The Open PCA Pump Project

An Exemplar Open Source Medical Device as a Community Resource

John Hatcliff
Brian Larson
hatcliff@ksu.edu
brl@ksu.edu
Kansas State University

Todd Carpenter
todd.carpenter@adventiumlabs.com
Adventium Labs

Paul Jones
Yi Zhang
Joseph Jorgens
Paul.Jones@fda.hhs.gov
Paul.Jones@fda.hhs.gov
Joseph.Jorgens@fda.hhs.gov
US Food and Drug Administration

ABSTRACT

Building safe and secure interoperable medical devices with accompanying assurance artifacts is challenging. Many start-up companies have great ideas for innovation, but are not familiar with appropriate safety/security-critical engineering processes, architecture principles, risk management, and assurance techniques. Larger, more experienced, companies may face hurdles in re-engineering their devices for interoperability and greater security. In academia, researchers often have good techniques for addressing some of the issues above, but are not familiar with how a realistic medical device is developed and assured. Building a prototype medical device for a classroom project or research work to validate proposed techniques is often a huge effort.

The Open PCA Pump illustrates a full suite of realistic development artifacts including use cases, requirements, architecture models, verified source code, testing and simulation infrastructure, risk management artifacts, and assurance cases that can be used to develop shared understanding of medical device innovations across the academic, industry, and regulatory communities.[1]

CCS CONCEPTS

• **Computer systems organization** → *Embedded software*;

KEYWORDS

reference architecture, requirements, assurance case, AADL, medical device, PCA pump, interoperable, safety, security, exemplary design artifacts, model-based engineering, BLESS, formal specification, software correctness proofs

1 INTRODUCTION

Many improvements in medical device capabilities are being achieved through rapid advances in technology. More effective sensing of physiological parameters, less disruptive and more controllable therapy delivery, greater computational power provided by processors that use less resources, better data storage and communication, more effective data analytics, smaller form factors, lower manufacturing costs, and greater architectural flexibility are all contributing to device innovation. Advances in interoperability are enabling individual devices to be flexibly arranged into different "system of systems" structures to achieve greater care-giver work flow efficiencies, improved automation through "closed loop" decision making and control, and more effective use of data [10].

However, there are challenges in continuing and accelerating the pace of innovation while achieving appropriate medical device safety and security. Architectures are growing more complex. Small companies have interesting ideas for innovation, but they are often not familiar with safety/security-critical development approaches or relevant regulatory regimes. Larger companies are struggling to adapt legacy architectures and code bases to address security concerns and flexibility needed for developing families of related products. while maintaining consistency and backward compatibility with legacy products. Industry teams often lack the skills necessary to develop safe, secure, and efficient closed loop control applications.

Medical devices are examples of safety-critical cyber-physical systems (CPS) – they are built from, and depend upon, the seamless integration of computational and physical components.[2] Numerous well-documented opportunities exist for CPS science and CPS community activites to impact the medical device domain, and conversely the medical world has many interesting problems that can drive research in the CPS community [21].

Continued progress in bridging these domains requires better cross-fertilization between

a) Academic communities including CPS, software engineering, hybrid systems, and formal verification research areas,
b) Medical device companies and industry associations,
c) Health care practitioners,
d) Standards development organizations and regulatory agencies, and
e) Third-party safety/security certification organizations.

Unfortunately, several barriers inhibit shared learning. For examples, medical device manufacturers invest their own money to make advances, and may choose to keep certain information proprietary so they can remain competitive, and regulatory agencies are

[1] This work is sponsored in part by US National Science Foundation Food and Drug Administration Scholar-in-Residence program (CNS 1238431,1355778,1446544,1565544), the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA/CDS) BAA HSHQDC- 14-R-B0005, the Government of Israel and the National Cyber Bureau in the Government of Israel via contract number D16PC00057.

MCPS'18, April 2018, Porto, Portugal.
The authors wished to acknowledge the second author Brian Larson for his vision for the Open PCA Pump and his work as the primary creator of the original Open PCA Pump artifacts.

[2]https://www.nsf.gov/news/special_reports/cyber-physical/

constrained from reporting on details of safety/security problems to avoid revealing proprietary information.

