Model-based Risk Analysis for an Open-Source PCA Pump using the AADL Error Modeling Annex*

Hariharan Thiagarajan[1], Brian Larson[2], John Hatcliff[1], and Yi Zhang[3]

[1] Department of Computer Science
Kansas State University, Manhattan, KS, USA
{thari, hatcliff}@ksu.edu
[2] Multitude Corporation, St. Paul, MN, USA
brl@multitude.net
[3] The MD PnP Interoperability & Cybersecurity Program
Massachusetts General Hospital, Boston, MA, USA
yzhang134@mgh.harvard,edu

Abstract. Risk management is a key part of the development of medical devices to achieve acceptable product safety and pass regulatory scrutiny. As model-based development (MBD) techniques are gaining ground in the medical device industry, the medical device industry needs guidelines on the best practices of integrating risk management principles and activities in MBD-driven product development.
In this paper, we demonstrate how the SAE standard Architecture, Analysis, and Definition Language (AADL) and its Error Modeling (EM) annex can be applied in the development of an open-source patient-controlled analgesic (PCA) pump to support the risk management tasks of ISO 14971 - the primary risk management standard in the medical device domain. While AADL EM has been applied in other domains, our work provides the first mapping of AADL EM to ISO 14971 concepts. It not only represents one of the largest applications to-date of AADL's EM framework, but also provides the industry and academia an example with considerable complexity to investigate methodologies and methods of integrating MBD and risk management. This work is part of the Open PCA Pump project, which presents a variety of open source integrated development artifacts for a realistic medical device.

Keywords: Error Modeling · Medical Device · Risk Analysis · Architecture Analysis and Design Language (AADL).

1 Introduction

The medical device domain, like other safety-critical domains, includes risk management as a key activity in development and certification. The international standard ISO 14971 [16] describes a risk management process for medical devices that has been adopted world-wide. The 14971 process includes identifying hazards (things associated with the device and its use that might cause harm), performing risk analysis (including hazard analysis) to identify hazardous situations (causality chains leading from root causes to device-user / device-patient interactions that might cause harm), developing risk controls (mitigations of hazard situations), and verifying risk controls.

Conceptually, risk management activities are interleaved with requirements engineering, design, implementation, verification as part of broader system engineering activities – activities that are similar to those in other safety-critical domains. However, the medical device domain has some unique challenges. Unlike other domains such

* This work was supported in part under the U.S. Army Medical Research Acquisition Activity Contract W81XWH-17-C-0251.

as avionics, nuclear, and even automotive, the medical device domain includes many small companies that do not have adequate resources and experienced safety staff to enable rigorous risk management. Additionally, products in the medical device space vary significantly in their architecture, safety concerns, and clinical uses. For these and other reasons, it is challenging for new manufacturers to understand both the risk management process, hazard analysis techniques, and how they should be integrated into the development life-cycle of a device.

The Open PCA Pump project [12, 21] was created to illustrate rigorous model-driven engineering (MBE) and formal methods on realistic medical device artifacts. With team members from industry, healthcare delivery organizations, regulatory agencies, and academia, the project aims to overcome barriers between these groups that have hindered the explanation of development and verification challenges, certification and regulatory concerns, and benefits of formal development techniques. The project addresses patient controlled analgesic (PCA) pumps – clinical care bedside pain relief devices that are widely used, despite having a history of safety problems. The project maintains a broad collection of linked and deeply integrated artifacts including use cases, concepts of operation, requirements, formal architectural models, formal behavioral specifications, testing and verification artifacts, and assurance cases.

As part of this effort, the Open PCA project is investigating the use of model-based safety analysis (MBSA) techniques in the context of the SAE standard Architecture, Analysis, and Definition Language (AADL) [1] – the modeling framework used on the project. AADL includes as a standardized Error Modeling (EM) annex [2] that provides modeling annotations for MBSA, and the Eclipse-based OSATE integrated development environment provdes basic analysis and reporting capabilities for the EM annex.

Unfortunately, there are few publicly available completely worked examples of AADL EM for large systems. The examples that exist are related to the avionics domain, and the EM annotation libraries supplied with OSATE are oriented to avionics domain (e.g., ARP 4761). Thus, in its current state, AADL EM and associated tooling does not include infrastructure to directly support the concepts, terminology, and reporting for medical domain risk management (ISO 14971), and there are no realistic medical examples that could help medical industry engineers and regulators understand AADL-based MBSA, its integration into the medical device domain, and its potential benefits.

The contributions of this paper are as follows:

- We report on an MBSA analysis for one of the largest, most complex medical device examples considered in the academic/industry literature to date. This work also represents one of the most complete to-scale application of the AADL EM SAE standard in any domain.
- We illustrate one approach to developing model annotation libraries that instantiate the AADL EM framework to support ISO 14971 risk management concepts,
- We demonstrate how a scalable dependence and error flow analysis framework called Awas [5] that we have developed for AADL can support key steps of ISO 14971 risk analysis that uses AADL EM,
- We summarize how the above framework can be used to support the overall risk management process explicitly defined in medical device standards.

All of the artifacts described in this paper including AADL models, EM specifications, and analysis reports are freely available under an open source license, along with other

Open PCA Pump artifacts (requirements, formal specifications, assurance cases, etc.) on our project website [21].[4]

2 Background - PCA Infusion Pump

A PCA infusion pump is a medical device intended to administer intravenous (IV) infusion of pain medications to the patient in a variety of clinical settings. During clinical use, a caregiver (typically nurse) first prepares the PCA pump by loading a vial of medication into the pump, priming the pump's infuset set (tubing and needle), and connecting the pump to the patient via infusion set. The caregiver then configures infusion paparemters (e.g., infusion volume, rate, and duration) on the pump's operator interface to initate the infusion.

A PCA pump is able to deliver medication in either a basal or bolus mode, where the former continuously delivers medication at a low rate and the latter delivers a bulk of medication in a short period of time. The patient can request for additional boluses for further pain relief by pressing a hand-held button provided by the pump, although too many patient-requested boluses can pose severe overdosing risks to the patient.

