Towards An AADL-Based Definition of App Architecture for MAPs

http://santoslab.org/pub/mdcf-architect

Sam Procter, John Hatcliff, and Robby SAnToS Lab
Kansas State University

Support:
This work is supported in part by the US National Science Foundation (NSF) (#1239543), the NSF US Food and Drug Administration Scholar-in-Residence Program (#1355778) and the National Institutes of Health / NIBIB Quantum Program.
Health Care Involves
A Variety of System Components

Clinical Protocols
Actuators
Sensor Data Displays
Sensor
Displays
Clinicians
Information Systems
Patient
Motivation

- What are the types of things we could do with device integration?
  - Information forwarding
  - Automation of clinical workflows
  - Closed loop control between devices
- Unlike personal computing, medical devices are not designed to work together
- Integrating medical devices would bring myriad benefits
- ... how can we do so safely?
Outline

- Background
  - PCA Interlock Scenario
  - Medical Application Platforms
  - AADL
- Vision
- Language
- Tool
- Future
Status Quo: MDDS

Medical Device Data Systems – Data only flows from producers to consumers; data must be faithfully re-presented
Patients are commonly given patient-controlled analgesics after surgery.

Crucial to care, but numerous issues related to safety.

Data for disabling the pump exists now (just a system invariant) -- we just need to integrate it.
Clinically Supported

Motivating Clinical Problem: PCA Overdose

“A particularly attractive feature may be the ability to automatically terminate or reduce PCA (or PCEA) infusions when monitoring technology suggests the presence of opioid-induced respiratory depression. To facilitate such capabilities, we strongly endorse the efforts to develop international standards for device interoperability and device-device communication.

It is critical that any monitoring system be linked to a reliable process to summon a competent health care professional to the patient's bedside in a timely manner. “
PCA Pump Safety Interlock

Fully leverage device data streams and the ability to control devices

**Devices**

- **PCA Pump**
  - Enable Pump for safe time window
- **Capnograph**
  - Monitoring Data + Alarm Information
- **Pulse Oximeter**
  - Monitoring Data + Alarm Information

**Combined PCA Vitals Monitoring**

- Device Task controller
- Enable bolus dose only when ticket present
- PCA Bolus “Enable” Ticket
- Aggregated Monitoring Status

**Clinician / Monitoring**

**Status Display for PCA Monitoring Application**
A *Medical Application Platform* is a safety- and security-critical real-time computing platform for...

- Integrating heterogeneous devices, medical IT systems, and information displays via communications infrastructure, and
- Hosting applications ("apps") that provide medical utility via the ability to acquire information from and update/control integrated devices, IT systems, and displays
Background

PCA Pump Interlock Architecture

Medical Application Platform

SUI App Display

View Display

Data for Display

App

Start / Stop Commands

Sensor + Alarm Data

Pulse Oximeter, Capnograph, and Patient Controlled Analgesia Pump

PCA

PR

SPO₂

ETCO₂

RR

Data should arrive once per second

View Display

Configuration, Alarm Clear

Attach Sensors

Clinician (App Administrator)
Background

Architecture Analysis and Description Language (AADL)

- SAE Standard, used in e.g., Avionics
- Enables model-driven, component-based development of
  - Software
  - Hardware
  - And the bindings between the two
- Previously applied to a single medical device, what about a system of multiple medical devices?
- How well can it work on a managed platform?
- Can we do anything beyond describing an app’s architecture with it?
Outline

- Background
- Vision
  - Analyses
  - Code generation
- Language
- Tool
- Future
Vision

Analyses and Regulatory Artifacts

Clinical Use Case / Workflow Description

Requirements

Hazard Analysis

Risk Assessment

App Developer

Medical Device Coordination Framework

FDA 510K Submission Package

3rd Party ICE Conformance & Safety Certification Submission Package

Assurance Case

3rd Party Certifiers

FDA Evaluators

App Deployment
Vision

Code Generation

A. The app’s architecture is specified in AADL
   1. Components as AADL Devices / Processes
   2. Connections are specified
   3. RT/QoS Parameters are via AADL’s property-specification mechanism

B. The app is programmatically translated to Java and XML

C. The app is launched on a compatible MAP
Outline

- Background
- Vision
- Language
  - Why MDD?
  - Why (a subset of) AADL?
  - Constructs
- Tool
- Future
MAP Characteristics

MAP constituted device instances are variable – the constituents that form the MAP constituted device may differ on different invocations of the device.

