Towards a Logical Foundation for Assurance
Arguments for Plug & Play Systems

Lu Feng, Andrew King, Insup Lee, Oleg Sokolsky

PRECISE Center
School of Engineering and Applied Science
University of Pennsylvania

VeriSure: Verification and Assurance Workshop at CAV 2015
San Francisco, 18 July 2015
Medical Device Interoperability

**Problem:** little to no integration of devices with each other

- Humans must automate even simple clinical workflows
- Unnecessary burdens placed on human caregivers
- Few opportunities for “sensor fusion” (better alarms and diagnostics)
Clinical Scenario: Laser Surgery / Ventilator

• Doctors enforce the following invariant
  – If laser = on then oxygen = off
  – If patient’s SpO2 < 95 then oxygen = on

• Systems of Systems approach
  – Let devices communicate and automate safety invariant enforcement
Benefits of Medical Device Interoperability

- Interoperable medical devices can self-coordinate
  - Provide continuous monitoring
  - Handle routine tasks and respond to obvious problems
  - Alert caregivers in more serious cases; reduce alarm fatigue
  - Physiological closed-loop control in many cases
Medical Device Plug-and-Play Open Systems

• Medical Device Plug-and-Play (MD PnP)
  – Interoperable medical devices based on plug-and-play
  – Vender neutrality based on open medical device interfaces
  – www.mdpn.org

• Emerging Interoperability Standards
  – ASTM Standard F2761–2009 for Integrated Clinical Environment (ICE) defines a high-level architecture and functional concept
  – The ICE architecture standard is the focal point for FDA’s evaluation of MAP (Medical App Platform) concepts in future medical systems
ICE Architecture

Supervisor

ICE App Code Language / Virtual Devices

Network Controller (NC)

ICE EI Interface Description Language

Ice Equipment Interface (EI) \( I_1 \)

Ice Equipment Interface (EI) \( I_2 \)

EL Adapter

Native EL-Compliant Physical Device

Physical Device

EL Adapter

Physical Device
### Virtual Medical Device (VMD)

- **MD PnP** enables the concept of VMD
  - A set of medical devices coordinating over a network for clinical scenario

- VMD does not physically exist until instantiated at hospitals

- The Medical Device Coordination Framework (MDCF)
  - Our prototype middleware for managing the correct composition of medical devices into VMD.

![Device Coordination Algorithm](image1)

![Medical Device Types](image2)

![Virtual Medical Device (VMD)](image3)

- Clinician selects appropriate VMD
- MDCF binds appropriate devices into VMD instance
- MDCF displays VMD GUI for clinician

---

**Penn Engineering**

**PRECISE**
VMD Architecture (Impl.)

- Virtual Medical Device 1
- Data Logger
- Ethernet Switch
- Application Server
- MedApp1
- MedApp2
- Alarm System
- MAP

Penn Engineering
VMD Architecture (Impl.)

- MedApp1
- MedApp2
- Application Server
- Ethernet Switch
- Data Logger
- Alarm System
- Virtual Medical Device 2
Safety Assurance Challenge for VMD

- The **new system integration paradigm** of VMD has serious implications for safety assurance, where the traditional approach won't scale.

- **Traditional safety critical systems**
  - fixed function
  - designed and integrated by a single system integrator

Aerospace  Automotive  Nuclear
Traditional System Integration

- End to end process managed by prime contractor

Diagram:
- **Concept of Operations**
  - **Requirements**
  - **Design**
  - **Subsystems Implementation**
- **Deployment**
  - **Systems V & V**
  - **Integration**

**System integration and V & V is done before system is delivered to the customer.**
VMD Development & Assembly

- ConOps
- Requirements
- Design
- Impl / V & V

App Execution (dynamic formation of MAP constituted device)

Performed by runtime environment of VMD

Performed by clinical staff

VMD Instance Assembly

Deployment

Market

Medical Device Manufacturer

VMD Platform Manufacturer

VMD App Developer
VMD Characteristics

- There is **no** prime contractor that is responsible for VMD integration and system-level V&V
  - Assembly is performed after deployment
  - Assembler (hospital staff) **does not have** expert-level technical knowledge of components & system behavior
  - **App developer** is responsible for overall system safety arguments
  - Platform services (compatibility checks) assist in determining at **app launch time** if platform and attached devices satisfy requirements of app
  - The app’s directions for assembly of the platform constituted device are stated **only in terms of properties/capabilities that are exposed on the interfaces** of the platform and devices
Medical Device Certification

