enabled)	Specification derivation (by the authors)
Paolo [43]	Existing System	Source code (Reverse Engineering)	Fault Detection	Safety	User interface	Author defined (Based on previous literature)
Harrison [30]	Existing System	Concrete System (NIDUS)	Modeling, Verfication	Safety	Dialysis Machine	Author defined (Risk log)
Harrison [31]	Existing System	Concrete System (NIDUS)	Modeling, Verfication	Correctness	Dialysis Machine	Author defined (Risk log)

TABLE III
A CATEGORIZATION OF PREVIOUS LITERATURE (CONT.)

Group	Formalism	Analyses Tools
Alur [3]	Timed Automata	SCR, Hermes, CHARON
Arney [7]	Z notation, Timed Automata	Z Tools, UPPAAL
Banerjee [11]	Hybrid automaton (CPS-MAS)	Own framework (CPS-MAS)
Babamir [10]	Petri nets	Z language, Petri Nets, C#
Neeraj Singh [54]	Set theory	Event-B Language
Silva [53]	Actor-Oriented Design	Ptolmey II
Xin Chen [17]	Dalla-man model	Dalla-man model, Flow* tool
Thiagarajan [57]	Architecture description	AADL
Mauro [44]	PVS (typed higher-order logic)	PVSio-web, Own tool (MISRA C)
Masci [42]	PVS (typed higher-order logic)	PVS
Alshalalfah [2]	Timed Automata	UPPAAL-SMC
Vega-Hernandez [59]	Mathematical	Abstract
Aliabadi [1]	Own approach (dynamic specification mining)	Own Tool
Aliabadi [51]	Bayesian statistics	Own Tool
Astillo [9]	Specification-based /Proximal Monitoring	UPPAAL, MATLAB
Astillo [8]	Specification-based	UPPAAL
Paolo [43]	PVS (typed higher-order logic)	PVS
Harrison [30]	MAL, CTL	IVY, NuSMV
Harrison [31]	MAL, CTL	IVY, NuSMV

the research is driven toward helping the development and design of future devices or was it mainly focused towards the analysis of existing systems. This is followed by elaborating on the type of system the research dealt with. This is mainly on whether the research was giving a model for an actual concrete system, that is in use, or is subject to a more abstract theoretical system; for example a study that analyzed Newcastle infant dialysis and ultra-filtration system (NIDUS), which is an actual system that is in use, we also considered the study of algorithms that are used in implemented systems

as concrete systems, while research that used general designs or concepts from The Generic Infusion Pump, as apposed to studying an implementation of it, is labeled an abstract system. In this categorization, we also have research that was more focused on code generation or providing an analysis tool.

We also categorized the papers according the their methodological focus, that is, whether the research is mainly focused on modeling and verification, code generation, or fault or intrusion detection. While there is an overlap between the categories, most of the papers involve both modeling and verification steps, we considered it the main focus for the label depending on the contribution of the paper. We also categorized the research based on the achievable goal of the experiment; whether it was to assess the correctness and functionality of the model, or is it to prove the safety or security of the system. If the focus is the software functionality and adhering to specification, it is labeled with the correctness goal. If the focus is on insuring that no harm is done to the user, it is labeled with the safety goal. If the focus is the confidentiality, integrity and availability of the software, it is labeled with the security goal.

We also show the targeted medical device in each case and the targeted properties that were tested or used for verification; for example if the researchers depended on an already established standard, specific documentation, or properties picked and defined by the authors. Finally, we provided the main formalism that was used in the research's methodology and the main tools that were used in the approach.

C. Papers Analysis and Discussions

1) APS Security: The most relevant paper in this search is by Alshlalfah et al. [2], where they used formal methods, specifically, time formal automata for the security analysis of the Multi-Basal and Proportional-Integrative-Differential controller algorithms used in an artificial pancreas. Their approach used UPPAAL-SMC to model the algorithms, a physiological system, and an adversary for attack scenarios, mainly an attack where the adversary monitors the insulin control channel and intercepts the messages. Their glucose model is based on the UVA/PADOVA model [41]. While the paper provides an interesting contribution to APS systems verification, it does not focus on the software side as much. It also does not consider do-it-yourself systems such as OpenAPS and its Android implementation, AndroidAPS. A finer grain modeling of the software and a more thorough analysis for its operation and security beyond message interception is still an open area for study.

