2018 IEEE International Conference on Software Quality, Reliability and Security

# Analysis of Experiences with the Engineering of a Medical Device using State-based Formal Methods

Atif Mashkoor
Software Competence Center Hagenberg GmbH &
Johannes Kepler University Linz, Austria
atif.mashkoor@{scch|jku}.at

Alexander Egyed
Johannes Kepler University Linz
Linz, Austria
alexander.egyed@jku.at

***Abstract*—The use of software has become ubiquitous and prevalent in modern medical devices such as hemodialysis machines. Consequently, the failure rate of medical devices due to software faults is also increasing. While next-generation software-intensive medical devices contribute to providing better health care and ease of use, their development is becoming unprecedentedly complex and challenging. The critical nature of this domain – particularly its direct implications on health and safety – requires extraordinary measures to ensure the correct and reliable function of such systems. Formal methods are proven to provide approaches, techniques, and tools for correct engineering of software and systems. However, their use in the contemporary medical software engineering is still marginal. In order to promote the use of (state-based) formal methods and showcase their effectiveness in design and development of critical medical devices, we present the hemodialysis case study challenge problem in this article. We also analyze the novelties and limitations of several solutions implementing the case study and explore research challenges that still need to be addressed in future.**

## I. Introduction

While medical devices are increasingly becoming advanced, sophisticated, and software-intensive, due to the immaterial nature of software and their complexity, the development and certification of medical devices is becoming a fundamental issue. Most traditional software engineering methods cannot guarantee the correct functioning of software systems and this is not acceptable in this domain. Certification is the process where a regulatory authority evaluates the safety and fitness of a medical device for public use. Contemporary approaches for certifying a medical device typically include premarket approval and postmarketing surveillance. Premarket approval is a process where manufacturers are required to demonstrate the safety and effectiveness of a medical device to a regulatory authority prior to its introduction to the market. Postmarketing surveillance is the process that demands the development and demonstration of vigilance procedures to identify potential hazards with the deployed medical equipment. Actually, clinical trials are not always sufficient for a detailed and comprehensive analysis of a medical device.

This work is partially supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

Although stringent mechanisms already exist to control the overall quality of medical devices, numerous accidents have been reported and several of them resulted due to software errors. After carefully examining the recalls of defective medical devices during the first half of 2010, Sandler et al. [1] concluded that more than 25 percent of the problems were attributed to software faults. Wallace et al. [2] present analyses of software failures in medical devices, which are noncritical in nature but lead to device recalls. The authors recommended that formal specification of requirements and improved processes for quality assurance should be used for the engineering of complex medical software and systems.

One of the reasons for the increased software-related recalls of medical devices is that the regulatory regimes responsible for a medical device review are often constrained with a problem of limited resources and large burdens [3]. In the US, the Food and Drug Administration (FDA) is obliged to review a device within 90 calendar days[1] – a time frame that is clearly insufficient for a detailed review given the ever growing number of devices requiring clearance. Ironically, the review itself is based on several thousands of written pages and does not include the direct assessment of the device [4]. The onus is on the device manufacturer to demonstrate that the device is safe for public use.

Apart from human computer interaction errors, the reasons behind several incidents with medical devices are the inherent difficulties with performing four important steps in the development process: 1) identify in advance all possible hazardous situations; 2) identify safety requirements that can mitigate identified hazardous situations; 3) verify a system design against safety requirements; and 4) check that the system implements correctly given design documents. While risk and hazard analysis techniques, e.g., Failure Mode Effects Analysis (FMEA) [5] and Fault Tree Analysis (FTA) [6], can be used for steps 1 and 2, steps 3 and 4 can be effectively addressed using formal methods.

Formal methods are rigorous techniques and tools based on mathematics and logic for modeling, design and analysis (verification and validation) of software and systems. Formal methods enable specifiers to write unambiguous and consistent

[1] https://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/HowtoMarketYourDevice/PremarketSubmissions/PremarketNotification510k/ucm070201.htm

978-1-5386-7757-5/18/$31.00 ©2018 IEEE
DOI 10.1109/QRS.2018.00021

requirements and design specifications that can subsequently be transformed into software. More specifically, formal methods are used to prove that critical properties hold, and correctness and conformance of a software implementation to behavioral models expressing the core operations of a device can be guaranteed. In state-based formal methods, system specifications are expressed as a state model where mathematical structures like sets and functions represent the state and transitions among states define the model behavior. Some of the well-known state-based formal methods are Abstract State Machines [7], B [8], Event-B [9], and Z [10].

In order to promote a rigorous development approach and to provide formal models that can be used by academia and industry as a realistic workbench for demonstrating tools and techniques for strengthening cybersecurity, functional safety, reliability and usability of medical devices, some initiatives have already been taken. McMaster's Software Quality Research Laboratory in cooperation with Boston Scientific has made public the specification of an older pacemaker[2]. FDA has also sponsored a pilot project on generic infusion pumps[3].