- In the U.S., FDA approves medical devices for specific use
  - Safety and effectiveness are assessed
  - Evaluation is process-based: ISO 9001 (quality management) and ISO 14971 (risk management)
  - Hazard analysis is key to approval
  - FDA’s 510(k) requires “substantially equivalent” to devices on the market

- No certification of interoperable medical devices
  - Currently, each collection of interconnected devices is a new medical device to be approved.
Current Regulatory Approach

Current regulation of integrated systems (e.g., central station monitors) requires "pair-wise" clearance: whenever a new type of device is added to the monitoring platform, the entire infrastructure must be re-cleared.

Assume monitoring system was originally developed, verified, and received regulatory clearance for devices of type X & Y.

In current regulatory approach, adding a new type of device (e.g., Z) typically causes the entire system to be re-submitted for regulatory clearance.
Pairwise Certification Complexity

Example “interoperable” device ecosystem: 3 different (model/manufacturer) blood oxygen sensors, 3 different (model/manufacturer) PCA pumps:

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Pumps</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO₁</td>
<td>PCA₁</td>
</tr>
<tr>
<td>PO₂</td>
<td>PCA₂</td>
</tr>
<tr>
<td>PO₃</td>
<td>PCA₃</td>
</tr>
</tbody>
</table>

Each sensor must be approved or certified for use with each pump and vice versa. This is burdensome for manufacturers and regulators.
Example “interoperable” device ecosystem: 3 different (model/manufacturer) blood oxygen sensors, 3 different (model/manufacturer) PCA pumps:

- Sensors: PO \(_1\), PO \(_2\), PO \(_3\)
- Pumps: PCA \(_1\), PCA \(_2\), PCA \(_3\)

Composition of sensor satisfying IS and pump satisfying IP is shown to be safe and effective.

Each sensor (or pump) only needs certification or approval w.r.t. the interface spec. Additionally, the ecosystem can grow without forcing recertification (or re-approval) of previously analyzed devices.
Some Observations …

• Safety can only be assured by predicting the emergent system behavior
  – Vendors cannot use traditional methods to directly predict a VMD's behavior, because the system does not exit until assembled by hospital users

• Safety requirements for specific clinical scenarios
  – Devices can interact in unexpected ways, creating new hazards for the patient
  – Manufacturers unlikely anticipate safety hazards for all possible clinical scenarios
The Proposed Platform Approach

- Maintain a curated ecosphere of Devices, Apps, and Platforms
  - **Apps** define “the system”:
    - Implement the clinical scenario algorithm
    - Specify required devices and their required behavior
    - App can be analyzed for safety using “models” as proxies for concrete devices and environment
  - **Devices** carry out required functions
    - Its (formal) capabilities model is captured by its “interface”
    - Adherence of a device to its capabilities needs to be “certified”
  - **Platforms** run the applications and facilitate system composition:
    - Ensures apps are only composed with compatible devices
    - Ensures app QoS requirements are met

- How does the ecosphere work?
Model–based Safety Reasoning

- Why model–based reasoning (MBR)?
  - Each App defines a set of possible systems, each of which is an allowed combination of medical devices and platforms
  - App vendors would not be able to analyze all possible systems directly since
    - The number of device/platform combinations may be huge
    - New devices may be admitted after the App is certified

- What type of models?
  - Models must capture all the relevant behavior of allowed system combinations
  - The suitability of models and their analysis is dependent on:
    - Ecosphere certification/assurance processes
    - Platform quality / capabilities
    - Ecosphere notion of device / app compatibility
    - Intended use of the system
    - The safety properties being checked
Safety Assurance for VMD

- Model-based analysis at design time
- Validation of modeling assumptions during assembly

Compliant devices

Clinical scenario model

Model is verified to be safe

Compliant deployment platform

VMD is safe
Development and Instantiation

- Model the VMD and verify its safety properties
  - Models of constituent devices
  - Scenario logic

- Two assumptions:
  - Devices behave according to their models
  - Execution and communication semantics are guaranteed by the deployment platform
Assume–Guarantee Safety Assurance

- **Goal**: guarantee that $P(A) (||_{j=1\ldots n} D_j) || E \models \phi$

The execution of App $A$ on the platform $P$, denoted by $P(A)$, together with the assembly of medical devices $D_1, \ldots, D_n$ in the environment $E$ satisfies the safety property $\phi$.