Interestingly, a recent survey by Johan Arcile and Étienne André [6], notes that in their survey they found very few applications of timed-automata in medical systems, with the previous paper being the main representative. They speculated that this could either be to a lack of interest in formal verification, or that the systems are too expressive or too large for timed-automata modeling. In our survey, as well as previous ones by John Majikes et al. [40] and Silvia Bonfanti [16], we show that while there might not be many papers using timed-automata specifically, this is definitely not due to a lack of interest in formal verification. Nonetheless, this shows that there is a gap in the research when it comes to using formal verification on real implementations in the medical field.

The earliest paper included from this search is from 2009 by Vega-Hernandez et al. [59], where they proposed a mathematical-based model for fault detection in artificial pancreas systems. Their approach is more abstract and focuses on the development of the fault detection algorithm. Other papers focused on fault detection are the work of Aliabadi et al. [1] [51], where they propose ARTINALI++ based on dynamic specification and ARTINALI# based on Bayesian search and scoring. While in this survey, intrusion detection and its techniques is not the focus per se, their research is relevant because they do consider OpenAPS and test their approach on it. They also provide a good specification of the possible attacks on the system, which could be used in our work. The work of Astillo et al. [9] and [8], where they introduced SMDAps and TrMAps is similar. Their approach to misbehavior detection is through proximal monitoring. Their proposed tool is designed with UPPAAL and it can work alongside different APSs, and in the paper, they test it on OpenAPS. This shows a different usage of UPPAAL; rather than modeling the system and analysing it, it is used to design the monitoring tool.

The other results are less relevant software code, but they tackle the security of APS from different prespective. This includes the works of Booch et al [61] which tackles the security for APS with a focus on encryption, Duguma et al. [20] and Strydis et al. [56] focus on communication protocols, and Zheng et al. [62] focuses on access control. Lazaro et al. [37] and Fatehi et al. [21] provide general overview of diabetes care devices. The rest of the results are mostly non-tech related, mainly medical research.

2) Formal Methods in Medical Software: Since the results on artificial pancreas security were relatively low, we also looked into the research on other medical devices, for example infusion pumps, which are more broad than artificial pancreas; they could deliver different fluids, other than insulin, to the body. They also may lack the software automatic-controller component, which we want to focus on. Nonetheless, they are close in functionality and could provide some insight in the technologies used. We searched for software analysis, formal methods usage, verification, and validation in medical devices in general.

After excluding non-relevant papers, we ended up with the following papers that utilize formal and modeling approaches in the area of verification and validation of APS and related medical devices such as patient control pumps and infusion pumps. Rajeev Alur et al. [3] early on have demonstrated the possibility of converting the informal design requirements of an infusion pump to formal systems such as extended finite state machines, using the reference specifications to enable formal analysis with the SCR, CHARON, and Hermes tools. Their approach was used to study a computer-assisted resuscitation algorithm.

David Arney et al. [7] have developed a safety model for a generic patient-controlled infusion pump and used formal methods for verification with a focus on hazards that could happen in the system. They captured safety and functional requirements from the reference manual and built a finite state automata for the system. In their tests, they checked for non-determinism, completeness, deadlocks, and safety properties. The main properties that were specified were mainly to to detect potential hazards such as output side occlusion, air bubbles forming, misuse, and over dosage. Based on that, the system was modeled with a focus on time intervals between boluses and detecting such abnormalities and acting based on them. Some of the challenges they identified in this approach is the need for tools that could handle the large number of states. They mentioned in future work the possibility of comparing the model to real implementation.

Ayan Banerjee et al. [11] also attempted the development of a modeling based approach for the safety verification of medical cyber-physical systems, including infusion pumps.

