University of Pennsylvania
ScholarlyCommons

Departmental Papers (CIS) Department of Computer & Information Science

4-1-2012

Rationale and Architecture Principles for Medical Application Platforms

John Hatcliff
Kansas State University

Andrew King
University of Pennsylvania

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Alasdair MacDonald
eHealth Technology

Anura Fernando
UL (Underwriters Laboratories)

See next page for additional authors

ACM/IEEE Third International Conference on Cyber-Physical Systems (ICCPS 2012) http://iccps2012.cse.wustl.edu/, held as part of CPSWeek http://triton.towson.edu/~cpsweek/ , in Beijing, China, April 2012. (Invited Paper)

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cis_papers/735
For more information, please contact repository@pobox.upenn.edu.

Author(s)
John Hatcliff, Andrew King, Insup Lee, Alasdair MacDonald, Anura Fernando, Michael Robkin, Eugene Vasserman, Sandy Weininger, and Julian M. Goldman

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/cis_papers/735

Rationale and Architecture Principles for Medical Application Platforms

John Hatcliff
Kansas State University

Andrew King, Insup Lee
University of Pennsylvania

Alasdair MacDonald
eHealth Technology

Anura Fernando
UL (Underwriters Laboratories)

Michael Robkin
Anakena Solutions

Eugene Vasserman
Kansas State University

Sandy Weininger
US Food and Drug Administration

Julian M. Goldman
Massachusetts General Hospital
CIMIT MD PnP Program

***Abstract*—The concept of "system of systems" architecture is increasingly prevalent in many critical domains. Such systems allow information to be pulled from a variety of sources, analyzed to discover correlations and trends, stored to enable real-time and post-hoc assessment, mined to better inform decision-making, and leveraged to automate control of system units. In contrast, medical devices typically have been developed as monolithic stand-alone units. However, a vision is emerging of a notion of a medical application platform (MAP) that would provide device and health information systems (HIS) interoperability, safety critical network middleware, and an execution environment for clinical applications ("apps") that offer numerous advantages for safety and effectiveness in health care delivery.**

In this paper, we present the clinical safety/effectiveness and economic motivations for MAPs, and describe key characteristics of MAPs that are guiding the search for appropriate technology, regulatory, and ecosystem solutions. We give an overview of the Integrated Clinical Environment (ICE) – one particular achitecture for MAPs, and the Medical Device Coordination Framework – a prototype implementation of the ICE architecture.

***Keywords*-computing platform, medical device, interoperability, safety-critical systems, certification**

I. Introduction

Historically, medical devices have been developed as stand-alone units. Though many devices on the market already include some form of connectivity (serial ports, Ethernet, 802.11 or Bluetooth wireless, etc.), connectivity is usually only a unidirectional flow to log data/events from these devices in a predetermined manner. There are no widely adopted interoperability standards that can support innovations based on leveraging connectivity. Each device usually functions as a stand-alone system with its own set of sensors to assess patient physiological properties and possibly actuators to deliver treatment. Typically, devices support general care cases and cannot be customized to take advantage of context-specific information about a particular care-giving context. Because devices are not context aware, they often generate nuisance alarms that a clinician can immediately recognize as irrelevant or not indicative of a patient's true health status. There are no engineering and development approaches that support easy and rapid implementation of "smart alarms" that integrate physiological data from multiple devices and use context awareness to suppress false alarms. In situations where devices are currently integrated, they are vertically integrated by a single manufacturer, and the configuration of the devices in the system is known a priori. Due to the manner in which devices are constructed, it is difficult to update the algorithms or other capabilities, leading to a situation where "technology refresh" is more difficult and costly than it should be and technology evolution in health care is slower than it could be.

This work is supported in part by the National Science Foundation under Grants #0734204, 0932289,1065887, and by the NIH/NIBIB Quantum program.

A. Information Integration in Other Domains

In contrast to the state of affairs in the medical context, the impact of information integration in other domains has been driven by three important trends:

Device Interoperability: Standards such the Universal Serial Bus (USB) Plug-n-Play, IEEE 1394 (FireWire), and IEEE 802.11 (Wi-Fi) have revolutionized the consumer electronics domain by enabling heterogeneous collections of devices from different manufacturers to *interconnect*. These standards provide communication protocols as well as device discovery mechanisms. Moreover, in the case of USB and FireWire, they provide a mechanism for a device to dynamically transmit a description of its capabilities at connection time, enabling *interoperability* (but at a level lower than application logic). Compliance testing to standards, often carried out by third parties, plays a crucial role and provides justification for a consumer to trust that a device will interoperate correctly. Third-party testing is supplemented with development environments and conformance test suites that device developers can use during development prior to official third-party testing to determine if they correctly implement the standard.

Platform Approach: Systems are integrated using a computing platform that provides an execution environment, resources, standard libraries, and services to enable developers to more easily assemble applications for a particular domain. Computing platforms are often based on virtual machines or standardized APIs, which make applications less dependent on underlying hardware (allowing hardware to be changed more easily) and enable applications to run

consistently on different hardware. Additionally, platforms provide greater safety and security by limiting access to hardware and software resources to well-defined APIs and enforcing limits at runtime.

Recently, the mobile device space has brought us some of the most successful platforms (e.g. iOS and Android). Rather than rely exclusively on pre-loaded applications and services, these platforms have drastically reduced the cost of entry for software manufacturers by providing comprehensive software development kits (SDKs) including compilers, debuggers, analyses, etc., tailored to the specific workflows involved in building and validating applications for these platforms. Combined with well-defined interfaces to phone features such as (a) hardware devices like GPS, accelerometers, and proximity sensors, (b) local and Internet-accessible services like voice recognition, and (c) accessory devices accessible via Bluetooth and Wi-Fi, this has led to the development of an enormous commodity market for third-party applications ("apps"), which provide rich functionality that adds significant value to the user experience. Platform uptake by both developers and users is further facilitated by an app certification process that rejects or revokes apps which use platform resources incorrectly or inappropriately or diminish the user experience.

