Requirements Specification for Apps in Medical Application Platforms

Brian Larson, John Hatcliff, Sam Procter, Patrice Chalin
Kansas State University
{*brl,hatcliff,samuel3,chalin*}*@ksu.edu*

***Abstract*—Existing regulatory agency guidance documents and process standards for medical devices (*i.e.*, IEC 62304) generally consider medical devices to be stand-alone monolithic systems. The format and content of a system requirements document largely follows that of conventional embedded safety-critical systems. However, a vision is emerging of a new paradigm of medical system based on the notion of a *medical application platform* (MAP). 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. To ensure a regulatory pathway for MAPs, it is necessary to adapt traditional development processes and artifacts to the specific characteristics of MAP architectures and constituent components. In this paper, we provide an initial proposal for developing and formatting *requirements* for MAP apps. For illustration, we consider an app that implements two "smart alarms" for pulse oximetry monitoring in a clinical context.**

***Keywords*-requirements specification, biomedical communication, biomedical monitoring, medical information systems**

I. Introduction

The evolution of networking technologies, interoperability standards, information integration, and automated controls technology has increased the speed of system innovation in many fields, but the medical industry has so far not kept pace. Elsewhere, including safety-critical infrastructures, the notion of an interoperable "system of systems" is increasingly prevalent. Such architectures 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. No widely-used *device* interoperability standards exist[1] Current verification and validation (V&V) techniques used in industry primarily target single monolithic systems. Moreover, the United States Food and Drug Administration's (FDA) regulatory clearance processes are designed to approve single stand-alone devices or collections of devices that are integrated by a single manufacturer with complete control over all components. 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 Information Technology (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.

[1]Though, *e.g.*, HL7 is a widely used "Health Information Technology" standard.

II. Medical Application Platforms

A vision is emerging of a new paradigm of medical system enabled by, what we call, *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 shared communication infrastructure and (b) hosting application programs (which we will refer to simply as *apps*) that provide medical utility via the ability to both acquire information from and exert control over integrated devices, IT systems and displays. While we will generally discuss the concept of a MAP in a clinical context, a MAP could also be implemented as either a portable (*i.e.*, ambulatory), home-based or mobile system.

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

There are a broad range of types of MAP application, including:

Medical display and storage: An app may transfer and possibly consolidate data from one or more devices to a patient's electronic medical record or a MAP-supported display. This could take the form of a composite display in a patient's hospital room, a remote clinical display at a nurses station, or a physician's smart phone.

Derived / smart alarms: An app may implement "derived alarms" to supplement the native alarm capabilities of a device. This might include implementing alarms for consumer oriented devices that do not provide native alarms—*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, monitoring trends / history, or comparison and correlation with data patterns from a 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: Many clinical procedures follow protocols or recommended steps that involve interacting with a collection of devices. An app may be used to partially automate workflow steps by automatically activating / deactivating devices, setting device parameters based on either a patient's medical record or on guidelines for a particular type of procedure.

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.

III. MAP Architecture

While there may be multiple suitable architectures for MAPs, one architecture that is gaining traction in the regulatory domain is ASTM standard F2761-09 for the Integrated Clinical Environment (ICE) [1]. ASTM F2761-09 is an initial standard in an anticipated family of standards that will flesh out detailed requirements and interfaces for ICE components. ASTM F2761-09 itself is a short standard that presents the high-level ICE architecture (corresponding to the components in dashed lines in Figure 1) and gives a brief description of each architecture component. Nevertheless, the ICE architecture has become the basis for US Food and Drug Administration (FDA) sponsored workshops and working groups that are developing a regulatory pathway for MAPs. Our research group has been extensively involved in these activities, and is attempting to make a contribution by developing an open-source MAP implementation that conform to ICE – the Medical Device Coordination Framework (MDCF)—developed jointly with researchers at the University of Pennsylvania and reported on in previous work [2], [3], [4]) and by developing sample artifacts and mock regulatory submission documents for ICE components.