- **Entities in the assume–guarantee reasoning rule**

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
<td>$P(A)$</td>
</tr>
<tr>
<td>Interface</td>
<td>$A_{I_i}^m$ ($i=1\ldots n$)</td>
<td>$A_{I_i}$ ($i=1\ldots n$)</td>
<td></td>
</tr>
<tr>
<td>Devices</td>
<td>$A_{I_i}^m$ ($i=1\ldots n$)</td>
<td>$D_{I_i}$ ($i=1\ldots n$)</td>
<td>$D_i$ ($i=1\ldots n$)</td>
</tr>
<tr>
<td>Platform</td>
<td>$P^m$</td>
<td></td>
<td>$P$</td>
</tr>
<tr>
<td>Environment</td>
<td>$E^m$</td>
<td></td>
<td>$E$</td>
</tr>
</tbody>
</table>
Assume–Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
</tr>
<tr>
<td>Interface</td>
<td>$AI_j^m$ ($j=1...n$)</td>
<td>$AI_j$ ($j=1...n$)</td>
</tr>
<tr>
<td>Devices</td>
<td>$Dl_j$ ($j=1...n$)</td>
<td>$D_j$ ($j=1...n$)</td>
</tr>
<tr>
<td>Platform</td>
<td>$P^m$</td>
<td>$P$</td>
</tr>
<tr>
<td>Environment</td>
<td>$E^m$</td>
<td>$E$</td>
</tr>
</tbody>
</table>

1. $A^m \simeq A$
2. $AI_j^m \simeq AI_j$
3. $P^m \simeq P$
4. $E^m \simeq E$

App developers need to assure that models are faithful to the implementation/platform/environment.
### Assume-Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
<td>$P(A)$</td>
</tr>
<tr>
<td>Interface</td>
<td>$Al_j^m$ (j=1…n)</td>
<td>$Al_j$ (j=1…n)</td>
<td>$D_j$ (j=1…n)</td>
</tr>
<tr>
<td>Devices</td>
<td></td>
<td>$Dl_j$ (j=1…n)</td>
<td>$P$</td>
</tr>
<tr>
<td>Platform</td>
<td>$P^m$</td>
<td></td>
<td>$P$</td>
</tr>
<tr>
<td>Environment</td>
<td>$E^m$</td>
<td></td>
<td>$E$</td>
</tr>
</tbody>
</table>

1. $A^m \simeq A$
2. $Al_j^m \simeq Al_j$
3. $P^m \simeq P$
4. $E^m \simeq E$
5. $A^m (\|_{j=1\ldots n} Al_j^m) \| P^m \| E^m \models \phi$

App developers use model checking to verify that the composed system model satisfies the safety property.
Assume-Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
</tr>
<tr>
<td>Interface</td>
<td>$AI_j^m$ ($j=1\ldots n$)</td>
<td>$AI_j$ ($j=1\ldots n$)</td>
</tr>
<tr>
<td>Devices</td>
<td>$P_m$</td>
<td>$D_i_j$ ($j=1\ldots n$)</td>
</tr>
<tr>
<td>Platform</td>
<td>$P_m$</td>
<td>$P$</td>
</tr>
<tr>
<td>Environment</td>
<td>$E^m$</td>
<td>$E$</td>
</tr>
</tbody>
</table>

1. $A^m \simeq A$
2. $AI_j^m \simeq AI_j$
3. $P_m \simeq P$
4. $E^m \simeq E$
5. $A^m (||_{j=1\ldots n} AI_j^m) || P_m || E^m \models \phi$

\[1\rightarrow 5\] $A (||_{j=1\ldots n} AI_j) || P || E \models \phi$
Task-Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Interface</td>
<td>$A^m$</td>
<td>$A$</td>
<td>$P(A)$</td>
</tr>
<tr>
<td>Interface</td>
<td>$A_{l_j}^m$ ($j=1$…$n$)</td>
<td>$A_{l_j}$ ($j=1$…$n$)</td>
<td></td>
</tr>
<tr>
<td>Devices</td>
<td>$P_m$</td>
<td>$D_{l_j}$ ($j=1$…$n$)</td>
<td>$D_j$ ($j=1$…$n$)</td>
</tr>
<tr>
<td>Platform</td>
<td>$E_m$</td>
<td></td>
<td>$E$</td>
</tr>
<tr>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1–5 $A (||_{j=1...n} A_{l_j}) || P || E \models \phi$