Embracing a "Systems Perspective": The notion of an integrated "system of systems" is becoming increasingly prevalent. Small to mid-scale examples include automotive and avionics systems, where many microcontrollers, sensors, and actuators interconnect via a communications infrastructure that allows information from each set of sensors to calculate actions of actuators across the entire system. In larger examples, military command and industrial control systems are increasingly moving from stand-alone, monolithic designs to integrated platforms. Just as platforms provide standard APIs to hardware services, information systems can provide a consistent view of local and remote services to applications, allowing for rapid application development and increased innovation as the process of moving from ideas to implementations is streamlined with standardized APIs and widely available easy-to-use tools.

B. The Need for Solutions

Why have these advances not propagated to the medical device domain? It is certainly not due to lack of demand. There are several significant pressures that are driving the demand for innovative solutions and new paradigms of medical systems. An aging population is expected to produce a huge increase in healthcare demand. A highly mobile population is increasing the need to more easily exchange healthcare information across large geographical areas. Government emphases on outcome-based medicine that focus on both quality enhancements and cost reductions are requiring a much more encompassing collection of clinical data and tracking of clinical workflows. Numerous opportunities for addressing these pressures would accrue if key aspects of the technology trends above could be harnessed for use in the healthcare arena. From a systems engineering perspective, it is well understood that the current state of practice uses non-integrated devices and health information systems cooperatively according to informal manual protocols to deliver clinical solutions. Providing the IT infrastructure to integrate devices and information systems and automatically coordinate their actions as a "system of systems" using computer coded protocols would provide opportunities to implement multi-device smart alarms and safety interlocks, enable clinical decision support systems, automate clinical workflows, and implement multi-device closed loop control. Engineering a safety-critical medical computing platform would encourage the development of a commodity market of safe and innovative clinical apps that implement these envisioned multi-device system functionalities.

C. The Obstacles to Progress

Fundamentally, what is needed to enable a "systems of systems approach" to clinical care is a means by which systems can be composed easily from heterogeneous "building blocks" (devices and IT systems of different types and from different manufacturers) in a safe way. While there are numerous challenges in achieving this vision, there are specific obstacles that impeding progress.

Lack of Appropriate Interoperability Standards: While there has been significant activity on standards and organizations for health information exchange (*e.g.*, HL7, DICOM, IHE), these do not address the technical requirements for device connectivity, safety, and security needed for systems engineering. The most mature existing device interoperability standard, IEEE 11073 (Point of Care, or POC), and its less ambitious and more modern relative IEEE 11073 (Personal Health Devices, or PHD) does not support needed real-time, safety, security, and device control capabilities.

Lack of Appropriate Architectures: Within other critical domains, progress is being made on architectures supporting a platform perspective and compositional construction of systems. In avionics, Integrated Modular Avionics supports building systems with hard real-time constraints using (a) networked computing modules with different levels of criticality, (b) common interfaces for both hardware and software, and (c) principles of portability across an assembly of common hardware modules. In the security community, the Multiple Independent Levels of Security (MILS) architecture aims to define common infrastructure components including a base computing platform (called a separation kernel) that facilitate rapid component-based development of information assurance applications that satisfy stringent certification criteria by appealing to foundational data and time partitioning properties provided by the underlying infrastructure components. One of the explicitly stated goals of both of these architectures is to facilitate the growth of

commodity markets of verified/certified system components, which will in turn enable faster, cheaper, and more reliable construction of systems. There is nothing resembling this type of approach for safety critical systems in the medical device community.

Lack of a Regulatory Pathway: Currently, the U.S. Food and Drug Administration (FDA) and analogous agencies world-wide phrase their regulatory activities in terms of complete systems — there is no pathway now for obtaining regulatory approval for individual system constituents and subsequently allowing a complete system to be assembled and deployed from approved constituents with significantly reduced regulatory approval of the system itself (*i.e.*, "component-wise" regulation). Under existing rules, this implies that when a system includes interchangeable constituents, the system manufacturer must gain regulatory approval for every possible pair (or more generally, permutation) of constituent devices forming the composite medical system (which has been termed "pair-wise" regulation).

Lack of an Ecosystem: Even if the above obstacles were overcome, there would still be a need for community and market-place organizations to form an ecosystem in which development and commercialization of component-based systems of systems could thrive. A key element of the unmitigated success of Apple's mobile app business has been the associated *ecosystem* which provides well-designed interfaces, common infrastructure components, development tools, stringent certification of apps (which allows users to gain an degree trust in the apps). In the medical device space, a corresponding ecosystem would need to include organizations for constructing well-designed interfaces for common medical devices and health IT systems, third-party certification organization that work to ensure trust between system components by rigorously establishing component safety and compliance to interoperability standards, vendors for platform and interfacing components, and education and training materials for those entering the market. In contrast to Apple's proprietary ecosystem whose design and governance lies solely with Apple, the MAP ecosystem should be designed and governed by a variety of stakeholders including device manufacturers, device integrators, standards organizations, certification organizations, health care delivery organizations, and regulatory authorities.

II. Medical Application Platforms

We believe that many of the above shortcomings of existing medical devices can be overcome with a new paradigm of medical systems called *medical application platforms* (MAPs). A MAP is a safety- and security- critical real-time computing platform for (a) integrating heterogeneous devices, medical IT systems, and information displays via a communication infrastructure and (b) hosting application programs (i.e., *apps*) that provide medical utility via the ability to both acquire information from and update/control integrated devices, IT systems, and displays. While our discussions will generally discuss the concept of a MAP in a clinical context, a MAP could also be implemented in a number of ways such as a portable, home-based, mobile or distributed system.