Seyed Morteza Babamir and Mehdi Borhani [10] used Petri net modeling and the Z language for specifications, then verified their security based on combined behavior through reachability graphs. They derived properties that were related to the model's reachability as well as properties provided by physicians. The first group included the following properties: blood sugar must not get low; if blood sugar is normal, the dose should not exceed the minimum; over-dosage should not occur; there should be insulin available in the system when a dose is to be delivered; a dose should be delivered if the blood sugar is low. For the second group, the properties are: cumulative dose should not exceed a permissible daily dosage, and in case of the it exceeds the daily dose, it should be less than the difference between the maximum cumulative dose and current cumulative dose. This approach is focused on the properties that should be monitored by system to assure safe functioning and is informed by both the model and expert requirements.

Neeraj Kumar Singh et al. [54] have used an approach based on the Event-B language. In their future work, they propose the possibility of generating code from the models and developing closed loop systems using them. In their work they defined 14 requirements and 9 safety properties defined based on that. Their paper provided an elaborate description of the system's informal requirements that they used to derive their formal properties in event B language. They included 14 requirements for the system. The detailed requirements are provided in Table IV. The requirements are relevant in testing the controller application and can be used as a base for future tests.

Lenardo C Silva. [53] proposed a model-based validation approach. They built reusable libraries with the use of the Ptolmey II modeling tool for the patient and devices which could be combined to simulate the system's overall behaviour. This approach is more adaptive and overcomes previous limitations of specific purpose models. They tested for different clinical scenarios that included the patient and device models,

TABLE IV
REQUIREMENTS USED BY SINGH ET AL. [54]

REQ1	The device must suspend all active basal delivery or bolus deliver during pump refilling and in the case of system failure.
REQ2	The device must undergo a power-on-self-test (POST) whenever device power is turned on.
REQ3	The device shall allow the user to manage system functionalities related to: stopping insulin delivery; validating basal profiles parameters; reminder management; and validating bolus preset parameters.
REQ4	The device shall allow the user to define a basal profile that consists of an ordered set of basal rates, ordered over a 24 hour day, as well as a temporary basal, that consists of a basal rate for a specified duration of time within a 24 hour day.
REQ5	The device can contain several basal profiles, but only one basal profile can be active at any single point in time.
REQ6	The device must allow the user to override an active basal profile with a temporary basal, without changing the existing basal profile.
REQ7	The device shall resume the active basal profile after the temporary basal terminates.
REQ8	The device shall enforce a maximum dosage for the normal bolus or extended bolus.
REQ9	The user shall be able to stop the active normal or extended bolus.
REQ10	The device must maintain an electronic log of every operation associated with an user alert, such as an audio alarm.
REQ11	The device shall maintain a history of basal and bolus dosages over the past n days. The n always differs among brands, though most store up to 90 days of data.
REQ12	The device shall enable the user to create a food database that can be used to store food or meal descriptions and the carbs associated with them.
REQ13	The device shall allow to the user to change parameter setting basal profile, bolus preset, and temporary basal.
REQ14	The device shall provide feedback to the user regarding system and deliverystatu

as well as verifying basic safety properties. For this purpose the requirements checked were: If a cartridge level is 0, then its status should become *empty*; if a cartridge level is lower than the dosage, then the pump status should be *stop*; if a specific profile is selected, then the administered dosage should be equal to the programmed dosage. These properties are less generic and are closer to the modeling approach used by the authors and reflects the components and their behavior in the system.

Xin Chen et al. [17] have used Dalla-Man models for the patient and a hybrid automaton model of the controller, and verified the correctness properties of the models. They focused on the control algorithm and checking that the blood glucose levels remained within the safe range overnight. The main properties checked for that include: that blood glucose levels should not get below 70 mg/dl; that the blood glucose level should not rise above 300 mg/dl; and that it should be in the range of 70 and 180 mg/dl when a patient wakes up. These properties are general safety and correctness properties that could apply to any APS system and are related to the functionality and results required from the system.

Hariharan Thiagarajan et al. [57] used Architecture, Analysis, and Definition Language (AADL) and error modeling for a patient controlled pump. Taisa Kushner et al. [36] used a data-driven modeling approach combined with reachability analysis of the controller software.