In academia, students and researchers don't have access to realistic devices (and especially device implementations and artifacts used in development of devices) – due to manufacturers keeping proprietary information for themselves. Additionally, students and instructors find it difficult to access medical device development and safety standards, and they don't understand the concerns of regulatory reviewers or nature of regulatory artifacts [13]. CPS researchers are producing significant technical advances, but for the reasons above they often do not realistically position their technologies in the context of a medical device. For example, significant gaps in safety and security engineering and regulatory evidence can impede uptake of new innovations.

Similarly, industry may also be unfamiliar with advanced systems engineering and software development technologies. Even if they are familiar, without convincing demonstration of benefits in a medical device context, they may view untried technologies as unacceptable cost and schedule risks.

Regulatory agencies are often unfamiliar with advanced CPS architectures, modeling, and verification technologies, and without evidence, they cannot assume these innovations will support manufacturer safety and security claims that they are responsible for reviewing [13]. Correspondingly, the CPS community itself does not understand or has not made an appropriate effort to explain how their solutions lead to reduced risks and increased confidence in the correctness of claims.

Standards development organizations, which typically work to codify community best practices in standards requirements, are unfamiliar with how to accommodate advanced technologies. Moreover, they don't have the basic non-proprietary illustrations of applications of those technologies to include in standards rationale and supporting informative annexes and technical reports.

2 OPEN PCA PUMP PROJECT GOALS

The Open Patient Controlled Analgesic (PCA) Pump project [28] was created to help break down these barriers. The project is a joint research effort between Kansas State University researchers, US Food and Drug Administration (FDA) engineers, and industry experts. The project aims to provide an open source design artifacts for a realistic medical device – a PCA Pump – that illustrate advanced development and assurance technologies in the context of realistic development processes and work products used to support safety and security reviews.

The development of the original Open PCA Pump artifacts was sponsored by the US National Science Foundation FDA Scholar-in-Residence Program, which has the goals of (a) improving regulatory science by transitioning academic research results in quality processes and rigorous development, verification, and assurance techniques, (b) informing the academic community of regulatory, safety, and security challenges.

The Open PCA Pump artifacts are being used by the Intrinsically Secure, Open, and Safe Cyber-physically Enabled, Life-critical Essential Services (ISOSCELES) project sponsored by the US Department of Homeland Security Cyber-Physical System Security (CPSSec) research program led by Adventium Labs. The DHS CPSSec project supports research to improve the security of critical infrastructure technologies. The ISOSCELES project is developing an open-source software platform, running on generic hardware, to provide both safety and security features for networked, interoperable medical devices to be used by (small) manufacturers more knowledgeable about medical function than computer security. The original Open PCA Pump requirements and architecture have been adapted to show how the general ISOSCELES platform can be specialized for a particular medical function.

3 AVAILABLE MATERIALS

Below we provide a summary of currently-available materials.

3.1 Development Artifacts

Concept of Operations A concept of operations document documents stakeholders, user needs, and an extensive collection of use cases addressing normal operations and response to fault conditions and other safety/security-related events. Use case diagrams capturing important device/patient/operator interactions are defined and simulated with jUCMNav [1].

Requirements Following the US Federal Aviation Administration's Requirements Engineering Management Handbook, [22] a requirements document [20] provides functional, safety, and security requirements, tracing of the requirements to stakeholders, goals, and use- or exception-cases, and allocation of requirements to functional architecture component(s) responsible for implementing each requirement.

Architecture The system architecture is specified in the Architecture Analysis and Design Language (AADL).

Formal Behavior Specifications Specifications are attached to the architecture using the Behavior Language for Embedded Systems with Software (BLESS) [17][19].

Implementation Real-time thread behaviors are specified using state machines attached as BLESS annex subclauses and are fully implemented using KSU's Sireum Scala-based framework from which C code suitable for embedded systems is generated.

Correctness Proof Inductive proofs that implementation met specification are stated and proved using the BLESS framework.

Risk Management Error behavior is modeled with AADL's error modeling annex (EMV2) [15][18] for hazard analysis and risk controls.

Assurance Case An assurance case for the Open PCA Pump's safety and effectiveness is specified using NOR-STA [24], which references requirements, architecture and verification artifacts.