Software Competence Center Hagenberg GmbH[4] – an industrial research facility based in Austria – has further extended this initiative by proposing a hemodialysis machine case study [11] and inviting researchers from academia and industry to contribute with their solutions. The ultimate aim of this initiative is: 1) to provide formal requirements and architectural models that can be used by device manufacturers early in their design process for safety analysis purposes, 2) to demonstrate the efficacy of formal methods for design and development of critical medical devices, and 3) to stimulate research endeavors related to the application of formal methods to safety- and security-critical medical devices.

The main goal of this article is to present the state of affairs related to the formal development of hemodialysis machines. We also examine (by highlighting novelties and limitations of) some of the deployed (state-based) formal methods for their capability to support the development of hemodialysis machines in particular and consequently the overall medical device software engineering process in general. We also discuss some challenges and future research directions within the context of this paper.

The article is organized as follows: In Section II, we present an overview of the hemodialysis process and discuss the related case study. In Section III, we discuss the state-of-the-art solutions implementing the case study. In Section IV, we present an overview of research directions and challenges associated with the formal development of hemodialysis machines. The article is concluded in Section V.

## II. Hemodialysis Challenge Problem

### A. The hemodialysis process

Kidneys are vital parts of a human body. Kidneys act as a filtration system that purifies blood by removing waste products and excess fluids. Additionally, kidneys balance chemicals (e.g., sodium, potassium, calcium, and phosphorous) in the blood, regulate the blood pressure, and stimulate the red blood cells production. When kidneys fail, the body starts accumulating unwanted toxic substances, the blood pressure rises, and the volume of body water increases. When kidneys stop working completely, an artificial process is needed to replace their functionality.

Hemodialysis is an extra-corporeal blood purification process and is the most popular treatment when the kidneys are in a state of renal failure. In a typical hemodialysis therapy, two needles are inserted into the patient's arm. The blood is drawn through the arterial puncture of the patient's arm. The blood then goes to the dialyzer by flowing through a thin tube. A dialyzer is composed of many fine hollow fibers which are made of semi-permeable membranes. While the blood is flowing through these fibers, the dialysate (a chemical substance) is mixed with the blood to remove its impurities and excess water and to adjust its chemical balance. Once the blood is cleansed and treated, it returns to the patient's arm through the venous access. A specific amount of blood is drawn outside the body of the patient at a time. The therapy takes 3 to 6 hours to complete and is performed on a regular basis. A professional caregiver monitors the treatment for compliance and safety. The hemodialysis process is briefly shown in Fig. 1.

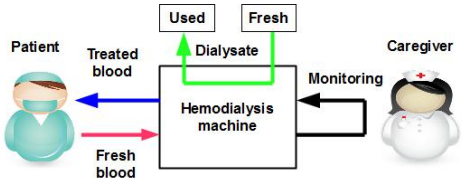



Fig. 1. Working principle of hemodialysis machines

### B. The challenge problem

In order to demonstrate the efficacy of formal techniques and tools in managing complexity and improving reliability of medical devices, we invite researchers to contribute to the hemodialysis machine case study challenge. The ultimate goal of the initiative is:

- to provide safe and reliable reference models that can be used by the device manufacturers early in their design process to verify critical properties,
- to establish assurance case arguments that can be used for certification purposes,
- to promote the use of state-of-the-art formal methods in design and development of safety-critical medical devices, and
- to stimulate research and development activities related to the application of formal methods to critical medical devices.

The hemodialysis machine case study challenge is described in [11]. It outlines machine's basic functionality, its safety conditions, and a top-level system architectural description. It

[2]http://sqrl.mcmaster.ca/_SQRLDocuments/PACEMAKER.pdf
[3]https://rtg.cis.upenn.edu/gip
[4]https://www.scch.at

describes how a typical hemodialysis therapy is performed that is comprised of three main phases:

1) Therapy preparation that is used to set the treatment parameters.
2) Therapy initiation that is used to physically connect the patient to the machine and perform the dialysis.
3) Therapy ending that is used to finalize the hemodialysis process by reinfusing the treated blood back to the patient.

This document also describes eight general safety requirements (including human computer interaction requirements) and 36 software safety requirements.

## III. Community response

Since the inception of the hemodialysis case study challenge, we have already received several solutions in various state-based formal methods. A few of the promising ones are discussed and analyzed in this section. It is worth noting that all these solutions succeeded in correctly formalizing the problem, i.e., the provided language constructs, the expressiveness, the theory and foundation of the employed approach, the adaptations, all were sufficient to model the problem at a satisfactory level. However, the main issue stems from the tool support associated with the employed formal methods, which proved, at times, limited and far from the satisfactory level.

### A. The Event-B solution

The first solution discussed here is presented by Hoang et al. [12] and is based on a set of tools revolving around the Event-B formalism. A similar Event-B inspired solution is presented in [13], [14]. Event-B is a formal method for systems and software engineering based on first-order predicate logic and set theory. A model in Event-B is composed of a state, invariants on the state, and a set of events defining the transitions on the state. The model development scheme associated with Event-B is based on formal refinement. The syntax and semantics of Event-B are designed so that, for each model, a set of independent proof obligations are generated. The model is considered verified when all proof obligations are discharged either interactively or automatically. The major strength of Event-B stems from the fact that the proof of correctness of the implementation is broken into many small and relatively simple proofs which are spread out all over the development process. The development of Event-B models is supported by the Rodin Platform [15].