The Abhinandan et al. [48] strategy is to prevent attacks at run-time. To do that they introduce a verification monitor to the system. By monitoring values coming from the blood glucose levels and electrocardiograms of heart rate, they could check for abnormal patterns that could signal an attack, such as a gaining access attack, or an attack on the integrity or availability of the system. They used a timed automata for the models and formal policy definitions. Their policies are based on a system that provides electrocardiogram and heart rate, which is not always available with APS systems and thus could not be used if not available.

These efforts demonstrate that there is ongoing interest in model-based techniques and a wide application for them in the field of validation and verification of medical devices. We also observe that the problem of validating APS systems can be complex, and involves many components. Most of this previous research has focused on developing generic infusion pump or controller models and analysing them, rather than analyzing actual software. This shows that there is room for research on utilizing these techniques on existing practical devices and software, and a potential for proposing a more holistic practical approach.

We can also see that the requirements being tested could vary from being generic safety requirement to more specific properties that reflects the model's behavior. One thing to note is that previous literature that focused on developing systems from the formal methods, instead of analyzing existing systems gives more flexibility in the modeling process and the models could be written to include the functionalities required in the properties. For example, a property checking if the pump is refilling will require a model where the pump sends that status. This could be included in the design if we are to develop the system from the model, but for our purpose, when we are using formal methods on software or a device that already exists, we will have more constraints in what functionalities are available.

3) Medical Software Certification: We also looked into terms related to certification of APS and other devices, and the use of modeling techniques as a possible alternative or a supplement to trial-driven tests. Paolo Arcaini et al. [5] used such an approach on hemodialysis machines. They used the IEC 62304 [25] standard and FDA's General Principles of Software Validation [22] for the standards requirements, and used abstract state machines and model refinement to perform an interactive and simulation validation. Jing Liu et al. [38] studied airborne systems, but their contributions can be applied to infusion pumps as well. Their approach attempted to verify component designs with high level requirements and offers an automated tool chain for the process of formal verification. They used AADL and Simulink for modeling, and based their verification on the AGREE model checker. Paul Curzon etl. [18] focused on human-computer interaction and provided model-based assurance and certification tools. This included formalizing safety requirements based on FDA guidelines, developing, and verifying a user interface for a generic infusion pump. They also looked into the possibility of verifying existing implementations that were implemented in C++.

Raoul Jetley [34] et al. also looked into validating medical devices based on usage models and test cases. Their approach consists of requirements elicitation, usage model specification, model checking, and automated test case generation. They focused on the modeling the system through finite states and the properties they checked for were related to the state model. This included requirements such as checking the reachability in the model's states, that all valid events are executable, that it lack deadlocks, and that malfunctioning events trigger an alarm. This approach is focused on the state model and its mathematical correctness. Models are encoded in Simulink Stateflow, and the model's correctness is tested for all the possible state combinations.

Ayan Banerjee et al. [12] looked into the use of formal methods in medical device safety using a hybrid-automata framework, and showed that this approach can be applied to characterize and analyze the design of closed-loop devices. In their future work they noted that their approach may be applied to more complex models and exploring more verification techniques. Neeraj Kumar Singh et al. [55] looked into the glucose homeostasis mechanism and the use of formal methods as an alternative to testing and simulation, which they sh may offer better coverage. They used the Event-B modeling language in their approach.

We have also looked into other material relevant to source code analysis and verification, such as code generation from verified models, and Android application testing and modeling, since some open source APS systems use Android. Gioacchino Mauro et al. [44] for example studied the extension of user interface tools through C code generation from formal models, and Paolo Masci et al. [42] studied the development of an infusion pump user interface from finite state machines with formalized safety requirements. Their code generation was limited to Lisp code, but they note that code generators for C and Java are being developed. Their approach was based on Generic Patient Controlled Analgesic requirements provided in the documentation by the Generic Infusion Pump workbench [50]. Some properties included: the flow rate of the pump should be programmable; the pump should be able to deliver a dosage between a minimum and maximum value per hour; the user should be able to set the infused volume for volumes under and above provided variables. This approach reflects the focus on the system interface and the interaction between the user and the system.