While PCA pumps (and infusion pumps in general) have allowed for a greater level of control and accuracy in drug delivery, they have been associated with persistent safety problems [27]. Through a study of adverse events and device recalls related to infusion pumps, the US Food and Drug Administration (FDA) concluded that many of these problems appear to be related to deficiencies in device design and engineering [28]. This conclusion has led FDA to take a broad set of steps to enhance infusion pump safety [28], including imposing specical design control over infusion pumps coming to the market [9]. This special control requires, among many other things, manufacturers to use risk management best practices in identifying and controling risks assocaited with their products.

3 Open PCA Pump Architecture

Many of the Open PCA Pump artifacts integrate with or have traceability relationships with AADL EM MBSA. The most closely related artifact is the architecture model, into which the annotations for the AADL EM MBSA are integrated. It is well beyond the scope of this paper to describe the architecture models in detail. Readers can download the full architectural model from the project website [21] and find a high-level description in [14]. In this section, we give a brief summary of the architectural model, focusing on attributes that are related to the MBSA.

The Open PCA Pump extends and specializes the ISOSCELES medical device reference architecture [8]. ISOSCELES defines a reference architecture as an AADL model that separates functional architecture (including software) from the physical architecture (components, wires and assemblies). The ISOSCELES reference architecture includes generic subsystems for operation, safety, user interface, network interface, power, and sensors/actuators which perform a medical function. The Open PCA Pump AADL model extends the ISOSCELES AADL model with sensors and actuators for drug infusion, and detailed software behavior.

Figure 1 shows the Open PCA Pump containment hierarchy which retains the ISOSCELES architectural layering of a functional architecture using ISOSCELES runtime services, isolated by a separation kernel, executed by physical hardware. The full architecture includes separate AADL projects for the ISOSCELES medical device

[4] The following is a direct link to the artifacts for this paper https://awas.sireum.org/doc/03-risk-analysis/index.html.

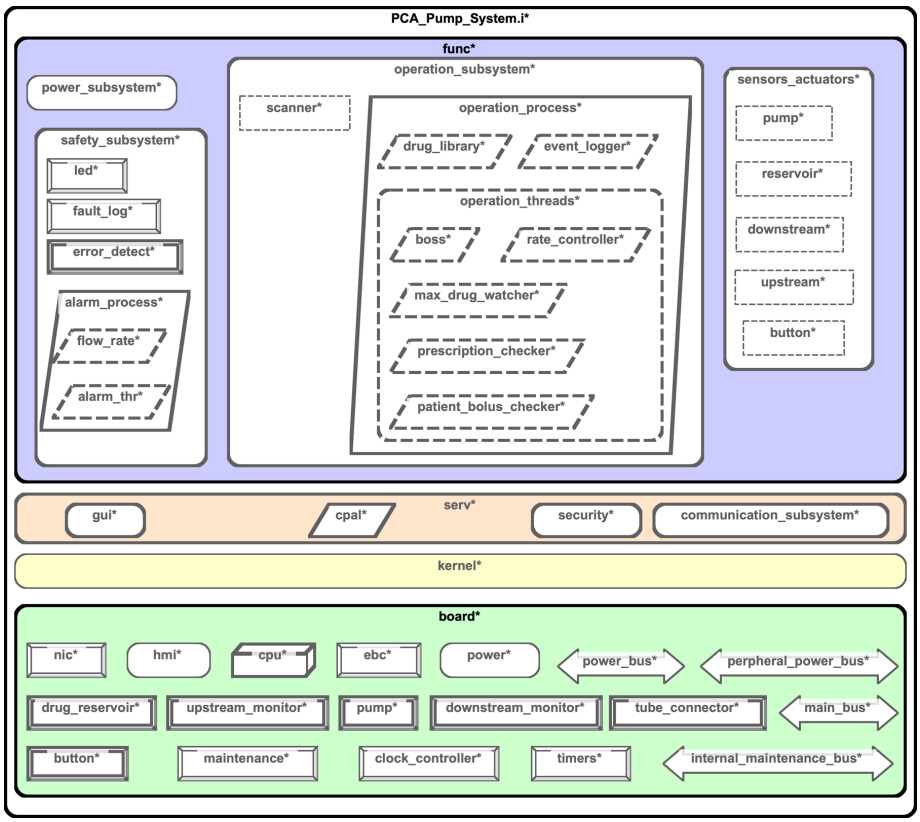


Fig. 1. Open PCA Pump Containment Hierarchy

platform and its Open PCA Pump refinement, having thirty-nine packages together. AADL distinguishes component *types* which define externally-visible interfaces, from component *implementations* which define internal behavior and decomposition into subcomponents. The Open PCA Pump AADL model defines 121 component types and implementations.

Figure 2 shows the top-level of the Open PCA Pump functional architecture, and its four subsystems: operation, safety, sensors/actuators (fluid), and power. The ports and connections between components are modeled using AADL feature groups – with each connection (line) aggregating many event and data flows (these are automatically broken out in visualizations of Section 6).

In contrast to other popular UML-based modeling languages, AADL also has textual view of the architecture. The emphasis of the AADL textual models in AADL tooling and work flows aids in the blending of modeling and programming of system design, while graphical and textual architecture models are synchronized by OSATE.

The Safety subsystem exemplifies a medical device safety architecture that seperate software and hardware used for detecting unsafe conditions and mitigating hazards from those used for normal device operation. Following the principles in [19], the Safety subsystem is designed to have three physical devices that can detect faults, log them, and indicate that infusion is halted due to a detected fault. More complex faults are detected by software (AADL threads), executing in a protected address space (AADL process).