The "medical utility" provided by MAP apps may take many forms, but a common theme is that they introduce a previously missing "system perspective" into the device context associated with patient care.

Medical Display and Storage: An app may transfer data from one or more devices to a patient's electronic medical record or a MAP-supported display such as composite display in a patient's hospital room, a remote clinical display at a nurses station, or a physician's smart phone. Thus, the MAP concept subsumes Medical Data Display Systems (MDDS) and real-time alarm management systems such as Philips Emergin.

Derived/Smart Alarms: Using more advanced logic, an app may implement "derived alarms" to supplement the native alarm capabilities of a device or implement alarms for consumer oriented devices that do not provide them natively (e.g. an app might implement upper and lower limit SpO_2 alarms for Continua compliant pulse oximeters such as the Nonin Onyx II). Alternatively, the app may implement a so-called "smart alarm" that provides more sophisticated analysis and decision logic based on physiological parameters from multiple devices [1], [2], monitoring trends/history, comparison and correlation with data patterns from broader population indicating problematic physiological conditions.

Clinical decision support: An app may pull information from devices, patient electronic medical records, drug interaction databases, and previous clinical studies to support clinician decision making, diagnoses, or guidance/suggestions for treatment.

Safety interlocks: An app may control one or more devices so as to implement system safety invariants that lock out potentially unsafe individual device behaviors or interactions between devices.

Workflow automation: As many clinical procedures follow certain protocols or recommended steps that involve interacting with a collection of devices in patterns known a priori, an app may partially automate workflow steps by automatically activating/deactivating devices or setting device parameters based on a patient's medical record or procedure- and context-dependent guidelines.

Closed-loop control: An app may use information collected from sensing devices and possibly a patient's medical record to control actuators on devices providing treatment or collecting diagnostic information from patients.

A. The Clinician's View

Figure 1 illustrates the clinician's view of a clinical-based MAP. For simplicity, we assume in this discussion that a

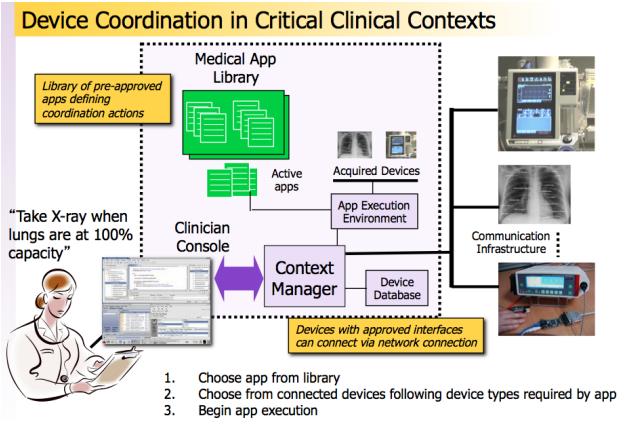

Figure 1. Clinician's View of a Medical Application Platform

MAP supports a single patient. A communication infrastructure connects medical devices that are communications-enabled (*e.g.* via Bluetooth, USB, or Ethernet) as well as other information systems such as a patient electronic medical record (EMR) and drug dosing databases. A device database records the unique identifiers and drivers/interfaces for devices that have been pre-approved for connection to the framework. The app execution environment would typically include a library of apps written by experts and (possibly, if it implements medical device functionality) approved by appropriate regulatory authorities (e.g., the US Food and Drug Administration). A clinician desiring a particular medical system behavior chooses an appropriate app from the library. Each app contains a list of device types and associated device capabilities that are required to carry out a medical system activity. During the app initialization phase, the app execution environment attempts to acquire devices that satisfy the device requirements of the app and are currently connected to the communication infrastructure. There may be more than one connected device that satisfies the device requirements of an app such as a stand-alone pulse oximeter (*e.g.*, the Nellcor OxiMax N-600x) and a pulse-oximeter incorporated into a multi-parameter patient monitor (*e.g.*, the Philips IntelliVue MP90). Thus, the clinician may assist with device acquisition (*e.g.*, by selecting from a list of available devices of the appropriate type). After a complete set of required devices has been selected and confirmed as available (a device may be restricted from participating in more than one app at a time if it is being subjected to automated control, while "read only" devices may be shared), app execution begins. App execution may proceed without intervention, or may stop to receive input from the clinician.

The MAP architecture and infrastructure should include a number of capabilities for guaranteeing safety and security. For example, in event of a broken or degraded communication connection between the MAP and a device there must exist a mechanism to transition both the device and apps that depend on the device to a safe state.

In the subsections below, we sketch several clinical scenarios drawn from Appendix B of [3], and describe how they could be supported by an app.

B. Application: Cardio-pulmonary Bypass

Patients undergoing a cardio-pulmonary bypass operation typically have their breathing supported by an anesthesia machine ventilator during the initial part of surgery, then during the actual operation are switched to a cardio-pulmonary bypass machine which oxygenates their blood directly, and then are switched back to a ventilator (after bypass). Incidents have occurred [4] in which the anesthesiologist forgot to resume ventilation after separation from cardio-pulmonary bypass. In at least one case, documented in [4], "...the delayed detection of apnea was attributed to the fact that the audible alarms from the pulse oximeter and capnograph had been disabled during bypass and had not been reactivated. [The patient] sustained permanent brain damage." Note that in this situation, an error occurred because the following system invariant was violated: either the anesthesia machine or the cardiopulmonary bypass machine must be connected to the patient. It is straightforward to use a medical device coordination framework with a connected anesthesia machine and bypass machine to detect this invariant violation and to raise an appropriate alarm.