IV. Summary and Key Findings

Our analysis of the surveyed papers demonstrate a wide array of approaches that utilize modeling and formal methods to medical devices. This is because formal methods can be used in the process of development and design of systems at different stages of the development process, as well as utilizing them in analyzing systems and verifying their properties. As shown in Table II, while there are multiple papers that use formal methods to analyze existing systems, they were

varied; one focuses on abstract systems, others propose tools for specification mining and anomaly detection, and some utilizing risk logs for insuring the safety and correctness of the system. We only encountered one that focuses on the software source-code, Paolo [43], and that paper was focused mainly on reverse engineering the user interface of the system. We see that there is a gap in the research in the application of formal modeling and analyses on the software level of concrete systems. Specifically, there is a lack of research that focus on the validation and verification of controller software in artificial pancreas systems (APS) on the software level. We can see that there is further potential for using model-based approaches and room for further research in the field, and in particular an opportunity to model, analyze and verify open source APS software directly from source code using model reverse engineering as pertain to model extraction and analysis.

Upon reviewing the existing literature on the modeling and verification of APS and medical devices in general, it appears that there is an active interest in this area of research. However, the research is not exhaustive; none of the papers surveyed provided verification and an in-depth analysis for the currently available open-source algorithms and their implementations in do-it-yourself APS. This can be explained by the fact that these applications are relatively recent and have a high degree of complexity. But this also shows that there is a room for contribution in the field; especially in the application of formal modeling methods on already implemented software.

REFERENCES

[1] Maryam Raiyat Aliabadi, Mojtaba Vahidi Asl, and Ramak Ghavamizadeh. Artinali++: Multi-dimensional specification mining for complex cyber-physical system security. *Journal of Systems and Software*, 180:111016, 2021.

[2] Abdel-Latif Alshalalfah, Ghaith Bany Hamad, and Otmane Ait Mohamed. Towards system level security analysis of artificial pancreas via uppaal-smc. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2019.

[3] Rajeev Alur, David Arney, Elsa L Gunter, Insup Lee, Jaime Lee, Wonhong Nam, Frederick Pearce, Steve Van Albert, and Jiaxiang Zhou. Formal specifications and analysis of the computer-assisted resuscitation algorithm (cara) infusion pump control system. *International Journal on Software Tools for Technology Transfer*, 5(4):308–319, 2004.

[4] AndroidAPS. Androidaps. androidaps.readthedocs.io/. Accessed: 2022-06-10.

[5] Paolo Arcaini, Silvia Bonfanti, Angelo Gargantini, and Elvinia Riccobene. How to assure correctness and safety of medical software: the hemodialysis machine case study. In *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, pages 344–359. Springer, 2016.

[6] Johan Arcile and Étienne André. Timed automata as a formalism for expressing security: A survey on theory and practice. *ACM Comput. Surv.*, 55(6), dec 2022.

[7] David Arney, Raoul Jetley, Paul Jones, Insup Lee, and Oleg Sokolsky. Formal methods based development of a pca infusion pump reference model: Generic infusion pump (gip) project. In *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*, pages 23–33. IEEE, 2007.

[8] Philip Virgil Astillo, Gaurav Choudhary, Daniel Gerbi Duguma, Jiyoon Kim, and Ilsun You. Trmaps: Trust management in specification-based misbehavior detection system for imd-enabled artificial pancreas system. *IEEE Journal of Biomedical and Health Informatics*, 25(10):3763–3775, 2021.

[9] Philip Virgil Astillo, Jaemin Jeong, Wei-Che Chien, Bonam Kim, Joung-Soon Jang, and Ilsun You. Smdaps: A specification-based misbehavior detection system for implantable devices in artificial pancreas system. *Journal of Internet Technology*, 22(1):1–11, 2021.

[10] Seyed Morteza Babamir and Mehdi Borhani. Formal verification of medical monitoring software using z language: a representative sample. *Journal of medical systems*, 36(4):2633–2648, 2012.