As the hemodialysis machine's requirements involved extensive sequencing and dynamic interactions between the hemodialysis machine and the operator, the specifiers mixed the Event-B development with a Unified Modeling Language (UML) like notation, iUML-B [16], to model the sequential properties of the system, and ProB-based animation, visualization and simulation tools [17] to analyze the behavior of the model. In order to verify the safety constraints, the proof capabilities of Event-B and a co-simulation of the closed-loop parts of the controller with a continuous domain model of the environment were exploited.

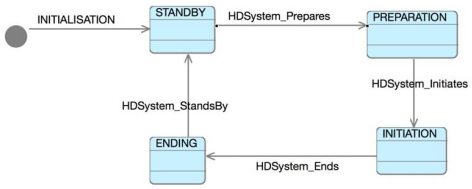



Fig. 2. Top level state machine representation of the model [12]

The solution model is specified using a refinement-based approach. The abstract model contains the main phases of the hemodialysis machine as shown in Fig. 2. It is then subsequently strengthened using 15 different refinement levels, each bringing new behaviors and safety requirements to the model. The model first introduces the sequential processes of hemodialysis machines using state-machines in the abstract model and is then concretized into several smaller incremental steps. Safety Properties are introduced as state-machine invariants. In order to validate the discrete behavior of the machine, the techniques of animation and model checking were used, whereas Domain Specific Visualizations (DSVs) were created to validate the continuous behavior of the model. Using the combination of animation, model checking and DSVs, the complete behavior of the machine can be simulated to even nontechnical stakeholders. For verification purposes, the technique of deductive theorem proving was used. In this technique, safety properties are defined as machine invariants that must be preserved by functional behavior of the specification. While most of the proofs were discharged automatically, some of them required human interaction.

The main novelty of the approach is its usage of a multi-formal development paradigm where the requirements are modeled using the UML-like notation and then subsequently verified in the formal framework of Event-B using the deductive theorem proving and model checking. The approach also lets the specification be validated using animation and DSVs. The main limitation of the Event-B approach, among others, is that the available code generation tools, e.g., [18], [19] and [20], are limited in functionality, only translate a subset of the B syntax during the automated translation process and also fail to establish that safety properties of the model remain preserved during the translation process.

### B. The Hybrid Event-B solution

The next solution discussed here is presented by Banach [21] and is based on Hybrid Event-B [22]. Hybrid Event-B is an extension of the Event-B method to facilitate designing of hybrid systems and complements Event-B by providing means to explicitly focus on continuously varying state evolutions. Like Event-B, a model in Hybrid Event-B is also composed of a state, invariant on the state, and a set of events defining the transitions on the state. However, Hybrid Event-B supports two kinds of variables: mode variables and pliant variables. While the former kind evolves via discrete

assignments, the later one evolves continuously. There are also two kinds of events supported by Hybrid Event-B. Mode events, like traditional events in Event-B, update the state instantaneously. Pliant events – newly introduced events in Hybrid Event-B – capture the continuous evolution of the state over a period of time, e.g., assignments are performed either through solving an ordinary differential equation or through a time-dependent expression. In Hybrid Event-B, time plays an important role and all variables are functions of time implicitly or explicitly. In Hybrid Event-B, the notion of time is read-only in general but the concept of clocks can be used more flexibly.

The Hybrid Event-B model of the hemodialysis machine is developed using a simple component-based approach. The model is composed of a central interface that stays in the center of the development and provides all needed shared variables. Four Event-B machines are specified in the model: Operator, modeling essential elements of the operator; Control, modeling the control system; BloodPump, modeling the behavior of blood pump; and SafetyAirDetector, modeling the safety air detector function that checks the blood in the tubing to be sure that air does not get into the bloodstream. Verification is achieved by expressing machine invariants and making sure that the behavior preserves them. The overview of the development architecture is shown in Fig. 3.

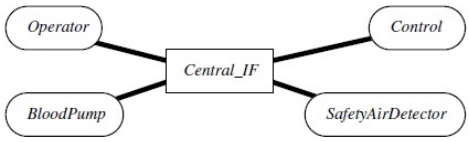



Fig. 3. Overview of the development architecture [21]

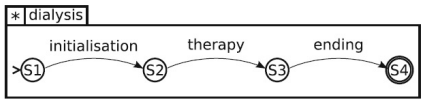
The development proceeds level by level. In the first level, basic vocabulary is introduced. The second level introduces the functionality of the blood pump. The third level introduces the safety air detector component. The fourth level deals with the preparation phase of dialysis that consists of rinsing and filling the machine, and entrance of treatment parameters. The initiation phase of the hemodialysis machine is modeled in the fifth level. The arterial connection is modeled via operator events. The filling of the blood tubing is performed by events synchronized between the operator, control and the blood pump. The treatment phase is modeled in the sixth level. The seventh level models the therapy ending phase.