C. Application: Laser Surgery Safety Interlock

Modern trachea or larynx surgery often utilizes a laser to remove cancers or non-malignant lesions and a tracheal tube to supply oxygen to the patient's lungs during the operation. A potential hazard is the accidental heating of the tracheal tube by the laser, which can produce an intense fire. Typically, the oxygen saturation level in the tracheal tube is reduced (e.g. to 25%) when the laser is in use to help reduce the chance of a fire in case of an accidental ignition of the tube. There have been a number of injuries and even deaths reported due to fires caused by a laser igniting the tube. Again, the potential system error in this scenario can be mitigated by coordination of the laser and ventilator system. Specifically, the device coordination logic can implement a simple safety interlock that disables the laser if the oxygen saturation is greater than a configured level (e.g. 25%). An alarm can be programmed using information from multiple devices (both the ventilator and laser) so that an alert is raised if there is an attempt to engage the laser when the oxygen level exceeds the configured maximum level.

D. Application: X-ray / ventilator Coordination

A simple example of automating clinician workflows via cooperating devices addresses problems in acquiring accurate chest X-ray images for patients on ventilators during surgery [5]. To keep the lungs' movements from blurring

the image, doctors must manually turn off the ventilator for a few seconds while they acquire the X-ray image, but there are risks in inadvertently leaving the ventilator off for too long. For example, Lofsky documents a case where a patient death resulted when an anesthesiologist forgot to turn the ventilator back on due to a distraction in the operating room associated with dropped X-ray film and a jammed operating table [6]. These risks can be minimized by automatically coordinating the actions of the X-ray imaging device and the ventilator. Specifically, a centralized automated coordinator running a pre-programmed coordination script can use device data from the ventilator over the period of a few respiratory cycles to identify a target image acquisition point where the lungs will be at full inhalation or exhalation (and thus experiencing minimal motion). At the image acquisition point, the controller can pause the ventilator, activate the X-ray machine to acquire the image, and then signal the ventilator to "unpause" and continue the respiration [7].

Note that each of these cases above involves very simple forms of coordination logic that can significantly improve the safety or the effectiveness of treatment for the patient. All of these applications could be implemented today but none are. In our experience, once the concept of device coordination is explained to a surgical clinician, they can almost always come up with an scenario that they have encountered where device coordination would be beneficial.

III. Medical Application Platform System Characteristics

The medical systems constructed using MAPs have characteristics that are substantially different from traditional medical devices and from other cyber-physical systems. In this section, we summarize the characteristics that are especially important to consider when trying to design appropriate architectures, engineering approaches, safety/security solutions, verification technologies, regulatory/certification regimes, and business ecosystems for MAPs.

Cyber-physical Systems of Systems: MAP systems are complex systems composed of smaller systems that are designed, in most cases, to function stand-alone. Drawing from Maier's characterization of SoS [8], the constituents are autonomous with independent operations and management. In many cases, there is an independent evolution of each constituent to respond to new technology and mission needs at its own pace and direction.

Interoperability with Heterogeneous Components: Related to the above, MAP constituents are produced by different manufacturers, each with different approaches to interfacing, specification/documentation of capabilities, quality management, etc. The success of the MAP approach depends to some extent on individual manufactures being willing to pursue interoperability as a business strategy and to follow device design and interface specification strategies that facilitate interoperability.

System Instances are Variable: In contrast to the typical system deployments, the constituents of a system (the system corresponding to a particular MAP constituted device for a specific app) will vary from one instance to the next. More concretely, an app A running at hospital H_1 may utilize one model (a pulse oximeter from manufacturer M_1) for one of its required devices whereas the same app running at hospital H_2 may utilize a different model (a pulse oximeter from manufacturer M_2). This implies that the MAP-constituted device instances determined by app A must satisfy the same safety and effectiveness requirements even though the constituents of those instances may be different and possess different capabilities.

Integration at Runtime After Deployment: In other domains such as avionics where complex systems are assembled from subsystems originating from different manufacturers, there is typically a prime contractor that serves as the *system integrator* and is tasked with assembling the system. The system integrator has expert-level technical knowledge of the system components, and is responsible for the overall system verification/validation, safety arguments, and certification. Integration/assembly is performed *before* deployment with full knowledge of the characteristics and behavior of the components being integrated.

In contrast, for MAP systems, there is no prime contractor who assembles a system, and no single manufacturer delivers the system to the customer. Instead, systems are "composed" at runtime at the point of care by clinical engineers (or even clinicians) by attaching devices to the communication infrastructure and launching apps that dynamically bind to devices with which they may have never been tested. The composition is performed by clinical engineers or clinicians who may not have detailed technical expertise of device components, real-time application programming, nor distributed safety-critical systems engineering.

Open and Extensible: When most conventional critical systems are deployed, the set of possible constituents is known in advance. With MAPs, the analogy to popular mobile platforms implies that one should be able to develop and deploy implementations of the communication infrastructure and app execution environment, and then subsequently use those infrastructure components to support new apps, new devices, and new device types that were not anticipated at the time of infrastructure development, regulatory approval, and deployment. While significant innovations in device or app technology/capabilities may trigger the creation of a new version of the infrastructure (which would need new regulatory clearance), it is likely that the most effective business models for MAPs will require adoption of appropriate architecture and engineering approaches that will avoid the need to seek new regulatory clearances for infrastructure components whenever new apps and devices are introduced.

Safety Critical: Unlike other enterprise systems of systems (*e.g.*., information systems), clearly MAP systems

are safety-critical since unintended behavior can cause human injury or death. Thus, architecture and individual components need to be designed and implemented to support various safety regimes such as fail safe, fail operational, and fail-passive depending on the usage context.