- Same app, and thus same conceptual “system”
- Just one architecture and development framework
- But, different component instances.
MAP Characteristics

MAP constituted device instances are variable – the constituents that form the MAP constituted device may different on different invocations of the device.

- Same app, and thus same conceptual “system”
- Just one architecture and development framework
- But, different component instances.
Why use AADL?

- History of successful safety-critical projects
  - Avionics / Boeing (SAVI): “integrate-then-build” approach
- Previously found that MAPs lend themselves to pub-sub
  - Device as publisher, apps as subscriber
  - Natural to model with AADL’s port connections
- Annexes support a number of regulatory and verification artifacts
  - Hazard Analysis (EMV2), Interface contracts (BLESS), etc.
Language

Why subset AADL?

- AADL is targeted at co-design, ie: complete systems
  - MAPs are managed platforms

- Semantic mismatches
  - Processes

- Insufficiency of pre-declared properties

- Unrealizable communication patterns
  - No shared-memory access in pub/sub middleware
Output rate: 1 sec .. 5 sec

Channel Delay: 50ms

Period: 50ms
WCET: 5ms
Language

System

Medical Devices

Software Components

Communication links between components

... and properties of those links!
Language

System

package PCA_Shutoff
public
with PulseOx_Interface, PCAPump_Interface, PCA_Shutoff_Legic,
PCAShutoff_Properties, MAP_Error_Properties, PCA_Shutoff_Display,
PCAShutoff_Errors, Capnograph_Interface, MAP_Errors,
PCAShutoff_Error_Properties;

system PCA_Shutoff_System
end PCA_Shutoff_System;

system implementation PCA_Shutoff_System.imp
subcomponents
  -- Physiological inputs
capnograph : device Capnograph_Interface::ICEcapnographInterface.imp;

  -- App Logic
appLogic : process PCA_Shutoff_Legic::ICEpcaShutoffProcess.imp;
appDisplay : process PCA_Shutoff_Display::ICEpcaDisplayProcess.imp;

  -- Controlled device
pcaPump : device PCAPump_Interface::ICEpcaInterface.imp;
connections
  -- From components to logic
respiratoryrate_logic : port capnograph.RespiratoryRate -> appLogic.RespiratoryRate;
pumpcommand_logic : port appLogic.CommandPumpNormal -> pcaPump.PumpNormally;
etco2_logic : port capnograph.ETCO2 -> appLogic.ETCO2
{MAP_Properties::Channel_Delay => 50 ms};

  -- From components to display
pumpcommand_displ : port appLogic.CommandPumpNormal -> appDisplay;
end PCA_Shutoff_System.imp;
end PCA_Shutoff;
Language

Device Interface Specification

Device API Only -- Presents the app’s view of the required device capabilities, not the full device capabilities.
Device Interface Specification

Device API Only -- Presents the app’s view of the required device capabilities, not the full device capabilities

```
package Capnograph_Interface
public
with PCA_Shutoff_Types;
  device ICEcapnographInterface
    features
      ETCO2 : out event data port PCA_Shutoff_Types::ETCO2;
      RespiratoryRate : out event data port PCA_Shutoff_Types::RespiratoryRate;
    end ICEcapnographInterface;
  device implementation ICEcapnographInterface.imp
    end ICEcapnographInterface.imp;
  end Capnograph_Interface;

package PCAPump_Interface
public
with PCA_Shutoff_Types;
  device ICEpcaInterface
    features
      PumpNormally : in event data port PCA_Shutoff_Types::PumpNormalCommand
        {MAP_Properties::Output_Rate => 50 ms .. 75 ms};
    end ICEpcaInterface;
  device implementation ICEpcaInterface.imp
    end ICEpcaInterface.imp;
  end PCAPump_Interface;
```
Language

Process Specification

External ports

Tasks (Threads)

Connections between external ports and threads
Language

Process Specification

```plaintext
process ICEpcaShutoffProcess
features
  - No ETCO2 thread / connection because it's a data port
  ETCO2 : in data port PCA_Shutoff_Types::ETCO2;
  RespiratoryRate : in event data port PCA_Shutoff_Types::RespiratoryRate;
  CommandPumpNormal : out event data port PCA_Shutoff_Types::PumpNormalCommand;
properties
  MAP_Properties::Component_Type => logic;
end ICEpcaShutoffProcess;

process implementation ICEpcaShutoffProcess.imp
subcomponents
  UpdateRespiratoryRateThread : thread UpdateRespiratoryRateThread.imp;
  PumpControlThread : thread PumpControlThread.imp;
connections
  incoming_rr : port RespiratoryRate -> UpdateRespiratoryRateThread.RespiratoryRate;
  outgoing_pump_command : port PumpControlThread.PumpNormal -> CommandPumpNormal;
end ICEpcaShutoffProcess.imp;
```
Language