The main novelty of the approach comes from its ability to explicitly distinguish between the discrete and continuous elements of hemodialysis machines. The resulted specification consists of two types of state transitions: the natural discrete changes of state and continuously varying state changes. The model takes the individual discrete events of hemodialysis machines and interleaves them with continuous events. This allows to specify the complete behavior of the model considering both discrete as well as continuous elements of hemodialysis machines. The limitation of the approach is the absence of the native tool support for Hybrid Event-B. Hybrid Event-B relies on the Rodin platform for specification and proving but some features of Hybrid Event-B are not supported by the Rodin platform, e.g., specification of continuously changing behavior, support for single and multi-machine systems, and continuous domain model checking.

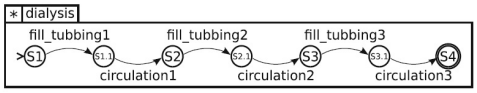
### C. The ASTD solution

The third solution discussed here is presented by Fayolle et al. [23] and is based on Algebraic State Transition Diagrams (ASTD) [24]. ASTD is a graphical notation that enables to model problems using state transition diagrams and classical process algebra. Actually, in ASTD, basic ingredients of state transition diagrams, e.g., hierarchy, OR-states, AND-states, guards, and history states, are fused with some noticeable features of process algebra, e.g., event synchronization, event traces, and environment-driven labeled transitions. While ASTD is used in this solution to specify the correct ordering of actions and to constrain the execution of events, the Event-B method and its refinement-based development paradigm is used to capture the data model and safety properties.

The model is specified using a refinement-based approach. At the most abstract level, the three main phases of hemodialysis machines are described, i.e., therapy preparation, therapy initiation, and therapy ending. Then they are detailed in subsequent refinements one at a time. The abstract level and the first refinement is shown in Fig. 4. One of the latter refinement levels is used to introduce failures in the system and to describe how to deal with those failures if they occur.




(a) First specification level




(b) First refinement of ASTD specification

Fig. 4. Initial ASTDs of the hemodialysis machine model [23]

The main novelty of this approach comes from a multi-formal development technique and the way how the sequencing order of the machine is described. Due to the use of a graphical notation, the model is easy to follow and validate. For verification purposes, the model relies on the strength of the Event-B platform that stems from theorem proving and model checking. However, this means that the approach also suffers from the limitations associated with the Event-B method and its tool-set, e.g., absence of an implicit notion of time, absence of a mechanical proof that Event-B refinements maintain system's temporal properties, and poor quality of the automatically generated code.

### D. The ASM solution

The next solution discussed here is presented by Arcaini et al. [25] and is based on the Abstract State Machines (ASM) method. The ASM method allows the engineering of synchronous/asynchronous multi-agent systems from requirements elicitation to their implementation. A typical ASM model can also be seen as an intuitive form of abstract pseudo-code. In this solution, the researchers used the Asmeta framework [26] to model the behavior and properties of hemodialysis machines.

The basis of the ASM method is provided by a rigorous mathematical theory of finite state machines where states are defined by arbitrarily complex data structures using functions. State is made up of locations that are concrete values of function parameters. The control flow is maintained through conditional statements and a set of rule constructors that defines state transitions by modifying the function values at a finite number of locations. Derived functions, which constitute an important auxiliary element in ASM models, compute values from a combination of functions at runtime. Two kinds of functions are supported by ASMs: static, that never change during any run of the machine, and dynamic, that are updated by agent actions. Additionally, functions are either monitored, i.e., only read by the machine and modified by the environment, or controlled, i.e., read and written by the machine.

```
asm Hemodialysis_GM
signature:
   enum domain Phases =
   {PREPARATION | INITIATION | ENDING}
   controlled phase: Phases

definitions:
   rule r_preparation = phase := INITIATION
   rule r_initiation = phase := ENDING
   rule r_ending = skip
```

```
rule r_run_dialysis =
 switch(phase)
    case PREPARATION: r_preparation[]
    case INITIATION: r_initiation[]
    case ENDING: r_ending[]
 endswitch
main rule r_Main =
   r_run_dialysis[]
default init s0:
   function phase = PREPARATION
```

Fig. 5. The ASM ground model of the hemodialysis machine [25]

In the ground (the most abstract) model, as shown in Fig. 5, specifiers simply describe the transitions among different phases of hemodialysis machines. During the first refinement step, the ground model is extended by including the details of the preparation phase. The second refinement extends the model with the initiation phase. The third refinement step models the activities that are performed at the end of the treatment, i.e., the behavior of the ending phase. In the last refinement step, property verification and error handling is performed.

Validation and Verification of the model are achieved by employing multiple formal techniques. At first, the interactive simulation is performed. During the interactive simulation, values are provided to monitored functions of the machine and the state computation is observed. The simulator, at each step, monitors that all the updates are consistent. Then a more powerful form of simulation is performed that is called the scenario-based validation. It permits to automate the simulation activity. A further quality assurance technique of model review can then be applied to the model to examine whether the model has some particular qualities that should help in developing, maintaining, and enhancing it. As a last quality assurance step, safety requirements of the system are specified as Linear Temporal Logic (LTL) formulas and model checked for verification purposes.