Security Critical: While conventional medical devices have little or no emphasis on security (as they are standalone), providing for secure operation of MAP systems is crucial. We view *security as a strict subset of safety*, since safe operation is impossible if the system is vulnerable to attack. Many MAP use cases are focused on transferring private medical information from devices to displays and electronic medical records, so providing security models/architectures that can be used to ensure correct functionality and conformance to privacy requirements like HIPAA is necessary for MAPs to be marketed and deployed. Moreover, the open nature of MAPs provides many opportunities for malicious apps and devices to disrupt system performance or information. Finally, the complex human operator environment with clinicians with differing roles/permissions and the need to override access controls in emergency situations provides significant challenges.

Component-Wise Regulation: If the existing pair-wise certification approach (see Section I) is applied to MAPs, the regulatory process would be so burdensome as to render moot any technical advantages offered by the envisioned framework. For example, if a company developed a MAP, the need to have a new regulatory review for the entire platform each time a new clinical application or device were integrated with the framework would likely be so costly and time-consuming as to inhibit a successful business model for platform providers and to stifle the development of a commodity market of MAP devices and apps. A *component-wise* regulatory paradigm, in which MAP apps and devices are reviewed individually and independently of the system instance in which they will operate, will enable clinical users to attach devices and launch MAP apps at the point of care (to obtain a MAP constituted device) such the actions of the clinical user do not constitute a regulated activity.

Highly Active Ecosphere: Recent experiences with smart phones as computing platforms (e.g., iPhone and Android) illustrates how well-designed open platforms can encourage innovation and give rise to an explosion in lightweight apps providing highly targeted functionality. Based on this experience, once MAP infrastructure is on the market, there will likely be a flood of MAP apps and device interfaces submitted for regulatory clearance. Moreover, the safety issues in open systems are much more challenging. For example, apps will need to work with devices with which they have previously not been tested, there is potential for interference between apps/devices is much greater, it will be easier for fly by night operators to roll their own apps/interfaces, etc. Therefore, it is crucial the entire MAP ecosphere be designed to ensure its integrity and trustworthiness.

This should include developing standard notions of interfacing for common device types and applications to avoid a proliferation of slightly different and poorly-designed interfaces. In addition, not only do regulatory regimes need to evolve to support compositional regulation, verification and regulatory idioms need to better support (a) increased speed in processing regulatory submissions, and (b) increased scrutiny of safety and functional/security claims.

IV. Solution Goals

Solutions that support the MAP vision may come in different forms. However, the community is faced with the harsh reality that up to this point in time there has not been a successful framework developed that supports the aggregate demands of heterogeneous components, compositional construction, extensibility, safety, and security. We believe that solutions *are* possible, but that to satisfy the demands above, the architecture, platform, and ecosystem for any solution must be sufficiently constrained and fulfill some rather strong goals/requirements.

A. Architecture and Interfacing Goals

Experience in large-scale system development and multi-vendor integration projects indicates that producing an appropriate architecture is crucial to the long-term success and integrity of a project.

Interoperability Architecture: The solution must define an interoperability architecture that identifies the components of the MAP framework at the level of granularity at which interoperability will be applied (i.e., granularity at which components can be interchanged using component-wise regulatory clearance).

Interoperability Points: To achieve composability and substitutability of one component with another that implements the same interface, the solution should identify the specific interoperability points (i.e., interfaces) within the interoperability architecture where components interact. To achieve safety and security, all component interaction must be limited to these explicitly declared interoperability points.

Interface Compliance/Compatibility: The solution should identify processes, verification and validation techniques, existing standards, and other methods that will be applied to ensure that (a) a component is *compliant* to its interface and (b) when composing components, that one side of the interface (e.g., on component A) is *compatible* with the other side of the interface (e.g., on component B plugging to component A.) It is anticipated that component-to-interface compliance should be carried out statically before a component receives regulatory approval and is deployed, whereas the notion of interface compatibility (*e.g.*, between an app and a device) must in some cases be checked dynamically by platform services in order to enable system composition at the point of care.

Rich Device Interface Language: To facilitate safety and the ability of apps to fully leverage device capabilities, the solution should provide an expressive device interface definition language that provides a formal mechanism to specify the both the functional and non-functional capabilities and behaviors of each device to be integrated into the MAP. The description must be sufficient for the MAP to determine if a device provides the capabilities required by a given app. The IEEE 11073 interoperability standard allows for medical devices to describe their capabilities in terms of physiologic signals measured, and how those signals are encoded on the device's network interface, but stops short of completely specifying the actuation capabilities of a device (*e.g.* how long it takes for an actuator to activate when it receives a command) and any local mode switching a device may implement (*e.g.* many infusion pumps will autonomously stop infusing under certain conditions such a detecting air in the infusion line)

B. Platform Goals

Due to the characteristics described in Section III (particularly the runtime composability and safety critical characteristics) a viable MAP should not just implement a good interoperable architecture but must also provide certain capabilities which apps and user can depend on to facilitate safe operation of the system. The capabilities listed below are not complete or sufficient for a safe MAP, but are necessary if the vision of a truly interoperable dynamic ecosystem of medical devices and apps are to flourish.

Direct Support for Static Verification of Systems: The operational semantics and behavior of the MAP must be precisely and formally defined, including operational semantics of the app virtual machine, what performance guarantees the MAP provides, and what it means for a device and application to be "compatible" with each other. Verification and validation techniques can then leverage these definitions to establish important correctness properties of the system. For example, app developers can apply these formally grounded descriptions to make precise, checkable, models of the system behavior that their apps specify.