3.2 Domain Background Materials

As noted in Section 1, one of the challenges for academics working on safety-critical systems (and medical devices in particular) is to understand real domain issues. Two of the authors gained this experience by directly working in the medical device community for several years in engineering leadership roles, but this is not always convenient. To address this, the Open PCA Pump project gathered resources that can be used by individual researchers or in classroom settings to understand real-world PCA pump requirements and use-cases in real-world settings.

Example categories of Open PCA Pump domain resources:

PCA Pump Training Videos Links to publicly available training for nurses on the setup and operation of PCA pumps.

Clinical Guidelines Guidelines, workflows, and best practices for safety from hospitals and health care.

Physician Order Forms Forms used by hospitals physicians to specify PCA pump therapy for patients.

PCA Pump Product Manuals User manuals and service manuals for selected PCA pumps currently on the market.

Safety Issues Clinical studies of pump safety issues and video lectures from industry experts on PCA pump safety incidents.

3.3 Pedagogical Materials

The Open PCA Pump project provides the following resources for classroom instructors and individual researchers:

PCA Pump Background Slides and recorded video lectures on PCA Pump background and clinical use.

Requirements Slides and recorded lectures on engineering requirements for safety-critical systems using the FAA's Requirements Engineering Management Handbook (REMH) [22].

Architecture Specifications Slides and recorded lectures on use of AADL on a simple medical device example.

Risk Management Lecture slides overviewing medical device risk management terminology and the ISO 14971 medical device risk management standard.

Project Concepts Description of possible classroom / research project involving the Open PCA Pump material.

4 PCA PUMP BACKGROUND

A PCA infusion pump is used to infuse pain medication into a patient's blood stream through an intravenous (IV) line. Pain medication is prescribed by a licensed physician, which is dispensed by the hospital's pharmacy. The pharmacy places the drug into a labeled vial, and the vial is typically moved by hospital staff to a drug cabinet located in the hospital ward associated with the patient's room. A clinician retrieves the vial from the drug cabinet, loads the vial into the pump, and attaches the pump's drug dispensing tube to the patient's IV line. By interacting with the pump's operator interface, the clinician enter parameters that indicate the prescribed drug and amounts of drug to infuse during different infusion modes. The pump infuses a prescribed basal flow rate (basal infusion mode) which may be augmented by a patient-requested bolus (this mode is activated when the patient presses a hand-held button) or a clinician-requested bolus (this mode is activated by the clinician via the device operator interface). When entering parameters, the clinician also set limits on total drug volume that can be infused over a set time period (e.g., one hour) and also sets a "lock out interval" indicating a time period that must elapse between each patient bolus dose. These limits provide safeguards against overinfusion of the drug – which is a significant hazard associated with a PCA Pump.

PCA pumps, unfortunately, have been associated with a large number of adverse events [5, 14]. The FDA notes [6] that while PCA pumps (and infusion pumps in general) have allowed for a greater level of control, accuracy, and precision in drug delivery—thereby reducing medication errors and contributing to improvements in

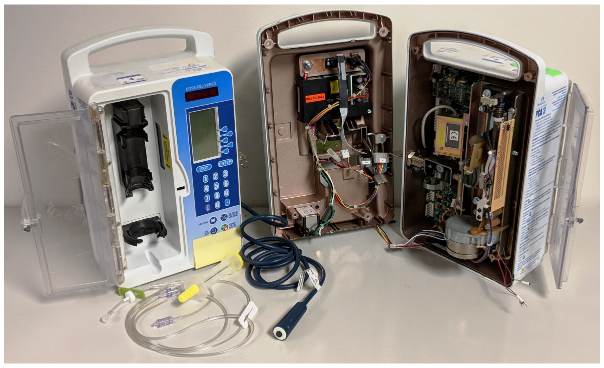
Figure 1: Example PCA Pump

patient care—infusion pumps have been associated with persistent safety problems. From 2005 through 2009, 87 infusion pump recalls were conducted by firms to address identified safety problems. Infusion pump problems have been observed across multiple manufacturers and pump types. Through analysis of pump-related adverse event reports and device recalls, FDA has concluded that many of these problems appear to be related to deficiencies in device design and engineering[6].