[11] Ayan Banerjee, Sandeep KS Gupta, Georgios Fainekos, and Georgios Varsamopoulos. Towards modeling and analysis of cyber-physical medical systems. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, pages 1–5, 2011.

[12] Ayan Banerjee, Yi Zhang, Paul Jones, and Sandeep Gupta. Using formal methods to improve home-use medical device safety. *Biomedical instrumentation & technology*, 47(s1):43–48, 2013.

[13] David Basin, Cas Cremers, Jannik Dreier, and Ralf Sasse. Symbolically analyzing security protocols using tamarin. *ACM SIGLOG News*, 4(4):19–30, nov 2017.

[14] Bruno Blanchet et al. An efficient cryptographic protocol verifier based on prolog rules. In *csfw*, volume 1, pages 82–96. Citeseer, 2001.

[15] Katie Boeckl, Katie Boeckl, Michael Fagan, William Fisher, Naomi Lefkovitz, Katerina N Megas, Ellen Nadeau, Danna Gabel O'Rourke, Ben Piccarreta, and Karen Scarfone. *Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks*. US Department of Commerce, National Institute of Standards and Technology . . . , 2019.

[16] Silvia Bonfanti, Angelo Gargantini, and Atif Mashkoor. A systematic literature review of the use of formal methods in medical software systems. *Journal of Software: Evolution and Process*, 30(5):e1943, 2018.

[17] Xin Chen, Souradeep Dutta, and Sriram Sankaranarayanan. Formal verification of a multi-basal insulin infusion control model. In Goran Frehse and Matthias Althoff, editors, *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, pages 75–91. EasyChair, 2017.

[18] Paul Curzon, Paolo Masci, Patrick Oladimeji, Rimvydas Rukšenas, Harold Thimbleby, and Enrico D'Urso. Human–computer interaction and the formal certification and assurance of medical devices: The chi+ med project. In *2nd Workshop on verification and assurance (Verisure2014), in association with computer-aided verification (CAV), part of the Vienna summer of logic*, 2014.

[19] Thomas P. Dover. Evaluating medical iot (miot) device security using nistir-8228 expectations, 2021.

[20] Daniel Gerbi Duguma, Philip Virgil Astillo, Yonas Engida Gebremariam, Bonam Kim, and Ilsun You. Comparative analysis of bluetooth le and edhoc for potential security protocol in artificial pancreas system. In Ilsun You, Hwankuk Kim, Taek-Young Youn, Francesco Palmieri, and Igor Kotenko, editors, *Mobile Internet Security*, pages 30–43, Singapore, 2022. Springer Nature Singapore.

[21] Farhad Fatehi, Anish Menon, and Dominique Bird. Diabetes care in the digital era: a synoptic overview. *Current diabetes reports*, 18(7):1–8, 2018.

[22] U.S. Food and Drug Administration. General principles of software validation — guidance for industry and fda staff. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-principles-software-validation. Accessed: 2022-06-10.

[23] U.S. Food and Drug Administration. Technical considerations for medical devices with physiologic closed-loop control technology. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/technical-considerations-medical-devices-physiologic-closed-loop-control-technology. Accessed: 2022-06-10.

[24] Centers for Disease Control and Prevention. Manage blood sugar. https://www.cdc.gov/diabetes/managing/manage-blood-sugar.html. Accessed: 2022-11-01.

[25] International Organization for Standardization. Iec 62304:2006 medical device software — software life cycle processes. https://www.iso.org/standard/38421.html. Accessed: 2022-06-10.

[26] Leo Freitas, William E. Scott, and Patrick Degenaar. Medicine-by-wire: Practical considerations on formal techniques for dependable medical systems. *Science of Computer Programming*, 200:102545, 2020.

[27] Zhicheng Fu, Chunhui Guo, Zhenyu Zhang, Shangping Ren, Yu Jiang, and Lui Sha. Study of software-related causes in the fda medical device recalls. In *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 60–69, 2017.