Composability & Flexibility: An app may be composed with a device at the point of care (*i.e.*, dynamically, after regulatory approval and deployment of system components). Therefore, the platform must provide services to dynamically check that a device's capabilities (expressed through its interface) are compatible with an app's requirements.

Global Resource Management: Medical systems have wall-clock time constraints. Since a MAP system instance is distributed, the platform must ensure that all communication and computation tasks meet their required deadlines. A global resource management capability would allow the MAP to proactively reserve resources (*e.g.*CPU, memory, and network bandwidth) for apps in order to guarantee any specified performance requirements apps may have.

Automatic Trust Establishment: The platform must facilitate the establishment of composition-time trust between the different components used in an instance.

Automatic Security Services: The system must ensure that data cannot be intercepted, monitored, or altered by unauthorized parties, and that unauthorized entities cannot induce unsafe behavior of authorized/honest components.

Direct Support for Runtime Verification of Systems: The system must provide facilities for continuous runtime monitoring, correcting for performance issues and minimizing the effect of malfunctioning components by identifying, isolating, and evicting them in real time.

C. Safety, Effectiveness and Certification Goals

Interoperability Demonstration for Component-Wise Regulation: To establish the groundwork for component-wise regulation within the architecture, the solution proposer should make a sequence of regulatory submissions that demonstrate the compositional safety rationale by sufficiently exercising each of the interoperability points of the framework. A claim of "sufficiently exercised" might be based on, providing regulatory submissions of component instances to demonstrate that any variation across instances is encapsulated in the proposed interfacing, for example.

Third-Party Certification Regime: In other domains, third-party certification bodies play a key role in ensuring trust and integrity by developing sophisticated infrastructure for testing/validating that products conform to safety or interoperability standards (familiar examples including third-party certification to Wi-Fi or USB standards). We believe a third-party certification regime that certifies MAP components compliance to safety and interfacing standards may be crucial for MAPs to (a) ensure trust when composing components from different vendors and (b) reducing the workload of the regulatory body in the context of a high volume of app and interface submissions.

D. Ecosystem goals

Consensus Interfacing for Device Interfacing: It will be necessary to develop an organization of key stakeholders (e.g., device and MAP manufacturers, app suppliers) that will design and maintain a well-organized collection of device interfaces that (a) MAP device manufacturers target when developing interfacing capabilities and (b) developers will target when coding MAP apps.

Standard Development Environments for MAP Apps and Interfaces: Similar to the App Development Kits supplied by Apple and Google and Device Driver Development Kit supplied by Microsoft, the MAP ecosphere should be supported by development environments for both apps and device interfaces. These development environments should include test suites, static analyses, and verification tools that both developers and third-party certifiers apply to validate compliance to MAP interface/app standards — ensuring that

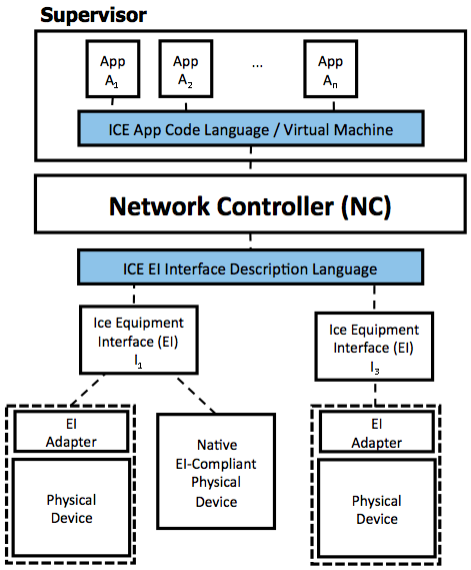

Figure 2. ICE Architecture

elements submitted for certification have been subjected to a uniform and rigorous validation. A potential model here is Continua, which provides a variety of code and testing infrastructure to its members.

V. Integrated Clinical Environment

Different architectural solutions for medical application platforms are possible. One architecture that is gaining traction is the Integrated Clinical Environment (ICE) [3] whose development has been led by the CIMIT Medical Device Plug-and-Play interoperability project and standardized in the ASTM F2761-2009 standard. ASTM F2761-2009 defines the principle components of a MAP architecture and provides rationale for the role that each of these components play in establishing interoperability and safety. ICE has figured prominently in ongoing FDA activities[1] related to interoperability and possible approaches to compositional regulation of MAPs. Several research projects are building elements of MAP prototypes that conform to ICE including an NIH/NIBIB Quantum project led by Dr. Julian Goldman at CIMIT, the Medical Device Coordination Framework [9] developed by the SAnToS Laboratory at Kansas State and the PRECISE Center at University of Pennsylvania, and a prototype illustrating safety principles in the presence of ICE failures at University of Illinois at Urbana-Champaign [10].

Figure 2 shows the primary components of the ICE architecture. In the rest of this section, we summarize the intended functionality and goals for each of these components. It is important to understand that ASTM F2761-2009 does not provide detailed requirements for these, nor has a reference implementation been provided. Thus, our summary should be viewed as "informed speculation" about what will ultimately emerge in ICE implementations.

Supervisor: The supervisor provides a secure isolation kernel and virtual machine (VM) execution environment for MAP apps. It would be responsible for ensuring that apps are partitioned in both data and time from each other.

Network Controller: The network controller is the primary conduit for physiologic signal datastreams and device control messages. The network controller would be responsible for maintaining a list of connected devices and ensuring proper quality of service guarantees in terms of time and data partitioning of datastreams, as well as security services for device authentication and data encryption.

ICE Interface Description Language: The description language is the primary mechanism for ICE-compliant devices to export their capabilities to the network controller. These capabilities may include what sensors and actuators are present on the device, and the command set it supports.