The main novelty of this solution comes from its rigorous approach to quality assurance and *easy to understand* formal notation [27]. The employed comprehensive model analysis approach based on activities like model checking, simulation, model review, testing, and conformance checking, stands out among others as far as the notion of model correctness is concerned. The limitation of the approach is that theorem proving is ignored during the development and property verification is only performed using model checking. This may not be sufficient for large, complex high-integrity medical devices, such as hemodialysis machines, due to inherent problems associated with model checking, e.g., state explosion. Additionally, only a limited-capability tool [28] is currently available for automatic transformation of an ASM model into a C++ programming language code – the translation works currently only for Arduino platforms[5].

### E. The Circus solution

The last solution discussed in this article is presented by Gomes et al. [29] and is based on Circus [30]. Circus is a combination of the Z formal method, Communicating Sequential Processes (CSP) [31] and the refinement calculus [32]. It defines systems as processes where the structural part of systems (state and invariants on the state) is described using the Z method and the behavioral part (state transitions) using CSP. While the specification style of Circus is inspired by Z, CSP provides the communication aspects. A typical Circus specification includes Z paragraphs, channel definitions and declarations, and process definitions. Circus also features parallelism, choice, assignment statements, and control structures, e.g., loops and conditions.

The solution starts with the modeling of hemodialysis machine requirements as Circus actions using pre- and post-conditions. Once all requirements are specified, then the general behavior of the machine is described to check the compliance. The functional behavior of the machine is defined in terms of circus processes. The system is modeled as a parallelism between the main therapy process and the requirements process. A second parallelism is modeled between the aforementioned parallelism and another process describing the state phases. The latter is a process that monitors the state transitions through signals and later on sets the values of the state variables to comply with the requirements process. Then the components are put in parallel with the process responsible for sensors readings. This is shown in Fig. 6. Finally, the entire system is put in parallel with the process managing the timing properties of the system.

The main novelty of the model is the use of a well-defined theory of process algebra to specify concurrent and parallel

[5]https://www.arduino.cc/

$$HDMachine \mathrel{\widehat=} HDGenCompInit \,;\; \left( \left( \begin{pmatrix} MainTherapy \\ [\![ HDGenCompStChanSet ]\!] \; SoftwareRequirements \end{pmatrix} \right) \right) \begin{matrix} [\![ TherapyPhaseChanSet ]\!] \; StatePhase \\ [\![ SensorReadingsComm ]\!] \; SensorReadings \end{matrix}$$

Fig. 6. The Circus model showing parallel processes of the solution [29]

aspects of the system. The explicit focus on timing properties of hemodialysis machines is another point that distinguishes this work among others. The main limitation of the approach is that no tool currently exists that directly supports the consistency and conformance checking of a Circus specification. In the current solution, the Circus model is translated into a machine-readable CSP which is then checked for verification purposes by tools such as Failures Divergences Refinement (FDR) [33]. Absence of tools for automatic code generation from Circus models is another limitation.

## IV. Research roadmap

In this section, we discuss the necessary outstanding issues associated with the (formal) software development of medical devices in general and hemodialysis machines in particular that still need to be addressed.

### A. Need for an integrated formal approach

A formal model-driven development paradigm lets engineers build software and systems that are correct by construction, i.e., verified and validated. However, there exists no single formal method, technique or tool that can be used alone for a complete development of medical device software.

Every formal method has its strengths and limitations. We already discussed that while some formal methods excel in verification and validation activities, some focus explicitly on correct sequential behavioral ordering. While some methods are centered around timing and temporal constraints, some enjoy extensive tool support or are easy to understand and use. The dataflow-oriented frameworks already popular in avionic and automotive domains, such as MATLAB Simulink[6] and Safety-Critical Application Development Environment (SCADE)[7], also lack sophisticated verification techniques which guarantee *correctness* [34] and suffer from scalability issues [35]. A comprehensive analysis of various formal methods is available in [36]. An ideal formal development of medical device software will actually be a mesh of several methods, techniques and tools working harmoniously together to achieve a common goal. However, interaction among different formal methods is currently a weak link in the development chain.

### B. Need for supplementary artifacts and better certification processes

A safe and correct medical device is the result of several artifacts supplementing each other, e.g., requirements specifications, design descriptions, and hazard analysis documents. [11] provides an overview of functional and safety requirements, and a high-level design of a hemodialysis machine. This document lays down a foundation for the model-based engineering, formal analysis and systematic implementation of medical device software, which are highly recommended activities by experts of the domain [37]. However, further artifacts are also necessary to complete the development process.

A very important artifact that is often required by regularity authorities is the document containing potential risks and hazards associated with the device. This is achieved through hazard analysis, which is a risk assessment process that identifies, documents and analyzes potential risks and hazards associated with the device such as operational, environmental, electrical, hardware, software, mechanical, chemical and usage hazards. The most effective tools for hazard and risk analysis are FMEA and FTA. FMEA is an inductive approach applied at the beginning of the design phase that assumes a basic defect in the system design, assesses its effect, and identifies potential solutions. FTA, on the other hand, is a deductive approach to hazard analysis. In this case, a risk or a hazard is discovered and analyzed for effects, and the initiating faults and events. Traditionally formal methods have a little to do with the hazard analysis process, but the process can be supplemented by formal approaches for improved performance and accuracy.