Through the *Infusion Pump Improvement Initiative* [6], FDA is taking broad steps to reduce infusion pump problems. Specifically, FDA aims to establish additional requirements for infusion pump manufacturers, proactively facilitate device improvements, and increase user awareness of problems and best engineering practices. As an example of best engineering practices, the FDA Guidance for Infusion Pumps [7] now recommends that pump manufacturers provide an assurance case with their regulatory submissions. These activities indicate the significant concerns that FDA has with pump safety, and they provide an impetus for research in software engineering, safety, security, and verification & validation applied to pump development. We hope to enable such research to some extent with the Open PCA Pump artifacts described herein.

5 OPEN PCA PUMP DEVELOPMENT ARTIFACTS

In the sections below, we give a brief walkthrough of the exemplar artifacts provided by the Open PCA Pump Project.

5.1 Concept of Operations

The purpose of a Concept of Operations (ConOps), e.g., presented in INCOSE Systems Engineering Handbook [29] is to identify users (more generally stakeholders), the environment in which system will operate, the needs to the users to be addressed by system functionality, and use cases of how users expect to interact with the pump. The ConOps documents the detailed use and exception cases following the methodology presented in the FAA's REMH [22] (which is based on [4]). Use cases describe normal operation. Exception cases describe response to hazards or other deviations from expected operation.

Both use and exception cases were developed as use case maps – graphical depictions of use case actions, actors, and branches – using jUCMNav [1]. Textual use and exception cases correspond to their use case maps. The ability of jUCMNav to define, set, and test variables denoting properties of use case steps, and execute 'scenarios' revealed several errors and omissions in use cases.

Use cases are linked to other artifacts in the following ways:

- the description of functional behavior in the use cases are used to identify primary functions of the device and to derive requirements for those functions,
- use case interactions indicate primary interfaces for the device which are reflected in the device architecture,
- use cases indicate information to be exchanged which is refined into requirements for data models and contraints,
- use cases indicate risk controls including moving the device to a safe state, operator notifications, etc.,
- use cases indicate security functions including operator authorization, etc., and
- uses indicate sequences of interactions that should be reflected in system test cases.

Researchers can experiment with different use case formalizations, and techniques to derive requirements and tests from use cases.

5.2 Requirements

Open PCA Pump requirements derive from use and exception cases of the ConOps. The requirements address the infusion functionality of the primary infusion modes, correctness of information input/output over the operator interface (including standards compliant alarm notifications), validation of operator-entered infusion settings using drug libraries, functionality of risk controls that system faults and exceptional circumstances, and security features of the pump. The requirements also address functionality of an interoperability interface for accessing pump functionality over the network via a medical application platform [10, 16]. Each requirement has a unique textual identifier to support traceability.

The requirements document also includes a section on system design. This section provides a high-level overview of the Open PCA Pump architecture, and listings that allocate each requirement to one or more components of the design. This enables traceability between the architecture and the requirements (using the requirements identifiers mentioned above).

Researchers can use the requirements as case studies for analyzing requirements consistency and completeness, expressing requirements in various formal specification languages, decomposing requirements to interface specifications, or building requirements compliant implementations.

5.3 Architecture

An AADL system architecture defines the structure of the PCA pump as components with precisely-defined interactions. OMG's SysML [25] is a popular, *general purpose* system modeling language. In contrast, SAE International's AADL was created specifically for *embedded electronics systems using software*, and for that purpose it is far superior. AADL has both graphical and textual representations. The graphical editor in the Open-Source AADL Tool Environment (OSATE) keeps graphical and textual representations synchronized allowing both to be edited consecutively.

Figure 2 shows the AADL graphical view of the PCA pump's high-level functional architecture. The fluid subsystem holds the drug reservoir, mechanical pump, and monitors of flow rate and pressure. The power subsystem provides stable DC power from AC mains or battery backup. The operation subsystem determines normal behavior and some exceptional behavior. The safety subsystem monitors other subsystems for malfunction, and handles exception cases to provide patient safety. The most basic elements of the functional architecture are hardware devices and software threads. AADL properties specify whether a thread is periodic (time-triggered) or sporadic (event-triggered). Communication between elements is realized using publish-subscribe event communication (with optional event payloads) or shared data cells with automatic propagation of updates between components that use the cell. These basic notions enable expression of a variety of high-level communication patterns. Real-time properties of both threading and communication can be specified using AADL properties.