VI. MDCF

The Medical Device Coordination Framework (MDCF) [9], [11] is an open-source project that provides many of the capabilities called out in the ICE standard. The MDCF effort was initiated by the U.S. Food and Drug Administration and is funded by the National Science Foundation Cyber-Physical Systems program as an open test bed to allow academics, industry, and government regulators to explore engineering and safety issues involved in medical application platforms. MDCF development is led by research teams from the SAnToS Laboratory at Kansas State University and the PRECISE Center and the University of Pennsylvania.

A. Goals

Table I lists initial goals for the MDCF software infrastructure that are currently guiding our design and implementation. These goals represent what we believe to be necessary to *begin* exploration of issues faced in realistic MAPs.

There are many other desirable qualities related to verification and validation, safety, security, and fault-tolerance not listed here; one of overarching goals is to provide an infrastructure that enables other research organizations (as well as our own) to investigate those issues. We choose to focus on specification/verification issues.

B. Platform Services

The MDCF is implemented as a collection of services which work together to provide some of the capabilities called out in Section III as essential for a medical application platform. The functionality of these services also decompose along the architectural boundaries defined in the ICE architecture (see Figure 3), thus the MDCF consists of "network controller" services, "supervisor" services and a global resource management service:

[1]For example, the FDA Workshop on Medical Device Interoperability, January 25, 2010 and subsequent formation of the Medical Device Interoperability and Safety Working Group.

1) Provide distributed networking middleware infrastructure that enables devices/displays/databases from different vendors to be integrated with minimal effort.
2) Provide payload capabilities that support common data formats used in the medical device and medical informatics domains.
3) Provide an app infrastructure that enables easy analysis, integration, and transformation of device information streams, as well as easy programming of device control logic.
4) Support a model-based app programming environment that makes it easy to assemble new functionality from building blocks.
5) Design the infrastructure to incorporate a flexible security framework that allows experimentation with security properties appropriate for MAPs.
6) Provide a framework that enables experimentation with techniques for real-time and reliability guarantees on message delivery and app execution.
7) Support the functional and real-time requirements of realistic MAP apps.
8) Use infrastructure that is freely available and open source (to enable academic research).
9) Use standards-based solutions that can support tech transition to industry.
10) Provide a library of mock medical devices to enable easy experimentation with the system by academics.
11) Serve as a test-bed for evidence-based static and dynamic verification technologies appropriate for safety-critical systems.
12) Illustrate the construction of appropriate certification and regulatory artifacts for apps, device interfacing, and infrastructure components.
13) Support prototyping of ICE-related concepts.

Table I
MDCF DEVELOPMENT GOALS

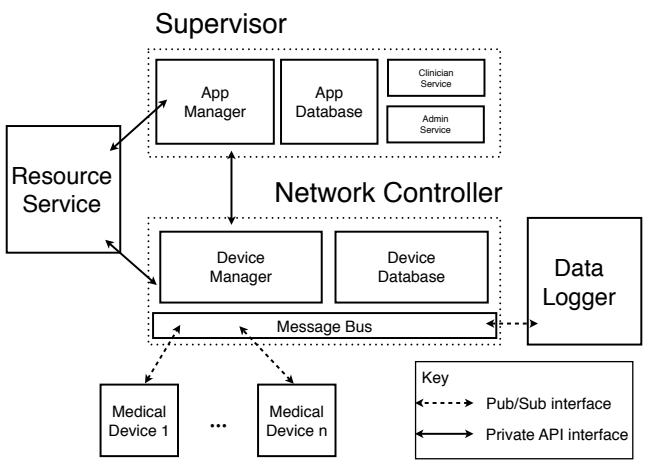

Figure 3. MDCF services decomposed along ICE architectural boundaries

1) Network Controller Services:

Message Bus: Abstracts the low level networking implementation (*e.g.* TCP/IP) and provides a publish/subscribe messaging service. All communication between medical devices and the MDCF occurs via the message bus, including protocol control messages, patient physiologic data, and commands sent from apps to devices. The Message Bus also provides basic real-time guarantees (*e.g.* bounded end-to-end message transmission delays) that apps can take as assumptions. Additionally, the Message Bus supports various fine-grained message and stream access control and isolation policies. While the current implementation of the message bus encodes messages using XML, the actual encoding strategy is abstracted away from the apps *and* devices by the message bus API which exposes messages as structured objects in memory.

Device Manager: Maintains a registry of all medical devices currently connected with the MDCF. The Device Manager implements the server side of the MDCF device connection protocol (medical devices implement the client side) and tracks the connectivity of those devices, notifying the appropriate apps if a device goes offline unexpectedly. The Device Manager serves another important role: it validates the trustworthiness of any connecting device by determining if the connecting device has a valid certificate.

Device Database: Maintains a list of all specific medical devices that the healthcare provider's bioengineering staff has approved for use. In particular, the database lists each allowed device's unique identifier (like an Ethernet MAC address), the manufacturer of the device, and any security keys or certificates that the Device Manager will use to authenticate connecting devices against.

Data Logger: Taps into the flows of messages moving across the message bus and selectively logs them. The logger can be configured with a policy specifying which messages should be recorded. Because the message bus carries carries every message in the system, the logger can be configured to record any message or event that propagates through the MDCF. Logs must be tamper resistant, tamper evident, and access to logs must itself be logged, and be physically and electronically controlled by a security policy.

2) Supervisor Services:

Application Manager: Provides a virtual machine for apps to execute in. In addition to simply executing program code, the Application Manager checks that the MDCF can guarantee the app's requirements at runtime and provides resource and data isolation, as well as access control and other security services. If the app requires a certain medical device, communications latency, or response time from app tasks but the MDCF cannot currently make those guarantees (*e.g.* due to system load or the appropriate medical device has not been connected) then the App Manager will not let the clinician start the app in question. If the resources are available, the application manager will reserve those resources in order to guarantee the required performance to the app. The application manager further detects and flags potential medically meaningful app interactions, since individual apps are isolated and may not be aware what other apps are associated with a given patient.