Regularity authorities also want to make sure that a medical device has already achieved an acceptable assurance level before being released. One of the ways to convince regulatory authorities in this regard is to prove this through assurance cases [38]. An assurance case validates the claim by supplying a conclusive argument supported by evidence such as test cases or program analysis. Again formal methods can provide assistance in this process by providing a compelling, comprehensive and valid proof that the designed system is dependable for the given problem and environment. For example, by using formal methods we can show that the system specification satisfies all desired safety properties, the code generation process transforms the specification into a correct piece of code, and the generated code satisfies all the intended properties with specified time limits.

As medical devices are becoming unprecedentedly complex, current certification processes are approaching their limits [37]. This situation results in expensive manufacturing costs, delays in finishing a marketable product, and increased chances of device failures, recalls and liability costs. Ironically, there exists no standard for testing, verification and validation of software for (specialized) medical devices such as implantable cardiac devices or extra-corporeal blood treatment devices. Although, there are some general medical standards like IEC 62304 for software life cycle processes, or guidelines like general principles of software validation[8], they are insufficient and incomplete considering the criticality of such devices.

During the premarket evaluation, currently, FDA neither

[6]http://www.mathworks.com/products/simulink/

[7]http://www.esterel-technologies.com/products/scade-suite/

[8]http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085371.pdf

requires the review of medical device software nor provides specific requirements for its verification [39]. The responsibility for testing, verification and validation of medical device software remains with device manufacturers which in turn show that they have applied established quality assurance techniques to certain levels of coverage. In the contemporary quality assurance techniques, verification and validation activities are performed quite late and often at the end of the design phase. In this case, problems are hard, costly and time consuming to fix.

Formal methods advocate a design strategy where design inconsistencies and requirements errors are detected and corrected in earlier stages of development. Through a formal proof one can also show that the design is potentially defect free, correctly implement its specification, and meets safety, security and reliability standards. There are already some proposals, such as [40] and [41], that demonstrate how safety and reliability of medical device software can be guaranteed using formal methods. As medical device software is becoming a major safety concern, there is a need for better and improved certification processes that ensure the trustworthiness and reliability of medical devices including their software.

Human computer interaction plays a significant role in medical devices. Some of the major sources for device recalls are often attributed to flawed human computer interaction designs, e.g., lack of guidance to control incorrect insertion of device parameters, insufficient control to prevent incorrect dosage of medicine, and poor user interfaces. In some cases, but not all, validation techniques, such as animation, simulation and Functional Mock-up Interface (FMI), can provide a clearer picture about the design, operation and usability of medical devices, which eventually helps in minimizing chances of trivial errors. On the other hand, scenarios like design of an easier process for clamping blood supply lines and making sure that all lines are clamped as needed through the dialysis program are, of course, beyond the jurisdiction of formal methods.

### C. Need to address cybersecurity concerns

In recent times, traditional safety-critical medical devices are also increasingly becoming security-critical. Modern medical devices are being designed as a system of systems that is composed of heterogeneous components addressing various networking, dynamic and uncertain environmental constraints. For example, with the possibility of using dialysis machines at homes, things to consider are: real-time monitoring and intervention by a remote caregiver in case of emergency, and easy data and information exchange between patients and clinics. As various components of hemodialysis machines start communicating with each other, this interoperability feature necessitates the consideration of security issues as well such that the proper functioning of the device is not affected by cybersecurity and privacy threats, e.g., hacking, hijacking or even ransomware.

In the future, we also need to focus on integrated modeling of functional safety and cybersecurity requirements and how to solve any possible conflict that may arise due to this integration [42]. For example, a safety condition may require that the medical procedure under performance is remotely supervised by a caregiver all the time so that the caregiver can intervene in case of a serious situation. However, a security requirement may suggest that in case of intrusion detection, the device can shutdown the network access completely or run in a minimal capability mode, i.e., send out the data, such as sensor readings and event logs, and do not accept commands from the network. Formal methods can bridge this gap between functional safety and cybersecurity requirements by their integrated modeling and analysis, e.g., the approach presented in [42] models both safety and security requirements in a unified model, and inconsistencies and conflicts between (safety and security) requirements can easily be spotted out during model refinement.

## V. Conclusion

Hemodialysis machines are safety-critical medical devices whose design and engineering include several intricate processes, activities and tasks. The software components of hemodialysis machines are characterized by high complexity, susceptible to frequent changes, and subject to several stringent certification and regulatory requirements. It is an established fact that the use of formal methods for modeling, design and development guarantees that defects are spotted and corrected earlier [43], thus making software safe, trustworthy and reliable.