The EM SA covers both hardware and software components. A variety of formal specifications, including specifications for certain risk control ports, are integrated into the architecture using the Behavior Language for Embedded Systems with Software

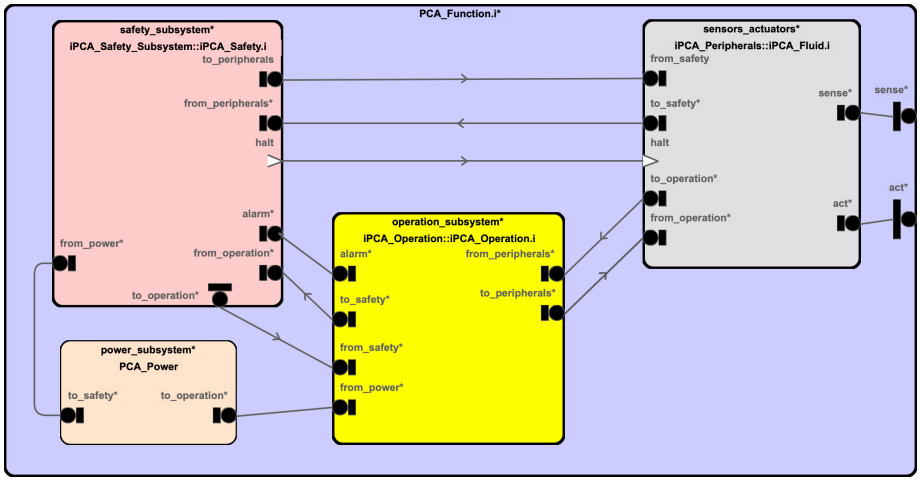


Fig. 2. PCA Pump Functional Architecture

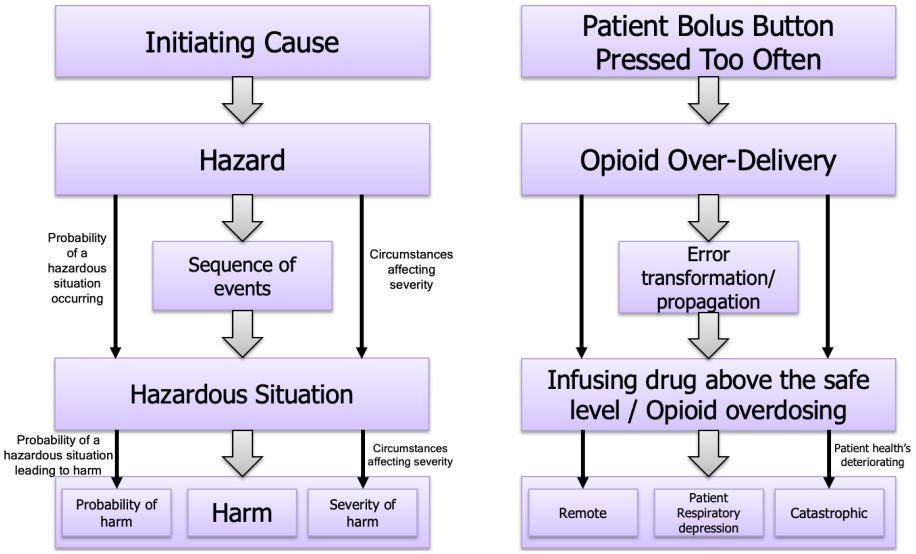


Fig. 3. ISO 14971 Key Risk Analysis Terms and Relationships

[18] – these can be verified against AADL Behavior Annex state machines using the BLESS proof engine.

4 Medical Device Risk Management Concepts

ISO 14971 defines a number of concepts that need to be reflected in medical device MBSA. *Harm* is defined as "injury or damage to the health of people, or damage to property or the environment". *Hazard* is a "potential source of harm", and a *hazardous situation* is a "circumstance in which people, property or the environment is/are exposed to one or more hazards." *Initiating cause* is not a ISO 14971 defined term, but it is used in the standard to refer to faults or other issues that lead to a hazard.

The left side of Figure 3 (slightly adapted from ISO 14971 Annex C) illustrates the relationships between these terms. The scope of our work is *functional safety* – potential harms associated with incorrect function of software and hardware elements of the device, rather than physical, chemical, mechanical, electrical, and biological safety discussed in ISO 14971. Table 1 provides instances of the terms of Figure 3 related to

Harm	Hazardous Situation	Hazard	Initiating Cause
Cardiac arrest	Infusing Opioid when patient's respiration is deteriorating	Opioid over-infusion	a. Bolus button pressed too frequent b. Incorrect pump calibration
Tissue or organ damage	Infusing air bubbles into the patient's blood stream	Air embolism	Continue infusion i) after inappropriate priming, ii) with empty reservoir, or iii) under tubing leakage

Table 1. *ISO 14971 Risk Analysis Concepts Applied to the PCA Pump (excerpts)*

the functional safety of the PCA Pump. The two primary hazards are opioid over-infusion and infusion of air bubbles into the patient blood stream – in both situations, serve consequences including death can be casued to the patient. Both of these hazards can have multiple initiating causes (excerpts are shown in Table 1), and our full models capture these along with associated mitigations. Due to space constraints in this paper, we limit detailed discussion to selected cases.

The right side of Figure 3 provides one such case in which opioid over-infusion has an initiating cause associated with the patient bolus button being pressed too frequently. One of the purposes of risk analysis is to identify causal relationships between the ultimate harm and potential causes – either working top-down (starting from the harm to the patient and hazardous situation at the boundary of the system and environment and working "down" through various notions of dependence in the system (control actions, information flows from sensors through system state and control laws, and other less direct notions such as interference between functions due to resource constraints)) or bottom up (starting from failures of components or communication, or events in the environment that may cause the system to move to an unsafe state). Looking at the Figure 3 from a bottom-up view, causality flows from events in the environment (i.e., the patient pushing the bolus botton repeatedly over a period of time) through button press detection, through the systems basic control logic which calculates pump flow rate, through commands to the pump motor, triggering increased opioid flow rates from the pump into the patient's blood vessels. In this scenario, the initiating cause is the repeated bolus button pushes over time. The system control logic commands the pump motor to increase flow rates, resulting in hazard (potential source of harm) of a high drug flow rate. The hazardous situation is the patient being exposed to a high drug flow over time, leading to a harm of cardiac arrest.

One of the goals of model-based hazard analysis is to enable top-down and bottom-up hazard analysis by capturing causality information within a system design / architecture and supporting automated tracing of causaility chains in forward (what future actions may be caused by the current action) and backward (which previous actions may cause the current action) manners. States and events (perhaps considered over time) in modeled causality chains need to be mapped to concepts such as initiating causes or hazardous situations. This type of automated analysis and associated reporting can support the broader risk management process.