Application Database: Stores the applications installed in the MDCF. Each application contains executable code and requirement metadata used by the application manager to allocate the appropriate resources for app execution.

Clinician Service: Provides an interface for the clinician console GUI to check the status of the system, start apps, and display app graphical user interface elements. Since this interface is exposed as a service, the clinician console can be run locally (on the same machine) that is

running the supervisor, or remotely (*e.g.* at a nurse's station).

Administrator Service: Provides an interface for the administrator's console. System administrators can use the administrator's console to install new applications, remove applications, add devices to the device database and monitor the performance of the system.

C. Resources and Applications

In addition to the open-source distribution [9], a variety of resources are available for the MDCF including tutorials, presentations, and video lectures on topics such as the architecture and code organization of the MDCF, building interfaces for real devices, building simulated devices, building apps, developing app regulatory/certification artifacts, and a variety of clinical application scenarios.

In addition to the simple apps included in the distribution, the MDCF has been used to develop some more significant examples including an app which provides closed-loop control of a PCA pump [12], a generic smart alarm framework that uses fuzzy set classification to generate more precise alarms by aggregating physiological readings from a multi-parameter monitor [1].

MDCF is also being used in demonstration efforts on the CIMIT-led NIBIB Quantum project that aims to deliver an ICE-compliant open-source hospital intranet, and as part of an NSF FDA Scholar-in-Residence project to demonstrate MAP concepts and artifacts to FDA engineers.

VII. Conclusion

We have presented the concept of a medical application platform as a means by which a much-needed "systems perspective" can be achieved in the cyber-physical system domain of medical systems. In addition, we have sketched some of the fundamental and distinguishing characteristics of MAPs and presented a selection of goals that we believe properly developed MAP solutions must satisfy.

There are several noteworthy activities that are aiming to mature these concepts and appropriately expose them to the broader community. The Medical Device Interoperability Safety Working Group (MDISWG) – a working group that grew out of the January 2010 FDA workshop entitled "Medical Device Interoperability: achieving safety and effectiveness" – is focused on clarifying the regulatory pathway for MAP systems by writing documents outlining principles of component-wise safety submitted to the FDA as part of its Investigational Device Exemption (IDE) program. An NIBIB Quantum Project led by Dr. Julian Goldman at CIMIT is aiming to develop a prototype healthcare intranet that complies with the ICE standard. MD FIRE is a CIMIT-led effort to promote the adoption of fully interoperable medical devices and systems in support of patient safety by developing sample RFP and Contracting language requirements for interoperability that health care delivery organizations can use when purchasing medical equipment. UL (Underwriters Laboratory) is developing a family of standards related to device interoperability that aim to address many of the issues presented in this paper.

Acknowledgements

The authors gratefully acknowledge discussions with the Medical Device Safety and Interoperability Working Group and NIBIB Quantum Project "Development of a Prototype Healthcare Intranet for Improved Health Outcomes" that contributed to the formation of ideas presented in this paper.

References

[1] A. L. King, A. Roederer, D. Arney, S. Chen, M. Fortino-Mullen, A. Giannareas, W. Hanson, III, V. Kern, N. Stevens, J. Tannen, A. V. Trevino, S. Park, O. Sokolsky, and I. Lee, "GSA: A framework for rapid prototyping of smart alarm systems," in *Proceedings of the 1st ACM International Health Informatics Symposium*, 2010.

[2] G. Hackmann, M. Chen, O. Chipara, C. Lu, Y. Chen, M. Kollef, and T. Bailey, "Toward a two-tier clinical warning system for hospitalized patients," in *Proceedings of the 2011 American Medical Informatics Association Annual Symposium (AMIA'11)*, Oct. 2011.

[3] *ASTM F2761-2009. Medical Devices and Medical Systems — Essential Safety Requirements for Equipment Comprising the Patient-Centric Integrated Clinical Environment (ICE), Part 1: General Requirements and Conceptual Model*, ASTM International, 2009.

[4] R. A. Caplan, M. F. Vistica, K. L. Posner, and F. W. Cheney, "Adverse anesthetic outcomes arising from gas delivery equipment: A closed claims analysis," *Anesthesiology*, vol. 87, no. 4, 1997.

[5] P. B. Langevin, V. Hellein, S. M. Harms, W. K. Tharp, C. Cheung-Seekit, and S. Lampotang, "Synchronization of radiograph film exposure with the inspiratory pause," *American Journal of Respiratory Critical Care Medicine*, vol. 160, no. 6, 1999.

[6] A. S. Lofsky, "Turn your alarms on," *APSF Newsletter*, vol. Winter, 2005.

[7] K. Grifantini, ""Plug and Play" hospitals: Medical devices that exchange data could make hospitals safer," MIT Technology Review, Jul. 2008.

[8] M. Maier, "Architecting principles for systems of systems," *Systems Engineering*, vol. 1, no. 4, 1998.

[9] "Medical Device Coordination Framework (MDCF) website," http://mdcf.santos.cis.ksu.edu.

[10] M. Sun, Q. Wang, and L. Sha, "Building reliable MD PnP systems," in *Proceedings of the Joint Workshop On High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability*, 2007.

[11] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger, "An open test bed for medical device integration and coordination," in *Proceedings of the 31st International Conference on Software Engineering*, 2009.

[12] A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter, "Prototyping closed loop physiologic control with the medical device coordination framework," in *ICSE Companion*, 2010.