As far as modeling is concerned, the current state-of-the-art of state-based formal methods is quite advanced to fulfill the requirements of modern medical device software development. For example, static and dynamic properties of systems like safety, security, dependability can be expressed and verified [42], and hybrid aspects of systems can be modeled and analyzed, be it discrete/continuous [21] or hardware/software [44]. However, as far as a method's tool-support is concerned, this is not as satisfactory as its modeling capability. For example, all tools have some weaknesses be it automatic code generation, mechanical proving of properties, automatic theorem proving, or state-space explosion. A detailed comparison of state-based formal methods' strengths and weaknesses is available in [36] and maybe interesting particularly for medical device manufacturers.

After analyzing several solutions implementing the hemodialysis machine case study, an example of high-confidence medical devices, we have reached to this conclusion that despite their numerous advantages in modeling, design and analysis, formal methods still need to address some issues associated with their harmonious integration and tool chains before they become mainstream methods for the development of medical software.

## References

[1] K. Sandler, L. Ohrstrom, L. Moy, and R. McVay, "Killed by code: Software transparency in implantable medical devices," *Software Freedom Law Center*, pp. 308–319, 2010.

[2] D. R. Wallace and D. R. Kuhn, "Failure modes in medical device software: an analysis of 15 years of recall data," *International Journal of Reliability, Quality and Safety Engineering*, vol. 8, no. 04, pp. 351–371, 2001.

[3] D. Zuckerman, P. Brown, and S. Nissen, "Medical device recalls and the FDA approval process," *Archive of Internal Medicine*, vol. 171, no. 11, pp. 1006–1011, 2011.

[4] P. Masci, A. Ayoub, P. Curzon, I. Lee, O. Sokolsky, and H. Thimbleby, "Model-Based Development of the Generic PCA Infusion Pump User Interface Prototype in PVS," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, F. Bitsch, J. Guiochet, and M. Kaniche, Eds. Springer Berlin Heidelberg, 2013, vol. 8153, pp. 228–240.

[5] D. H. Stamatis, *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press, 2003.

[6] R. E. Barlow, *Fault Tree Analysis*. John Wiley & Sons, Inc., 1973.

[7] E. Börger and R. F. Stark, *Abstract State Machines: A Method for High-Level System Design and Analysis*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[8] J.-R. Abrial, *The B Book*. Cambridge University Press, 1996.

[9] ——, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[10] J. M. Spivey, *The Z notation: a reference manual*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[11] A. Mashkoor, "The Hemodialysis Machine Case Study," in *5th International Conference on ASMs, Alloy, B, TLA, VDM, and Z (ABZ'16)*, Springer, 2016, vol. 9675, pp. 329–343.

[12] T. S. Hoang, C. Snook, L. Ladenberger, and M. Butler, "Validating the Requirements and Design of a Hemodialysis Machine Using iUML-B, BMotion Studio, and Co-simulation," in *5th International Conference on ASM, Alloy, B, TLA, VDM, and Z (ABZ'16)*, ser. Lecture Notes in Computer Science. Springer, 2016, vol. 9675, pp. 360–375.

[13] A. Mashkoor and M. Biro, "Towards the trustworthy development of active medical devices: A hemodialysis case study," *IEEE Embedded Systems Letters*, vol. 8, no. 1, pp. 14–17, 2016.

[14] A. Mashkoor, "Model-driven development of high-assurance active medical devices," *Software Quality Journal*, vol. 24, no. 3, pp. 571–596, 2016.

[15] J.-R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in Event-B," *International Journal on Software Tools for Technology Transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[16] C. Snook, "iUML-B state machines," *Proceedings of the 5th Rodin User and Developer Workshop*, 2014.

[17] M. Leuschel and M. Butler, "ProB: An Automated Analysis Toolset for the B Method," *Journal Software Tools for Technology Transfer*, vol. 10, no. 2, pp. 185–203, 2008.

[18] A. Fürst, T. Hoang, D. Basin, K. Desai, N. Sato, and K. Miyazaki, "Code Generation for Event-B," in *Integrated Formal Methods*, ser. Lecture Notes in Computer Science, E. Albert and E. Sekerinski, Eds. Springer International Publishing, 2014, vol. 8739, pp. 323–338.

[19] D. Méry and N. K. Singh, "Automatic code generation from Event-B models," in *Proceedings of the Second Symposium on Information and Communication Technology*, ser. SoICT'11. New York, NY, USA: ACM, 2011, pp. 179–188.

[20] A. Edmunds, M. Butler, I. Maamria, R. Silva, and C. Lovell, "Event-B Code Generation: Type Extension with Theories," in *Abstract State Machines, Alloy, B, VDM, and Z*, ser. Lecture Notes in Computer Science, J. Derrick, J. Fitzgerald, S. Gnesi, S. Khurshid, M. Leuschel, S. Reeves, and E. Riccobene, Eds. Springer Berlin Heidelberg, 2012, vol. 7316, pp. 365–368.

[21] R. Banach, "Hemodialysis Machine in Hybrid Event-B," in *5th International Conference on ASM, Alloy, B, TLA, VDM, and Z (ABZ'16)*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2016, vol. 9675, pp. 376–393.

[22] R. Banach, M. Butler, S. Qin, N. Verma, and H. Zhu, "Core Hybrid Event-B I: Single Hybrid Event-B Machines," *Science of Computer Programming*, vol. 105, pp. 92 – 123, 2015.