6 $D_{l_j} \simeq D_j$

Device manufacturers need to assure that a device's capability specification conforms to its actual behavior.
### Assume–Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>A^m</td>
<td>A</td>
</tr>
<tr>
<td>Interface</td>
<td>Al_j^m (j=1…n)</td>
<td>Al_j (j=1…n)</td>
</tr>
<tr>
<td>Devices</td>
<td></td>
<td>Dl_j (j=1…n)</td>
</tr>
<tr>
<td>Platform</td>
<td>P^m</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>E^m</td>
<td></td>
</tr>
</tbody>
</table>

1–5 \( A (\| j=1…n Al_j) \| P \| E \models \phi \)

6 \( Dl_j \simeq D_j \)

7 \( Al_j \simeq Dl_j \) (or \( Dl_j \) refines \( Al_j \))

The compatibility between the App's interface about the required device specification and the actual devices' capability needs to be checked, e.g. by third-party certifiers.
### Assume-Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
</tr>
<tr>
<td>Interface</td>
<td>$Al_j^m$ $(j=1 \ldots n)$</td>
<td>$Al_j$ $(j=1 \ldots n)$</td>
</tr>
<tr>
<td>Devices</td>
<td>$P^m$</td>
<td>$P(A)$</td>
</tr>
<tr>
<td>Platform</td>
<td>$Em$</td>
<td>$P$</td>
</tr>
<tr>
<td>Environment</td>
<td>$Em$</td>
<td>$E$</td>
</tr>
</tbody>
</table>

1–5  $A (\parallel_{j=1 \ldots n} Al_j) \parallel P \parallel E \models \phi$

6  $Di_j \simeq D_j$

7  $Al_j \simeq Di_j$

1–7  $A (\parallel_{j=1 \ldots n} D_j) \parallel P \parallel E \models \phi$
Assume-Guarantee Reasoning Rule

<table>
<thead>
<tr>
<th>Model</th>
<th>Software / Specification</th>
<th>Physical Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>$A^m$</td>
<td>$A$</td>
</tr>
<tr>
<td>Interface</td>
<td>$AI_j^m (j=1…n)$</td>
<td>$AI_j (j=1…n)$</td>
</tr>
<tr>
<td>Devices</td>
<td>$P^m$</td>
<td>$Pm$</td>
</tr>
<tr>
<td>Platform</td>
<td>$Em$</td>
<td>$Em$</td>
</tr>
<tr>
<td>Environment</td>
<td>$Em$</td>
<td></td>
</tr>
</tbody>
</table>

1–7  $A (\parallel_{j=1…n} D_j) \parallel P \parallel E \models \phi$

8  $A \parallel P \simeq P(A)$ /* $P(A)$ means $Dj$’s are compatible for $A$ */

1–8  $P(A) (\parallel_{j=1…n} D_j) \parallel E \models \phi$

The execution of App $A$ on the platform $P$, denoted by $P(A)$, together with the assembly of medical devices $D_1, \ldots, D_n$ in the environment $E$ satisfies the safety property $\phi$. 
G: AllSat
There is adequate assurance that all possible instantiations of \{App\} satisfy \(\varphi\) in \{E\}

S: PlatArg
Argue via the platform approach

G: ModelSat
\(\{A^m\}, \{E^m\}\), and for all \(1 \leq j \leq n\) \(\{A_i^m\}\)
\(n = \#\) of dev. req.

G: ModelsAdequate
the models \(\{A^m\}, \{A_i^m\}\)
\(1 \leq j \leq n\), and \(\{P^m, E^m\}\) are adequate for \(\varphi\) and \{E\}

G: PlatformAssurance
There is adequate assurance that all platforms in the ecosphere will correctly execute \{App\}'s \{A\} and will correctly perform app device matching.