[28] Tuomas Granlund, Juha Vedenpää, Vlad Stirbu, and Tommi Mikkonen. On medical device cybersecurity compliance in eu. In *2021 IEEE/ACM 3rd International Workshop on Software Engineering for Healthcare (SEH)*, pages 20–23, 2021.

[29] Andy Harper. Exploring the challenges and designing potential solutions for insulin pump technologies, 2020.

[30] M. D. Harrison, M. Drinnan, J. C. Campos, P. Masci, L. Freitas, C. di Maria, and M. Whitaker. Safety analysis of software components of a dialysis machine using model checking. In José Proença and Markus Lumpe, editors, *Formal Aspects of Component Software*, pages 137–154, Cham, 2017. Springer International Publishing.

[31] Michael D. Harrison, Leo Freitas, Michael Drinnan, José C. Campos, Paolo Masci, Costanzo di Maria, and Michael Whitaker. Formal techniques in the safety analysis of software components of a new dialysis machine. *Science of Computer Programming*, 175:17–34, 2019.

[32] Lutz Heinemann, G Alexander Fleming, John R Petrie, Reinhard W Holl, Richard M Bergenstal, and Anne L Peters. Insulin pump risks and benefits: a clinical appraisal of pump safety standards, adverse event reporting, and research needs: a joint statement of the european association for the study of diabetes and the american diabetes association diabetes technology working group. *Diabetes care*, 38(4):716–722, 2015.

[33] IEEE. Ieee draft standard for wireless diabetes device security–information security requirements for connected diabetes solutions. *IEEE P2621.2/D2.0, November 2021*, pages 1–27, 2021.

[34] Raoul Jetley, S. Purushothaman Iyer, Paul L. Jones, and William Spees. A formal approach to pre-market review for medical device software. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 1, pages 169–177, 2006.

[35] Raoul Praful Jetley, Paul L. Jones, and Paul Anderson. Static analysis of medical device software using codesonar. In *Proceedings of the 2008 Workshop on Static Analysis*, SAW '08, page 22–29, New York, NY, USA, 2008. Association for Computing Machinery.

[36] Taisa Kushner, David Bortz, David M Maahs, and Sriram Sankaranarayanan. A data-driven approach to artificial pancreas verification and synthesis. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 242–252. IEEE, 2018.

[37] Caterina Lazaro, Erdal Oruklu, and Ali Cinar. Security challenges and solutions for closed-loop artificial pancreas systems. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1097–1100, 2017.

[38] Jing Liu, John D Backes, Darren Cofer, and Andrew Gacek. From design contracts to component requirements verification. In *NASA Formal Methods Symposium*, pages 373–387. Springer, 2016.

[39] Loop. Loop. https://loopkit.github.io/loopdocs/. Accessed: 2022-06-10.

[40] John J Majikes, Rahul Pandita, and Tao Xie. Literature review of testing techniques for medical device software. In *Proceedings of the 4th Medical Cyber-Physical Systems Workshop (MCPS'13), Philadelphia, USA*, 2013.

[41] Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. The uva/padova type 1 diabetes simulator: new features. *Journal of diabetes science and technology*, 8(1):26–34, 2014.

[42] Paolo Masci, Anaheed Ayoub, Paul Curzon, Insup Lee, Oleg Sokolsky, and Harold Thimbleby. Model-based development of the generic pca infusion pump user interface prototype in pvs. In *International Conference on Computer Safety, Reliability, and Security*, pages 228–240. Springer, 2013.

[43] Paolo Masci, Patrick Oladimeji, Paul Curzon, and Harold Thimbleby. Using pvsio-web to demonstrate software issues in medical user interfaces. In Michaela Huhn and Laurie Williams, editors, *Software Engineering in Health Care*, pages 214–221, Cham, 2017. Springer International Publishing.

[44] Gioacchino Mauro, Harold Thimbleby, Andrea Domenici, and Cinzia Bernardeschi. Extending a user interface prototyping tool with automatic MISRA c code generation. *Electronic Proceedings in Theoretical Computer Science*, 240:53–66, jan 2017.

[45] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 696–701, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[46] OpenAPS. Openaps. https://openaps.org/. Accessed: 2022-06-10.