In the ISO 14971 process (see [13] for a summary), the manufacturer identifies device characteristics related to safety, including potential notions of harm and hazards. Hazard analysis then identifies hazardous situations working top-down or bottom-up (or both) as summarized above. The risk of each hazardous situation is determined. In ISO 14971, *risk* is defined as the "combination of the probability of occurrence of harm and the severity of that harm". If risk reduction is needed to bring risk to an acceptable level, risk controls are designed, implemented, and verified. Risk controls can involve (a) removing the source of harm, (b) reducing the probability of occurrence of harm (either by the device controlling aspects of the system or environment) and/or severity of the

harm, or (c) detecting the occurence of a harm and calling for user intervention. In the hazardous situation of Figure 3, the hazard (related to the presence of opioid) cannot be eliminated since that is the source of therapy (pain relief). However, probability of occurrence and severity can be reduced via pump controls to limit the maximum dosage of opioid that the patient can receive over a period of time. In addition, detection of harm occurrence can include the simultaneous use of additional medical devices (e.g.pulse oximetry and capnography) that monitor patient vital signs.

Thus, in addition to capturing causality information with mappings to initiating cause, hazards, and hazardous situations, the broader risk management process benefits from model-based capture of device-based risk controls and indication of the scope of verification of risk-based controls. Note that risk controls of type (a) or (b) above are properties of the design design/function can be captured via "before" and "after controls" device models, whereas type (c) is a property of the broader environment (outside of the boundary of the device). Our focus in this paper is on types (a) and (b) (other forms of modeling including environment and clinical workflow modeling can address type (c) risk controls).

Lastly, the reporting feature of our proposed techniques can assit the residual risk evaluation step in the ISO 14971 process. However, determing whether the residual risk is within an accpetable level depends on clinical use of the subject devices and industry best practices, which is beyond the scope of this paper.

5 AADL EM Modeling for the OpenPCA System

As described in [2], the AADL EM annex document enables modeling of different types of faults, fault behavior of individual components, fault propagation across components in terms of peer-to-peer interactions and deployment relationships between software components and their execution platform, and aggregation and propagation of fault behavior across the component hierarchy.

AADL EM provides an *error type* facility to specify categories of errors and faults for a particular application or organization, each of which are represented as *error tokens* – values of error types. One main activity in AADL EM modeling is to specify error propagation rules within the model that describe how error tokens propagate from component inputs to outputs and along other model relationships (e.g., deployment bindings). The basic causality and dependence information captured in the AADL EM error propagation annotations can be used to support both forwards and backwards analyses of causality chains.

In the discussions that follow, we use the following role names to distinguish the activities of different stakeholders of the framework: *tool designer* refers to authors of this paper and associated steps necessary to configure and extend AADL for our described ISO 14971 framework; medical device *manufacturer* refers to associated steps taken to configure the framework for their organization, which may include multiple medical devices; *analyst* refers to activities associated with using the configured framework to support risk analysis for a particular medical device.

Using AADL EM involves a *layer* of concepts and annotations. On top of the architecture definitions (e.g., as presented Section 3), error flow/propagation annotations describe how errors propagate between component inputs and outputs (intra-component). In underlying tools, these are combined with model associations such as component connections and bindings to create a error flow graph. The elements that flow along the arcs in the graph are error tokens defined using the EM error token/type facility.

To design the framework, we used AADL's *property set* extensibility mechanism to add schemas for new properties capturing the ISO 14971 notions of harm, hazard,

hazardous situation, and initiating cause. We configured AADL's property association mechanism to allow the analyst to associate declarations of hazard, hazardous situation, and initating cause to various points in the architecture and to specific error tokens. The analyst uses the existing AADL EM error type mechanism to declare fault/error classifications appropropriate to the product. Further, the analyst uses the EM error propagation rules to indicate how error, faults, and their effects propagate through the system, according to their knowledge of the system's behavior and structure. Section 6 describes new analysis automation and reporting mechanisms that we have developed that aggregate all of these annotations into structure information that can be actively browsed, queried, and used to generate ISO 14971 aligned risk analysis reports with active elements that link directly to models and causality visualizations.

Preparing the ISO 14971 Framework – Tool Designer: Part of our effort to configure AADL EM for ISO 14971 involved defining schemas for related properties. The listing below shows the schema for the notion of harm. Schemas for Hazard, Hazardous Situation, and Cause are similar.

```
Harm: type record (
  ID: aadlstring; -- unique ID used as the primary reference to the harm
  Description: aadlstring; -- description used in report generation
  Severity: ISO14971_80001::SeverityScales; -- associate pre-decl. severities
);
```

AADL EM comes with a standard error type library that captures many of the notions in the fault taxonomy of [4]. For the artifacts described in this paper, we configured a simplified type library that is sufficiently general for supporting medical device risk analysis activities. Device manufacturers can further specialize the library to introduce notions of fault and error specific to their products. As an example of such customization, previous work from our research group created specializations for supporting risk analysis for interoperability and security related issues [24].

Instantiating the ISO 14971 Framework – Device Manufacturer: The device manufacturer may configure the framework for their general risk management process. This includes defining qualitative categories for severity and frequency (e.g., choosing among options listed in ISO 14971 2009 annexes and supporting technical reports, extending the AADL error library to reflect taxonomies of faults and hazards used within the manufacturer's risk management process.