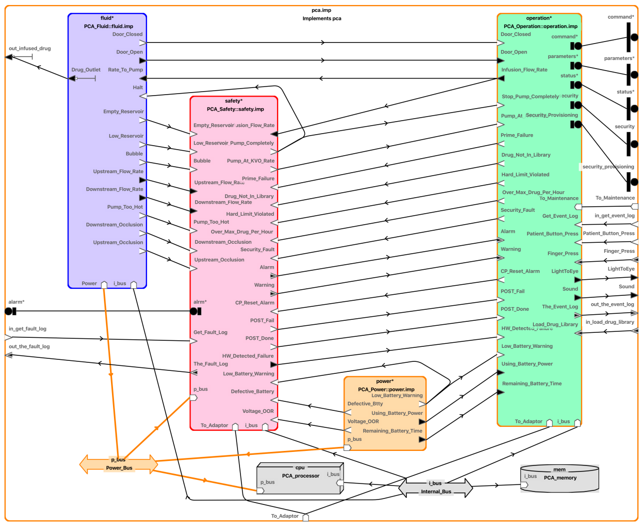

Figure 2: Functional Architecture

The PCA Pump architecture specification serves as a "single source of truth" about system components and communication from which source code for interfaces and communication is automatically generated. The architecture provides structural abstraction of the system for framing several different types of analyses including schedulability,control/data flow, and error propagation (see [11] for an expanded discussion). It is also used for defining interaction points and interfaces to which one may add contracts that capture behavioral interface specifications (see Section 5.4). Having the architecture serve as a touch point for all of these aspects encourages consistency and traceability.

Researchers can use the architecture description to compare and contrast other architecture description languages, evaluate different forms of architecture level analyses, and evaluate architecture-driven code generation approaches.

5.4 Formal Specifications and Behavior

The Open PCA Pump artifacts provide an excellent opportunity to illustrate and evaluate formal behavioral specification languages as well techniques for demonstrating the implementations comply to formal specification languages. The Open PCA Pump artifacts include specifications and behaviors written in the Behavior Language for Embedded Systems with Software (BLESS).

AADL can be enhanced with annex sublanguages. BLESS is phrased as an AADL annex sublanguage. Components are specified by `BLESS::Assertion` properties attached to component features, usually ports. For out(going) event ports, the assertion property states what is true about the system at the instant an event is issued. Conversely, in(coming) event port assertions state what is assumed at the instant an event arrives at that port. Connections between ports support assume-guarantee contracts.

Individual component behavior is expressed in BLESS annex subclauses as state-transition machines, that may perform an action (execute a simple program) when a transition is taken. The state-transition machines are annotated with assertion to be a 'proof outline' containing all of the information necessary to prove that operational behavior meets its declarative specification. The BLESS proof engine (a plugin to OSATE available at [19]) generates verification conditions, and with interactive selection of tactics, transforms proof outlines into an inductive proof that every possible execution meets its specification.

A safety feature of the pump that reduces the chance of overdose is a physician-specified "lock out interval" that disables the effect of the patient bolus request button for a period of time after the initiation of a patient bolus. The prevents the patient from giving themselves doses too frequently. The `Patient_Bolus_Checker` thread enforces the lock out interval and shows a simple example of BLESS specification and behavior. Assertions are contained within double angle brackets (« »).

```
thread Patient_Bolus_Checker
  features
    Minimum_Time_Between_Bolus: in data port ICE_Types::Minute
      {BLESS::Assertion => "<<:=MINIMUM_TIME_BETWEEN_BOLUS>>";};
    Patient_Button_Request: in event port;
    Patient_Request_Not_Too_Soon: out event port
      {BLESS::Assertion => "<<PATIENT_REQUEST_NOT_TOO_SOON(now)>>";};
    Patient_Request_Too_Soon: out event port
      {BLESS::Assertion => "<<PATIENT_REQUEST_TOO_SOON(now)>>";};
end Patient_Bolus_Checker;
```

```
thread implementation Patient_Bolus_Checker.i
  annex BLESS
  {**
  invariant <<LPB()>>
  variables
    last_patient_bolus: time:=0
      <<LPB: :Patient_Request_Not_Too_Soon@last_patient_bolus and
        not (exists t:time in last_patient_bolus,,now
            that Patient_Request_Not_Too_Soon@t)>>;
  states
    start: initial state
      <<last_patient_bolus=0 and now=0 and LPB()>>;
    run: complete state
      <<LPB()>>;
    check_last_bolus_time: state
      <<LPB() and Patient_Button_Request@now>>;
    done: final state;
 transitions
   go: start-[ ]-> run{};
   button: run -[on dispatch Patient_Button_Request]-> check_last_bolus_time{};
   nottoosoon: check_last_bolus_time
       -[(now-Minimum_Time_Between_Bolus?) >  last_patient_bolus]-> run
     {  <<LPB() and Patient_Button_Request@now and
        (now-MINIMUM_TIME_BETWEEN_BOLUS@now) > last_patient_bolus>>
     Patient_Request_Not_Too_Soon!
      ; <<Patient_Request_Not_Too_Soon@now>>
      last_patient_bolus:=now
     <<Patient_Request_Not_Too_Soon@now and last_patient_bolus=now>>};
   toosoon: check_last_bolus_time
     -[(now-Minimum_Time_Between_Bolus?) <= last_patient_bolus]-> run
     {Patient_Request_Too_Soon!};
   quit: run-[on dispatch stop]->done{};
  **};
end Patient_Bolus_Checker.i;
```