Thread Specification

External ports

Properties
Language
Thread Specification

```plaintext
thread UpdateRespiratoryRateThread
features
  RespiratoryRate : in event data port PCA_Shutoff_Types::RespiratoryRate;
properties
  Timing_Properties::Deadline => 75 ms;
  Timing_Properties::Period => 95 ms;
  MAP_Properties::Worst_Case_Execution_Time => 7 ms;
  Thread_Properties::Dispatch_Protocol => Sporadic;
end UpdateRespiratoryRateThread;

thread implementation UpdateRespiratoryRateThread.imp
end UpdateRespiratoryRateThread.imp;

thread PumpControlThread
features
  PumpNormal : out event data port PCA_Shutoff_Types::PumpNormalCommand;
properties
  Timing_Properties::Deadline => 50 ms;
  Timing_Properties::Period => 105 ms;
  MAP_Properties::Worst_Case_Execution_Time => 10 ms;
  Thread_Properties::Dispatch_Protocol => Periodic;
end PumpControlThread;

thread implementation PumpControlThread.imp
end PumpControlThread.imp;
```
Component Development

- Development of component architecture using AADL / OSATE2
- Automatic generation of component architecture (skeletons)
- Automatic generation of component layout and app topology (configuration)
- Development of core behavioral code (business logic) using IDE of choice
- Translator can be retargeted to other languages as desired
Language Subset

AADL Constructs Used

<table>
<thead>
<tr>
<th>AADL Construct</th>
<th>MAP Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Components</strong></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>Layout</td>
</tr>
<tr>
<td>Device</td>
<td>Medical Device API for App</td>
</tr>
<tr>
<td>Process</td>
<td>Software Component</td>
</tr>
<tr>
<td>Thread</td>
<td>Task</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td></td>
</tr>
<tr>
<td>System-level port connection</td>
<td>Channel</td>
</tr>
<tr>
<td>Process-level port connection</td>
<td>Task Trigger</td>
</tr>
<tr>
<td>Process implementation-level port connection</td>
<td>Task-Port Communication</td>
</tr>
</tbody>
</table>
Language

Translation Target

- Display.compsig.xml
- Logic.compsig.xml
- Logic.java
- LogicSuperType.java
- Logic.compsig.xml (QoS/RT)
- Task1
- Task2
- Task3
- System.cfg.xml
- Display.java
- DisplaySuperType.java
- Display.compsig.xml (QoS/RT)
- Dev1.java
- Dev2.java
Outline

- Background
- Vision
- Language
- Tool
  - OSATE2
  - Availability
- Future
Tool

OSATE2

- Open-source, Eclipse-based tool
- Our work is available as a plugin
  - Uses the model-traversal built into OSATE2
Tool

OSATE2
Tool

OSATE2
Source available online at http://github.com/aadl-translator
Installable into OSATE2 via update site: http://santoslab.org/pub/mdcf-architect/updatesite
Full documentation online at http://santoslab.org/pub/mdcf-architect
Outline

- Background
- Vision
- Language
- Tool
- Future
  - Hazard Analysis
  - Tool Extensions
Future

Hazard Analysis and Risk Assessment

Clinical Use Case / Workflow Description

Requirements

Hazard Analysis

Risk Assessment

App Developer

Assurance Case

3rd Party ICE Conformance & Safety Certification Submission Package

MDCF

FDA 510K Submission Package

FDA Evaluators

3rd Party Certifiers

FDA Evaluators

App Deployment
Future

Tool extensions

- Abstraction Depth
  - Model methods / functions
- Data Types
  - CORBA IDL
- MAP Device Drivers
- Assurance / Safety Cases
Towards An AADL-Based Definition of App Architecture for MAPs

http://santoslab.org/pub/mdcf-architect

Sam Procter, John Hatcliff, and Robby SAnToS Lab
Kansas State University

Support:
This work is supported in part by the US National Science Foundation (NSF) (#1239543), the NSF US Food and Drug Administration Scholar-in-Residence Program (#1355778) and the National Institutes of Health / NIBIB Quantum Program.