Identifying Harms, Hazards, and Hazardous Situations for a Device – Analyst: Drawing on information gathered from the ISO 14971 process steps of "gathering characteristic related to safety" and "identifying hazards" (clauses 5.3 and 5.4) the analyst introduces model annotations to capture harms and hazards. The discovery of harms/hazards cannot be supported to any significant degree by automated tooling, but instead relies on domain knowledge, experience, clinical trials, and medical domain literature to identify relevant harms and hazards. US FDA Guidance documents are often a good source for this type of information. The list below illustrates the definition of an over-infusion hazard and an associated harm of respiratory depression. In addition to identifiers used in reporting, the harm specification classes severity of this harm as catastrophic (essentially uses an enumerated type value from a set of qualitative severity values configured by the manufacturer). Other harms/hazards (omitted here) that we captured for the PCA Pump related to functional safety include the air embolism and under-infusion as presented in Section 4.

a. Harm and Hazard instance

```
--Harm
H1: constant ISO14971_80001::Harm => [
  ID => "H1";
  Description => "Respiratory Depression";
  Severity => Catastrophic;
];

--Hazards
Haz1: constant ISO14971_80001::Hazard => [
  ID => "Haz1";
  Description => "Drug over-infusion";
];
```

b. Cause Instance

```
FrequentButtonPress : constant
  ISO14971_80001::Cause => [
   ID => "FrequentButtonPress";
   Description => "";
   Probability =>  Frequent;
];
```

Similarly, the analyst introduces property that will label events/state in the device or environment that represent an initial step in a causality chain. The example below illustrates the introduction of a reporting label for a frequent bolus botton press (one of the root causes of a potential over-infusion hazard).

Note that additional causes may also be discovered and added in the process of the analysis (e.g., applying the tools of Section 6).

To configure the error propagation layer, based on their domain knowledge, analysts introduce EM error types representing different types of root causes and observable problematic device behaviors that may contribute to harms. The listing below shows excerpts that define error types for problematic environment actions that (without mitigation) may cause harms as well as observable device behaviors that may lead to harms.

```
  error types --

  -- Example error types associated with causes in the environment
  DrugKindError : type; --wrong drug is loaded into reservoir
  TooSoonPress : type;  -- bolus button pressed too soon/often
  ThirdPartyPress : type; --when someone other than the patient presses the button
     ...

   -- Example error types associated with hazards (as observable at
   --  the device boundary)
  AirEmbolism : type; -- air bubble in fluid emitted from device
  DrugOverInfusion : type; -- too much drug, possibly harmful
  DrugUnderInfusion : type; -- too little drug, may not be enough to reduce pain
**}
```

Now the analyst uses AADL EM annotations to connect the layers in the framework – linking the elements above to the architecture description. Such annotations are added throughout the architecture, but an especially important step is the treatment of the system boundary to reflect both environmental causes of hazards (typically associated with device inputs) and observable device behaviors that may lead to harm (typically associated with device outputs). The listing below shows excerpts of the system boundary model – focusing on annotations that address frequent bolus request / over-infusion.

```
system PCA_Pump_System extends Platform::Generic_System
 features
  -- feature group collecting sensor inputs
  sense: refined to feature group iPCA_Feature_Groups::Sensing_iPCA;
  -- feature group collecting observable actions on environment
  act: refined to feature group iPCA_Feature_Groups::Actuation_iPCA;
  -- representation of other inputs, e.g., operator supplied info (excerpts)
  fill_drug: in data port Physical_Types::Fluid_Volume;
 properties

  ISO14971_80001::SystemInfo => [
   Name => "Open Pca Pump";
   Description => "Patient controlled analgesic infusion pump";
   IntendedUse => "Infuse safe levels of opioid into the patient for pain management";
  ];

  ISO14971_80001::Hazardous_Situations => (HazardousSituations::OverInfusion,
    HazardousSituations::UnderInfusion, HazardousSituations::IncorrectDrug);

  annex EMV2 {**
   use types iPCA_Error_Model, ErrorLibrary; -- indicate error types to be used
   error propagations
    -- drug output may be incorrect flow rate, or wrong kind of drug
```

```
    act.drug_outlet: out propagation {DrugStopped, DrugOverInfusion,
                                      DrugUnderInfusion, DrugKindError};
    -- the reservoir may be filled with the wrong kind of drug
    fill_drug: in propagation {DrugKindError};
    -- someone other than the patient presses the button
    sense.patient_button_press: in propagation {TooSoonPress, ThirdPartyPress};
    -- signal to barcode reader may be corrupted
    sense.barcode_signal: in propagation {ValueError};
    -- clinician may enter data incorrectly
    sense.ui_touch: in propagation {OperatorError};
  end propagations;

  properties
    ISO14971_80001::causes => (Causes::FrequentButtonPress)
                              applies to sense.patient_button_press.TooSoonPress;
    ISO14971_80001::causes => (Causes::IncorrectDrug)
                              applies to fill_drug.DrugKindError;
    ISO14971_80001::Hazards => (Hazards::Haz1)
                              applies to act.drug_outlet.DrugOverInfusion;
    ISO14971_80001::Hazards => (Hazards::Haz2)
                              applies to act.drug_outlet.DrugUnderInfusion;
    ISO14971_80001::Hazards => (Hazards::Haz3)
                              applies to act.drug_outlet.DrugKindError;
  **};
end PCA_Pump_System;
```

In particular, on the `patient_button_press` sensor input, an EM flow annotation of `ButtonError` models button presses that occur too often. The AADL EM `applies` construct associates the `Cause::FrequentButtonPress` cause with the `ButtonError` flow token, which has the effect of linking the error token (and flows proceeding) from the token to the reporting framework as a possible cause of (and causality chain leading to) a hazard. Similarly, the `DrugOverInfusionToken` is associated with `drug_outlet` output, and then associates flow leading into that token as well as the token itself with the `Haz1` annotation which is understood by the reporting framework.

Using the analysis framework to identify Sequences of Events – Analyst: ISO 14971 Clause 5.4 states that "For each identified hazard, the manufacturer shall consider the reasonably foreseeable sequences or combinations of events that can result in a hazardous situation, and shall identify and document the resulting hazardous situation(s)." To support this requirement, the analyst adds flow annotations to components throughout the architecture to model causality paths and then uses the analysis capabilities in Section 6 to compute various forms of reachability and report generation.