5.5 Risk Management

International standards such as ISO 14971 mandate risk management processes for medical device develop that including identify how a device might harm the patient, performing hazard analysis to identify the faults and other root causes that can lead to harms, designing risk controls to reduce the severity or likelihood of such scenarios, and verifying the risk controls are appropriately implemented. AADL's Error Modeling annex (EMv2) provides annotations that can be attached to the architecture specifications to document possible sources of faults, component failure modes, propagations of the effects of faults and errors, and points at which a system can cause harms by interacting with its environment. Various reports can be automatically derived from this information including Fault Tree Analyses (FTA) diagrams and Failure Modes and Effects Analysis (FMEA) tables (see [18] for an illustration of EMv2 applied to a simpler infant incubator medical device).

The Open PCA Pump artifacts currently provide EMv2 annotations for selected components. Complete annotations will be added in the coming months. The risk management content is essential for medical device safety compliance and regulatory submissions, but it is often overlooked in the academic community. Researchers can use the Open PCA Pump risk management artifacts to understand how a realistic hazard analysis might be constructed for a medical device, experiment with extracting the EMv2-based information into different types of hazard analysis reporting, evaluate alternative approaches to capturing hazard analyses with different levels of precision, investigate solution strategies to risk management challenges in interoperable medical systems [12], and propose strategies for how risk controls might be specified and verified in conjunction with the hazard analysis models.

6 ISOSCELES

The Open PCA Pump Project provides two ways to experiment with pump implementations. The KSU Sireum framework [27] is used to automatically generate source-level interfaces and communication infrastructure, and developers can use Sireum to (a) implement the internal behaviors of components (or these can be autogenerated from BLESS) and (b) simulate the behavior of the entire pump. The simulation prototype provides an operator interface graphical user interface and demonstration interfaces for seeding various types of faults and safety-related events.

The ISOSCELES project [3] is providing a platform on which the Open PCA Pump can be deployed. ISOSCELES is a reference implementation for mixed-criticality medical and Internet of Things (IoT) systems. Based on a strong separation architecture, the reference implementation enables manufacturers to focus on the clinical side of their product, reducing the time and effort recreating the underlying safe and secure platform and associated regulatory evidence. ISOSCELES has been instantiated on the Xen hypervisor, and the seL4 and NOVA separation microkernels. The Xen version is suitable for low criticality devices, and easily supports rapid prototyping. The seL4 version is highly hardened and fits in a small memory footprint. ISOSCELES targets low-power x86 and ARM embedded processors. The current seL4 prototype runs on Intel-Atom and AMD G-series. The Xen prototype runs on ARM Cortex-A7 and AMD G-series. To realize the hardware aspects of the Open PCA Pump, these embedded prototypes drive the electromechanical components of a decommissioned PCA pump.