G: {Env}ModelAdq.
The model \{E^m\} captures the behavior of \{E\} relevant to \(\varphi\).

G: DevModel{N}Adq.
Given ecosphere compliance assurances: If \{D\} complies with \{DI\} and \{DI\} is compatible with \{AI_i\} then \(\{A_i^m\}\) captures the behavior of \{D\} relevant to \(\varphi\).

G: AlgModelAdq
The model \(A^m\) captures the behavior of \(A\) relevant to \(\varphi\).

Cntxt: Models
The models \{A^m\}, \{E^m\}, and for all \(1 \leq j \leq n\) \(\{A_i^m\}\)
\(n = \#\) of dev. req.

Cntxt: Ecosphere
Ecosphere compliance mechanisms and associated assurance.
Case Study: PCA Control App

vmd ClosedLoopPCA
devices
  pcaPump : PCA
  po : PulseOximeter
logicmodules
  controller : PCATicketGenerator
dataflows
  po.SpO2 $\rightarrow^{50\text{ms}}$ controller.SpO2
  controller.ticket $\rightarrow^{100\text{ms}}$ pcaPump.ticket
Example Assurance Case

**G: NoOverInfusion**
There is adequate assurance that all possible instantiations of PCAapp prevent PCA overinfusion.

**S: PlatArg**
Argue via the platform approach.

**G: ModelSat**
The timed automata models do not exhibit overinfusion.

**S: ModelChecking**
Satisfiability is verified using a model-checker.

**G: ModelsAdequate**
The Patient, PCAappAlg, PulseOx, and PCA models are adequate.

**S: PlatAssurance**
There is adequate assurance that all platforms in the ecosphere will correctly execute PCAapp and will correctly perform PCAapp's device matching.

**G: ModelAdeq**
The PulseOx model captures all relevant behavior of compliant and compatible pulse-oximeters.

**S: PulseOxModelAdeq**
The application model captures the relevant behavior.

**G: PCAapAlgModelAdeq**
The application model captures the relevant behavior.

**S: ModelChecking**
satisfiability is verified using a model-checker.

**G: ModelSat**
The timed automata models do not exhibit overinfusion.

**S: PlatArg**
Argue via the platform approach.

**G: PlatAssurance**
There is adequate assurance that all platforms in the ecosphere will correctly execute PCAapp and will correctly perform PCAapp's device matching.

**G: PatientAdeq**
The patient model is sound for checking for over-infusion.

**S: AcceptedTextBookModel**
The patient model is an accepted good model of opioid pharmokinetics.

**S: DevAssurance**
Argue that PulseOx models all the relevant behaviors (timing, accuracy) that are allowed by the ecosphere pulseox compliance criteria.

**S: ModelChecking**
satisfiability is verified using a model-checker.

**G: PcaModel**
Sound

**Ev: UPPAAL**
UPPAAL model-checking results.

**Ev: TextbookCitatio**
The reference to the patient model.

**Ev: EcoDevCompliance**
The compliance criteria used by the ecosphere.

**S: ModelBasedDevel.**
Argue that the application code was automatically derived from the timed automata model so the model contains all application behavior.

**Ev: TIMESToolLog**
Log files generated by the TIMES tool from the code derivation event.

**S: PlatformAssurance**
There is adequate assurance that all platforms in the ecosphere will correctly execute PCAapp and will correctly perform PCAapp's device matching.

**G: PCAappAlgModelAdeq**
The application model captures the relevant behavior.

**S: ReferenceEcoAssur.**
The ecosphere platform compliance criteria meets an accepted level of assurance for life-critical apps.

**G: PCAModel**
Sound

**Ev: EcoDevCompliance**
Ref. to the compliance criteria for platforms.
Summary

• Propose an assurance argument pattern to assist the safety analysis of plug & play MCPS that consist of
  – a set of medical devices
  – an App (i.e., a software component that coordinates the medical devices for a specific clinical scenario),
  – and a platform that runs the App

• Present an assume–guarantee compositional proof rule/framework for plug & play MCPS and show how it can be used to as a logical basis for the proposed pattern
  – model–based analysis at design time
  – validation of modeling assumptions during assembly
Thank You!
Questions?

http://precise.seas.upenn.edu