The fragments in the listing below illustrates how flow annotations are added to capture error propogations indicating that a component (a) may be a *source* of an error, (b) may propagate errors (and possibly transform the type of error), and (c) may *sink* an error (i.e., serve as a mitigation for an error).

```
-- In mechanical pump
calibration_over : error source drug_outlet{DrugOverInfusion};
mp_err: error path drug_intake{DrugKindError} -> drug_outlet{DrugKindError};
over: error path bindings {HighValue} -> drug_outlet{DrugOverInfusion};
...
-- In Patient bolus checker
pbc: error sink patient_button_request {TooSoonPress, ThirdPartyPress};
```

In the component for the mechanical pump which takes actuation commands from the control logic (including setting the flow rate), the first line in the listing models the fact that a lack of calibration of the pump itself could cause fluid to be moved out of the `drug_outlet` port at a rate that exceeds the pump's specification, resulting in an drug over-infusion error (a corresponding under-infusion error is omitted). The second line models a situation where the wrong drug enters the `drug_intake` port (intuitively, because the nurse has entered a vial in the drug reservoir with the wrong drug) – in this case, the error propagates from the input to the output (i.e., the wrong drug flows through mechanical pump). The third line models a situation where the control logic has command a flow rate that is too high: the `HighValue` error is transformed to a `DrugOverInfusion` error indicating that the bad command causes a problematic high flow out of the mechanical pump. The final line models the patient bolus checker component that (partially) mitigates errors related to the bolus button being pushed

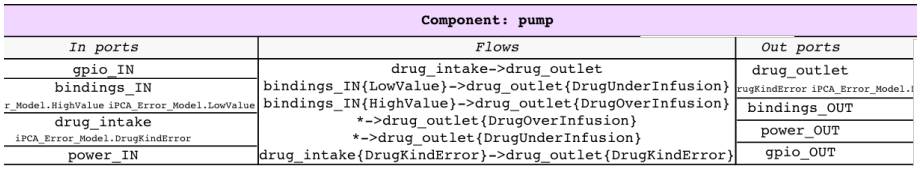


Fig. 4. Awas AADL Intra-component Error Flows Visualization

too soon or by a third party by limiting the number of active button pushes over a time period. In this case, the component acts as a sink for the errors.

As flows are explored using the tools in Section 6, the analyst is interested in understanding the relationships between causes, hazards, and harms. A hazardous situation describes relationships between a hazard and a harm. The analyst records a hazardous situation by introducing a model property such as in the listing below. The hazardous situation instance below describes a scenario in which the Haz1 leads to the harm H1. During the analysis, the causality relationship between hazards and initiating causes are computed. Hence, providing a complete scenario of error flow from initiating cause to hazards to hazardous situation and finally leading to harms (this is reflected in the 14971 reports described in the following section).

```
-- Hazardous Situation
OverInfusion : constant ISO14971_80001::Hazardous_Situation => [
 ID => "OverInfusion";
 Description => "Infusing drug when the patient's health is deteriorating";
 Hazard => Hazards::Haz1;
 Paths_to_Harm => ([
  Harm => Harms::H1;
  Contributing_Factors => (ContributingFactors::HealthDeteriorating);
  Probability_of_Transition => Remote;
 ]);
 Risk => High;
 Probability => Remote;
];
```

6 AADL EM Analysis Support

A variety of analyses can leverage the error flow and ISO 14971 property annotations in the previous section. The OSATE AADL EM plug-in provides several different forms of safety analysis including fault tree analysis and a simple functional hazard analysis. In this section, we illustrate Awas [5] which complements these existing analysis with a scaleable interactive visualizations and queries of error flows. In the ISO 14971 context, these capabilities are applied to automated discover and visualize potential "sequence of events" leading from causes to hazards, to hazardous situations, to harms.

Awas builds *component* and *system* visualizations that are tailored to illustrating *flow-related* aspects. Figure 4 illustrates how Awas builds a component-level summary of flow properties that show component inputs (left side), outputs (right side), and the error flow rules (middle) that the analyst has specified to capture how error tokens propagate from inputs to outputs.

Awas builds a dependence graph composed from intra-component flows (as in Figure 4) together with several forms of inter-component dependences including port connections, component bindings, etc. The flow graph representation and analysis algorithms are written in Scala and compiled to Javascript using the Scala.js framework[5]. This generates a highly navigable, dynamic visualization of flows integrated across all levels of the system hierarchy. The most basic capability is forward/backward reachability analysis. Analysts simply click on a component or port and press a button to carry out basic queries such as "where in the system do the modeled errors (and their

[5] www.scalajs.org

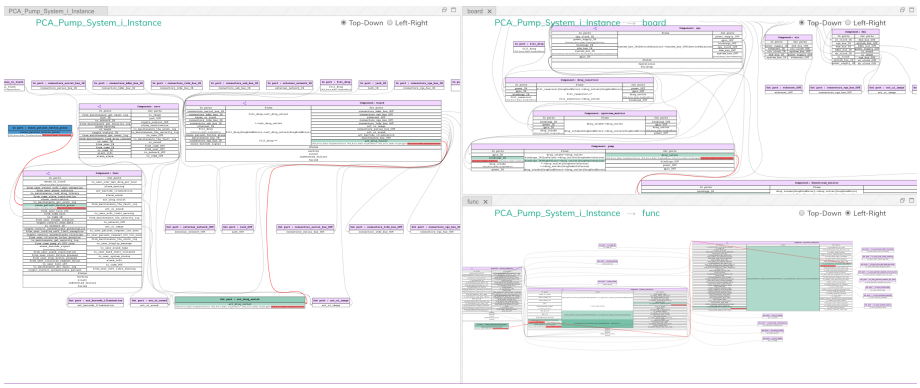


Fig. 5. Awas AADL System-wide Error Flow Visualization (selected sub-systems)

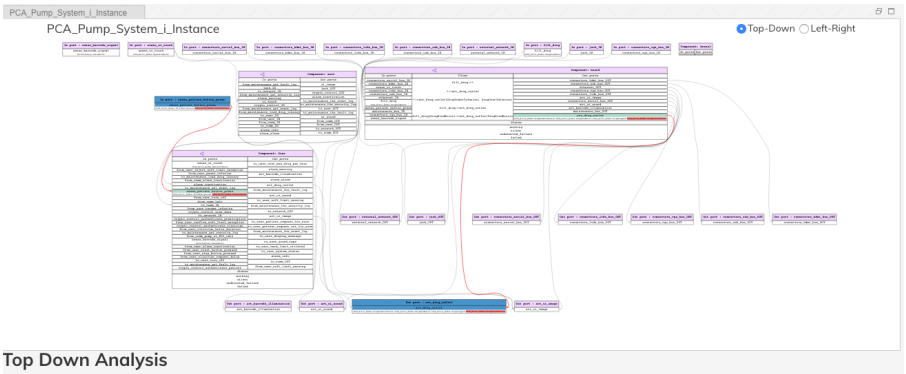
subsequent impacts) from this port/component flow?" or "what system elements are contributing errors that flows into this port/component?".