7 RELATED WORK

The Open PCA Pump requirements document builds upon University of Pennsylvania's and FDA's Generic Infusion Pump (GIP) project [8]. The GIP project includes a much smaller set of requirements and loosely connected set of artifacts. These GIP artifacts were utilized in follow-on work by researchers at UPenn / FDA and elsewhere on the application of verification techniques that tended to emphasize properties that could be checked by real-time model checkers like UPPAAL [2]. Researchers at the University of Minnesota adapted the GIP requirements to a PCA Pump and produced an AADL architectural model that defined the boundaries of key subsystem and specified properties that could be verified via model-checking. Our work expands on the original GIP requirements document in a number of dimensions, and provides a collection of deeply integrated, consistent, and realistic artifacts as well as both software-simulated and hardware-based realizations.

Masci and colleagues have multiple studies of how infusion pump operator interfaces and associated device state changes can be verified against user interface requirements in PVS [9, 23].

The second author of this paper was instrumental in obtaining the release of a PACEMAKER System Specification (PSS) [26] from Boston Scientific, for teaching embedded software, building hardware prototypes, cyber-physical system modeling, and formal specification and verification. Since its release over ten years ago, the PSS has been used in over forty academic publications. The Open PCA Pump artifacts are a much more expansive collection of integrated artifacts, and we hope that the success of the PSS is indication of the potential impact that the Open PCA Pump project can have on the community.

8 CONCLUSIONS AND FUTURE WORK

The integrated artifacts provided by the Open PCA Pump were developed jointly by academic researchers, industry safety engineers, and FDA personnel to enable a greater cross-fertilization of ideas between multiple communities. In current interactions with the FDA, we using the artifacts to further explore organization and reviewing approaches for assurance artifacts for interoperable systems [30]. In standards committees, we are using the artifacts to illustrate architectural specification and risk management approaches. In the ISOSCELES work, the artifacts are being used to illustrate how the ISOSCELES platform can be applied. Current research efforts are adding verified implementations of the pump, fault-injection testing frameworks, and implementations of the pump on different separation platforms.

REFERENCES

[1] Daniel Amyot. 2018. jUCMNav - Eclipse plugin for the User Requirements Notation. http://jucmnav.softwareengineering.ca/foswiki/ProjetSEG/WebHome. (2018).

[2] David Arney, Raoul Jetley, Paul Jones, Insup Lee, and Oleg Sokolsky. 2007. Formal Methods Based Development of a PCA Infusion Pump Reference Model: Generic Infusion Pump (GIP) Project. In *Proceedings of 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*.

[3] Todd Carpenter, John Hatcliff, and Eugene Y. Vasserman. 2017. A Reference Separation Architecture for Mixed-Criticality Medical and IoT Devices. In *Proceedings of the 1st ACM Workshop on the Internet of Safe Things (SafeThings'17)*. ACM, New York, NY, USA, 14–19.

[4] Alistar Cockburn. 2001. *Writing Effective Use Cases.* Addison-Wesley, Boston, MA.

[5] Joint Commission. 2005. Preventing Patient-Controlled Analgesia Overdose. *Joint Commission Perspectives on Patient Safety* (October 2005), 11.

[6] FDA Infusion 2010. US FDA Infusion Pump Improvement Initiative. (April 2010).

[7] FDA Infusion Pump Guidance 2014. Total Product Life Cycle: Infusion Pump - Guidance for Industry and FDA Staff. https://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm209337.pdf. (2014).

[8] Generic Infusion Pump [n. d.]. Generic Infusion Pump Project Homepage. http://rtg.cis.upenn.edu/gip.php3. ([n. d.]).

[9] Michael D. Harrison, Paolo Masci, Jose Creissac Campos, and Paul Curzon. 2017. Demonstrating that Medical Devices Satisfy User Related Safety Requirements. In *Software Engineering in Health Care*, Michaela Huhn and Laurie Williams (Eds.). Springer International Publishing, Cham, 113–128.

[10] John Hatcliff, Andrew King, Insup Lee, Anura Fernandez, Alaisdair McDonald, and Eugene Vasserman. 2012. Rationale and Architecture Principles for Medical Application Platforms. In *Proceedings of the 2012 International Conference on Cyberphysical Systems*.

[11] John Hatcliff, Brian R. Larson, Jason Belt, Robby, and Yi Zhang. 2018. A Unified Approach for Modeling, Developing, and Assuring Critical Systems. In *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 225–245.