[47] OpenAPS. Openaps reference design. https://openaps.org/reference-design/. Accessed: 2022-06-10.

[48] Abhinandan panda, Srinivas Pinisetty, and Partha Roop. A secure insulin infusion system using verification monitors. In *Proceedings of the 19th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '21, page 56–65, New York, NY, USA, 2021. Association for Computing Machinery.

[49] Nathanael Paul, Tadayoshi Kohno, and David C Klonoff. A review of the security of insulin pump infusion systems. *Journal of diabetes science and technology*, 5(6):1557–1562, 2011.

[50] Generic Infusion Pump. Generic infusion pump. https://rtg.cis.upenn.edu/gip/. Accessed: 2022-06-10.

[51] Maryam Raiyat Aliabadi, Margo Seltzer, Mojtaba Vahidi Asl, and Ramak Ghavamizadeh. Artinali#: An efficient intrusion detection technique for resource-constrained cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 33:100430, 2021.

[52] Jana Schmitzer, Carolin Strobel, Ronald Blechschmidt, Adrian Tappe, and Heiko Peuscher. Efficient closed loop simulation of do-it-yourself artificial pancreas systems. *Journal of Diabetes Science and Technology*, 16(1):61–69, 2022.

[53] Lenardo C Silva, Hyggo O Almeida, Angelo Perkusich, and Mirko Perkusich. A model-based approach to support validation of medical cyber-physical systems. *Sensors*, 15(11):27625–27670, 2015.

[54] Neeraj Kumar Singh, Hao Wang, Mark Lawford, Thomas S. E. Maibaum, and Alan Wassyng. Stepwise formal modelling and reasoning of insulin infusion pump requirements. In Vincent G. Duffy, editor, *Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management: Ergonomics and Health*, pages 387–398, Cham, 2015. Springer International Publishing.

[55] Neeraj Kumar Singh, Hao Wang, Mark Lawford, Thomas SE Maibaum, and Alan Wassyng. Formalizing the glucose homeostasis mechanism. In *International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management*, pages 460–471. Springer, 2014.

[56] Christos Strydis, Robert M. Seepers, Pedro Peris-Lopez, Dimitrios Siskos, and Ioannis Sourdis. A system architecture, processor, and communication protocol for secure implants. *ACM Trans. Archit. Code Optim.*, 10(4), dec 2013.

[57] Hariharan Thiagarajan, Brian Larson, John Hatcliff, and Yi Zhang. Model-based risk analysis for an open-source pca pump using aadl error modeling. In *International Symposium on Model-Based Safety and Assessment*, pages 34–50. Springer, 2020.

[58] Chiara Toffanin, Milos Kozak, Zdenek Sumnik, Claudio Cobelli, and Lenka Petruzelkova. In silico trials of an open-source android-based artificial pancreas: a new paradigm to test safety and efficacy of do-it-yourself systems. *Diabetes technology & therapeutics*, 22(2):112–120, 2020.

[59] O. Vega-Hernandez, F. Campos-Cornejo, D. U. Campos-Delgado, and D. R. Espinoza-Trejo. Increasing security in an artificial pancreas: diagnosis of actuator faults. In *2009 Pan American Health Care Exchanges*, pages 137–142, 2009.

[60] Dolores R Wallace and D Richard Kuhn. Failure modes in medical device software: an analysis of 15 years of recall data. *International Journal of Reliability, Quality and Safety Engineering*, 8(04):351–371, 2001.

[61] Haotian Weng, Chirath Hettiarachchi, Christopher Nolan, Hanna Suominen, and Artem Lenskiy. Ensuring security of artificial pancreas device system using homomorphic encryption. *Biomedical Signal Processing and Control*, 79:104044, 2023.

[62] Guanglou Zheng, Wencheng Yang, Craig Valli, Rajan Shankaran, Haider Abbas, Guanghe Zhang, Gengfa Fang, Junaid Chaudhry, and Li Qiao. Fingerprint access control for wireless insulin pump systems using cancelable delaunay triangulations. *IEEE Access*, 7:75629–75641, 2019.