In the example of Figure 5 [6], the analyst clicks on the system boundary `sense.patient_button_press` input port with an error token indicating a possible "too frequent" bolus button push and presses the Forward analysis button to have the tool discover and mark up the where in the architecture the effects of this error may propagate (paths are shown in red, and components and ports along the path are shown in green and red). The Open PCA architecture includes approximately 19 sub-systems/component levels of hierarchy. Using the window-tiling capability of Awas, Figure 5 shows three such subsystems opened (system top-level, a portion of the functional architecture, and lower-level hardward resources). Behind the scenes, the reachability information is computed almost instantaneously across the entire system, A simple scroll of a mouse wheel zooms into a particular system section or component of interest. Double-clicking on components drills down to their subcomponent models. Projections of the system can be performed on components/flows of user-specified categories, or components along user-specified paths.

This supports expected ISO 14971 workflows as follows. Working in either a bottom up manner (from causes to hazardous situations) or top-down manner (from hazardous situations to causes), the analyst uses both forward and backward Awas reachability to discover causality chains in the error-flow annotated architecture. Annotations marking causes and hazardous situations are incrementally added to the model as important aspects of error propagations are revealed in Awas. The web-site supporting this submission illustrates further capabilities in the browser-based deployment of Awas (no tool installation needed) including the ability to define and save more sophisticated queries written in a form of path logic. As the analyst discovers error propagations and begins to annotate the architecture for mitigation strategies, this enables common queries corresponding to hazardous situations to be replayed as mitigations are added to confirm that impacts of causes are eliminated or reduced.

On top of the general Awas cabilities, we have developed a reporting tool that produces information in the formats suggested by ISO 14971 and associated medical domain risk management guidance. Figure 6 illustrates an excerpt of this report that captures the association between hazardous situations and related concepts. This information is automatically extracted from the model based on the model annotations

[6] Note that the purpose of these screenshots is to illustrate application of the Awas tools at scale (capturing system-wide browsing across a large system with many complex components). The screen captures of the tool cannot capture both the scalability aspect while preserving the readability of the component/port/details, etc. In the Awas tool, mouse scrolling easily zooms in and out to reveal component details.



Top Down Analysis

Hazards	Contributing Factors	Initiating Causes	Hazardous Situation	Harm
Haz1	PatientHealthDeteriorating	PumpIncorrectCalibration FrequentButtonPress	OverInfusion	H1
Haz2	PatientInGoodHealth	PumpIncorrectCalibration	UnderInfusion	H2
Haz3	ClinicianFailedToCheck	IncorrectDrug	IncorrectDrug	H3

Fig. 6. Awas ISO 14971 Report (excerpts) illustrating Sequence of Events Leading to Hazardous Situation

of Section 5 and the Awas reachability analysis. Both PDF and HTML versions of the report are produced. The HTML report (Figure 6) is "animated" in the sense that one can highlight a certain hazardous situation (the selection is shown in blue), the Awas visualization for the causality chain from cause through hazardous situation to harm is automatically computed and displayed in the report, corresponding to the ISO 14971 requirement that the analyst uncover "series of events" (see Figure 3) along the causality pathway). Figure 6 shows excerpts capturing only a portion of information related to the over-infusion hazard. The website artifacts show a much expanded report capturing a number of other hazardous situations.

7 Related Work

The development and application of MBSA techniques have been widely studied in other safety-critical domains such as avionics and automotive, with the goals to support system-level safety assurance and to help manufacturers better comply with the mandatory safety standards (such as ISO 26262 [15]). These techniques combine safety analysis techniques (such as HiP-HOPS [22]) with system design models specified in languages like AADL, SysML, and Mathworks' Simulink, to provide the developers better visibility into how design changes or errors within the system model propagate across the system and give rise to risks. Extensive review of MBTA techniques in these domains can be found in [3, 25].

MBSA techniques based on AADL EM, like [10, 7] and the one presented in this paper, follow the same safety analysis paradigm. However, AADL EM provides strong annotation support to specifying the error behavior and propagation rules for system components, and includes a rich set of built-in errors (while developers can also define their own error types). These unique features of AADL EM enable efficient semi-automatic risk analysis for AADL system models.

Research and industrial practices of applying MBSA to the medical device domain are still limited. The Generic Infusion Pump (GIP) project [26] is a collaborative effort between the FDA and multiple universities to demonstrate the application of MBE methods to an open-source generic infusion pump design. Risk analysis for the GIP, however, was performed as manual application of hazard analysis techniques to the GIP system model. There have been few other studies in applying hazard analysis techniques, such as System Theoretic Process Analysis (STPA) [6, 20] and HAZOP (Hazard Operability) [11], to different types of medical devices. Similar to the GIP

project, these efforts mainly depended on manual risk analysis, without making risk management an integral part of the MBE toolchain.

The work presented in this paper continues our previous effort [17, 23] in extending the AADL toolset to support hazard analysis in medical device development, and illustrates how AADL EM can be leveraged to support each risk management activity throughout the ISO 14971 process. Risk control measures formulated with the help of AADL EM can be passed to downstream development activities for implementation (e.g., through automatic code generation) and verification.

8 Conclusion

We have provided an overview of how AADL EM can be used to support MBSA for medical devices in alignment with the ISO 14971 standard. Since past work on AADL EM have primarily targetted the avionics domain, this further illustrates the potential usefulness of AADL EM via demonstration on a large example with steps mapped to safety-standard terminology, workflows, and reporting in a different domain. For medical device manufacturers and regulators that may be unfamiliar with any form of MBSA, this work illustrates how an architecture-integrated MBSA may be carried out with a broader collection of artifacts that are part of the Open PCA project.