[12] J. Hatcliff, E. Y. Vasserman, T. Carpenter, and R. Whillock. 2018. Challenges of distributed risk management for medical application platforms. In *2018 IEEE Symposium on Product Compliance Engineering (ISPCE)*. 1–14.

[13] John Hatcliff, Alan Wassyng, Tim Kelly, Cyrille Comar, and Paul L. Jones. 2014. Certifiably safe software-dependent systems: Challenges and directions. In *Proceedings of the on Future of Software Engineering (ICSE FOSE)*. 182–200. https://doi.org/10.1145/2593882.2593895

[14] Rodney W. Hicks, Vanja Sikirica, Winnie Nelson, Jeff R. Schein, and Diane D. Cousins. 2008. Medication errors involving patient-controlled analgesia. *American Journal of Health-System Pharmacy* 65, 5 (March 2008), 429–440.

[15] SAE International. 2015. *SAE AS5506/1, AADL Annex E: Error Model Annex.* SAE International, http://www.sae.org.

[16] Andrew King, Dave Arney, Insup Lee, Oleg Sokolsky, John Hatcliff, and Sam Procter. 2010. Prototyping Closed Loop Physiologic Control with the Medical Device Coordination Framework. In *ICSE Companion*.

[17] Brian Larson, Patrice Chalin, and John Hatcliff. 2013. BLESS: Formal Specification and Verification of Behaviors for Embedded Systems with Software. In *Proceedings of the 2013 NASA Formal Methods Conference (Lecture Notes in Computer Science)*, Vol. 7871. Springer-Verlag, Berlin Heidelberg, 276–290.

[18] Brian Larson, John Hatcliff, Kim Fowler, and Julien Delange. 2013. Illustrating the AADL Error Modeling Annex (V.2) Using a Simple Safety-critical Medical Device. In *Proceedings of the 2013 ACM SIGAda Annual Conference on High Integrity Language Technology (HILT '13)*. ACM, New York, NY, USA, 65–84.

[19] Brian R Larson. 2018. Behavior Language for Embedded Systems with Software (BLESS) website. http://bless.santoslab.org. (2018).

[20] Brian R Larson, John Hatcliff, and Patrice Chalin. 2013. Open Source Patient-Controlled Analgesic Pump Requirements Documentation. In *Proceedings of the 5th International Workshop on Software Engineering in Health Care*. IEEE, Piscataway, NJ, 28–34. https://doi.org/10.1109/SEHC.2013.6602474

[21] I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, A. Roederer, and K. K. Venkatasubramanian. 2012. Challenges and Research Directions in Medical Cyber-Physical Systems. *Proc. IEEE* 100, 1 (Jan 2012), 75–90. https://doi.org/10.1109/JPROC.2011.2165270

[22] D. Lempia and S. Miller. 2009. *Requirement Engineering Management Handbook.* Technical Report DOT/FAA/AR-08/32. US Federal Aviation Administration.

[23] Paolo Masci, Yi Zhang, Paul Jones, Paul Curzon, and Harold Thimbleby. 2014. Formal Verification of Medical Device User Interfaces Using PVS. In *Proceedings of the 17th International Conference on Fundamental Approaches to Software Engineering - Volume 8411*. 200–214.

[24] Gdansk University of Technology. 2018. NOR-STA: Support for Achieving and Assessing Conformance to NORms and STAndards. http://www.nor-sta.eu/en. (2018).

[25] Object Modeling Group (OMG). 2017. OMG System Modeling Language (SysML) v1.5. http://www.omg.org/spec/SysML/1.5/. (2017).

[26] Boston Scientific. 2007. PACEMAKER System Specification. http://sqrl.mcmaster.ca/pacemaker.htm. (2007).

[27] Sireum [n. d.]. Sireum: A high-assurance software development platform. http://sireum.org. ([n. d.]).

[28] Kansas State University. 2018. Open PCA Pump Project. http://openpcapump.santoslab.org. (2018).

[29] D.D. Walden, G.J. Roedler, K.J. Forsberg, R.D. Hamelin, and T.M. Shortell (Eds.). 2015. *INCOSE Systems Engineering Handbook.* Wiley, Hoboken, NJ.

[30] Yi Zhang, Brian Larson, and John Hatcliff. 2018. Assurance Case Considerations for Interoperable Medical Systems. In *Computer Safety, Reliability, and Security*, Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 42–48.