[23] T. Fayolle, M. Frappier, F. Gervais, and R. Laleau, "Modelling a Hemodialysis Machine using Algebraic State-Transition Diagrams and B-like Methods," in *5th International Conference on ASM, Alloy, B, TLA, VDM, and Z (ABZ'16)*, ser. Lecture Notes in Computer Science. Springer, 2016, vol. 9675, pp. 394–408.

[24] M. Frappier, F. Gervais, R. Laleau, B. Fraikin, and R. St-Denis, "Extending statecharts with process algebra operators," *Innovations in Systems and Software Engineering*, vol. 4, no. 3, pp. 285–292, 2008.

[25] P. Arcaini, S. Bonfanti, A. Gargantini, A. Mashkoor, and E. Riccobene, "Integrating formal methods into medical software development: The ASM approach," *Science of Computer Programming*, vol. 158, pp. 148-167, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642317301430

[26] A. Gargantini, E. Riccobene, and P. Scandurra, "Model-Driven Language Engineering: The ASMETA Case Study," in *Software Engineering Advances, 2008. ICSEA'08. The Third International Conference on*, 2008, pp. 373–378.

[27] F. Kossak, A. Mashkoor, V. Geist, and C. Illibauer, "Improving the understandability of formal specifications: An experience report," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, C. Salinesi and I. Weerd, Eds. Springer International Publishing, 2014, vol. 8396, pp. 184–199.

[28] S. Bonfanti, M. Carissoni, A. Gargantini, and A. Mashkoor, *Asm2C++: A Tool for Code Generation from Abstract State Machines to Arduino*. Cham: Springer International Publishing, 2017, pp. 295–301.

[29] A. O. Gomes and A. Butterfield, "Modelling the Haemodialysis Machine with Circus," in *5th International Conference on ASM, Alloy, B, TLA, VDM, and Z (ABZ'16)*, ser. Lecture Notes in Computer Science. Springer, 2016, vol. 9675, pp. 409–424.

[30] J. Woodcock and A. Cavalcanti, *The Semantics of Circus*. Berlin, Heidelberg: Springer, 2002, pp. 184–203.

[31] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice Hall Int., 1985.

[32] R.-J. Back and J. Wright, *Refinement calculus: a systematic introduction*. Springer Science & Business Media, 1998.

[33] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. Roscoe, "FDR3 — A Modern Refinement Checker for CSP," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, E. Ábrahám and K. Havelund, Eds., vol. 8413, 2014, pp. 187–201.

[34] I. Daskaya, M. Huhn, and S. Milius, *Formal Methods for Industrial Critical Systems: 16th International Workshop, FMICS 2011, Trento, Italy, August 29-30, 2011. Proceedings*. Springer Berlin Heidelberg, 2011, ch. Formal Safety Analysis in Industrial Practice, pp. 68–84.

[35] R. Reicherdt and S. Glesner, *Proceedings of the 12th International Conference Software Engineering and Formal Methods (SEFM'14)*. Springer International Publishing, 2014, ch. Formal Verification of Discrete-Time MATLAB/Simulink Models Using Boogie, pp. 190–204.

[36] F. Kossak and A. Mashkoor, "How to select the suitable formal method for an industrial application: A survey," in *Proceedings of the Fifth International Conference on ASMs, Alloy, B, TLA, VDM, and Z (ABZ'16)*. Springer International Publishing, 2016, pp. 213–228.

[37] I. Lee, G. J. Pappas, R. Cleaveland, J. Hatcliff, B. H. Krogh, P. Lee, H. Rubin, and L. Sha, "High-confidence medical device software and systems," *Computer*, vol. 39, no. 4, pp. 33–38, April 2006.

[38] P. Bishop, R. Bloomfield, and S. Guerra, "The future of goal-based assurance cases," in *Workshop on Assurance Cases*, 2004, pp. 390–395.

[39] Z. Jiang, M. Pajic, and R. Mangharam, "Cyber-physical modeling of implantable cardiac medical devices," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 122–137, Jan 2012.

[40] R. Jetley, S. P. Iyer, and P. Jones, "A formal methods approach to medical device review," *Computer*, vol. 39, no. 4, pp. 61–67, 2006.

[41] R. Jetley, S. P. Iyer, P. L. Jones, and W. Spees, "A formal approach to pre-market review for medical device software," in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, vol. 1, Sept 2006, pp. 169–177.

[42] A. Mashkoor and J. Sametinger, "Rigorous modeling and analysis of interoperable medical devices," in *Modeling and Simulation in Medicine Symposium (MSM'16)*. ACM, 2016, pp. 800–807.

[43] G. J. Holzmann, "Formal methods for early fault detection," in *FTRTFT'96*, Springer, 1996, pp. 40–54.

[44] A. Buga, A. Mashkoor, S. T. Nemeş, K.-D. Schewe, and P. Songprasop, "Conceptual Modelling of Hybrid Systems," *MEDI'17*, Springer, 2017, pp. 277–290.