We have also demonstrated how the Awas AADL dependence analysis and visualization tool can be applied to support specific steps in AADL safety analysis and how reporting capabilities can be developed to support the ISO 14971 risk management process. We believe that the visualizations and error flow browsing capabilities can provide multiple practical benefits to practitioners working on full-scale systems.

Extensions of this work can address deeper integration of the Awas-based analysis with other AADL-based MBSA tools that support automated generation of fault trees, FMEA-like analyses, and model-based versions of STPA [23]. Incorporation of model-checking techniques, including probabalistic techniques can further enchance the precision of the analyses and provide quantitative risk evaluations.

References

1. Architecture Analysis & Design Language (AADL) (Rev. C). Aerospace Standard AS5506C (2017)
2. Architecture analysis and design language (AADL) annex volume 1: Annex e: Error model annex (2015)
3. Aizpurua, J.I., Muxika, E.: Model-based design of dependable systems: limitations and evolution of analysis and verification approaches. International Journal on Advances in Security **6**(1&2), 12–31 (2013)
4. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**(1), 11–33 (2004)
5. Sireum Awas web site. https://awas.sireum.org (2018)
6. Blandine, A.: Systems Theoretic Hazard Analysis (STPA) applied to the risk review of complex systems: an example from the medical device industry. PhD dissertation, Massachusetts Institute of Technology (2013)
7. Brunel, J., Feiler, P., Hugues, J., Lewis, B., Prosvirnova, T., Seguin, C., Wrage, L.: Performing safety analyses with aadl and altarica. In: Proceedings of 4th International Symposium on Model-Based Safety and Assessment. pp. 67–81 (2017)
8. Carpenter, T., Hatcliff, J., Vasserman, E.Y.: A reference separation architecture for mixed-criticality medical and iot devices. In: Proceedings of the ACM Workshop on the Internet of Safe Things (SafeThings). ACM (November 2017)
9. Center for Devices and Radiological Health: Infusion Pumps Total Product Life Cycle–Guidance for Industry and FDA Staff, institution = US Food and Drug Administration. Tech. Rep. FDA-2010-D-0194 (2014)

10. Delange, J., Feiler, P.: Architecture fault modeling with the aadl error-model annex. In: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications. pp. 361–368 (2014)
11. Guiochet, J., Hoang, Q.A.D., Kaaniche, M., Powell, D.: Model-based safety analysis of human-robot interactions: The miras walking assistance robot. In: Proceedings of IEEE 13th International Conference on Rehabilitation Robotics (ICORR). pp. 1–7 (2013)
12. Hatcliff, J., Larson, B., Carpenter, T., Jones, P., Zhang, Y., Jorgens, J.: The Open PCA Pump Project: An exemplar open source medical device as a community resource. SIGBED Rev. p. 8–13 (Aug 2019)
13. Hatcliff, J., Vasserman, E.Y., Carpenter, T., Whillock, R.: Challenges of distributed risk management for medical application platforms. In: 2018 IEEE Symposium on Product Compliance Engineering (ISPCE). pp. 1–14 (May 2018)
14. Hatcliff, J., Larson, B.R., Belt, J., Robby, Zhang, Y.: A unified approach for modeling, developing, and assuring critical systems. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Modeling. pp. 225–245. Springer International Publishing, Cham (2018)
15. ISO: ISO 2626201:2018 Road vehicles — Functional safety — Part 1: Vocabulary (2018)
16. ISO: ISO 14971:2019(E) Medical devices – Application of risk management to medical devices (2019)
17. Larson, B., Hatcliff, J., Fowler, K., Delange, J.: 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. pp. 65–84. HILT '13, ACM, New York, NY (2013)
18. Larson, B., Chalin, P., Hatcliff, J.: 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, pp. 276–290. Springer (2013)
19. Larson, B.R., Jones, P., Zhang, Y., Hatcliff, J.: Principles and benefits of explicitly designed medical device safety architecture. Biomedical Instrumentation & Technology **51**(5), 380–389 (2017)
20. Masci, P., Zhang, Y., Jones, P., Campos, J.C.: A hazard analysis method for systematic identification of safety requirements for user interface software in medical devices. In: Proceedings of 15th International Conference on Software Engineering and Formal Methods. pp. 284–299 (2017)
21. Open PCA Pump Project web site. http://openpcapump.santoslab.org (2018)
22. Papadopoulos, Y., McDermid, J.A.: Hierarchically performed hazard origin and propagation studies. In: Felici, M., Kanoun, K. (eds.) Computer Safety, Reliability and Security, pp. 139–152. Springer (1999)
23. Procter, S., Hatcliff, J.: An architecturally-integrated, systems-based hazard analysis for medical applications. In: 2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE). pp. 124–133 (Oct 2014)
24. Procter, S., Vasserman, E.Y., Hatcliff, J.: Safe and secure: Deeply integrating security in a new hazard analysis. In: Proceedings of ASSURE 2018: the 6th international workshop on assurance cases for software-intensive systems. pp. 1–10 (Sep 2018)
25. Sharvia, S., Kabir, S., Walker, M., Papadopulos, Y.: Model-based dependability analysis: state-of-the-art, challenges, and future outlook. In: Mistrik, I., SOley, R., Ali, N., Grundy, J., Teknerdogan, B. (eds.) Software Quality Assurance, p. 251–278. Morgan Kaufmann (2016)
26. University of Pennsylvania Real Time Systems Group: The Generic Infusion Pump (GIP), http://rtg.cis.upenn.edu/gip.php3
27. US Food and Drug Administration: Examples of Reported Infusion Pump Problems, https://www.fda.gov/medical-devices/infusion-pumps/examples-reported-infusion-pump-problems
28. US Food and Drug Administration: Infusion Pump Improvement Initiative, https://www.fda.gov/medical-devices/infusion-pumps/infusion-pump-improvement-initiative