In the ICE architecture, the Network Controller provides the MAP communication infrastructure to which medical devices and other hospital information systems are attached,

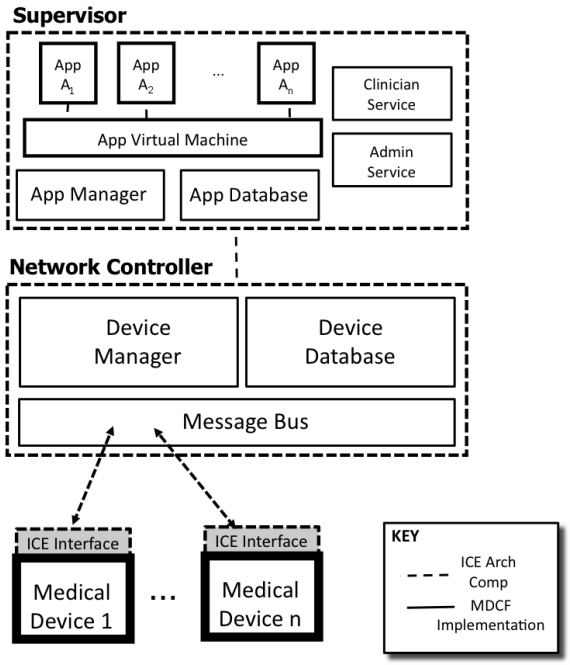

Figure 1. Integrated Clinical Environment (ICE) / MDCF Architecture

the Supervisor provides an execution platform for apps. The Network Controller provides high-assurance network communication capabilities that establish virtual "information pipes" between devices and apps running in the Supervisor. It exposes the ICE Interfaces of attached devices to Supervisor apps, and is agnostic to the intended use of the clinical apps that it supports. In the MDCF implementation of ICE (corresponding to the components in solid lines in Figure 1), a Device Manager component of the Network Controller maintains a registry of all medical devices that are 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 applying a key exchange protocol to determine if the connecting device has a valid digital certificate to indicate that has been previously certified to conform to its interface and has received regulatory approval.

In the MDCF, the Supervisor can be thought of as a virtual machine that hosts *Supervisor Apps*. We are currently working on ensuring that it provides separation/isolation-kernel-like [5] data partitioning (information cannot inadvertently leak between apps, and apps cannot inadvertently interfere with one another) and time partitioning (real-time scheduling guarantees that the computations in one app cannot cause the performance of another to degrade or fail). Each app declares *device types* indicating the types/capabilities of devices upon

which it depends. When a clinician initiates the app launch process, the Supervisor queries the Network Controller to determine if a device that meets those requirements is currently on the network and associated with the patient under consideration. If more than one device satisfies a particular app device requirement, the operator chooses a particular device to bind to the app. It is important to understand that the app, as it executes, gives rise to a system that satisfies the FDA technical/legal definition of a medical device, where the medical device is a composite entity formed from the logic of the app code, platform infrastructure components such as the Supervisor and Network Controller, and any devices and IT systems upon which the app depends. We refer to this composite device as a *platform constituted device*[2]. In regulatory terms, the app defines the *intended use* of the platform constituted device. Stepping back, one may observe that a MAP has a varying intended use (depending on the particular app that is running) and may simultaneously support multiple platform constituted devices (when more than one app is running simultaneously).

In previous work, we have identified a number of characteristics that distinguish MAPs from conventional medical devices and from embedded systems in other domains [6]. We briefly review some of the characteristics most relevant to the work presented in this paper.

Compositional construction, certification, and regulation: Instances of the ICE components of Figure 1 may be developed by different vendors, and will be individually (a) certified (by an independent certification authority) to comply with their ICE interface, and (b) subject to regulatory approval (*e.g.*, FDA's 510K). Note that this is a radical departure from the existing FDA practice of regulating only complete systems.

Integration after deployment, at the point-of-care: 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 assembled at the point of care by clinicians attaching devices to the communication infrastructure and launching apps that dynamically bind to devices with which they may have never been tested. The assembling is performed by clinicians who have no detailed technical expertise of device components, real-time application programming, nor distributed safety-critical systems engineering.

The above characteristics alone indicate that there are significant challenges to achieving safety and correctness. A number of engineering principles, inspired from related efforts in other domains such Integrated Modular Avionics (avionics domain) and the Multiple Independent Levels of Security (MILS) architecture (security domain), will need to be brought to bear to ensure safety/correctness. Foremost among these are: (a) the need for a robust architecture that will precisely define interface and strong functional and real-time requirements, (b) a rigorous third-party certification regime that establishes trust between between vendors of a particular component so that they can be confident that other components, upon which they depend, will function correctly, and (c) high-assurance infrastructure components that provide common platform services and space/time-partitioning for apps/devices and communication between these.

Space does not permit a complete exposition of how all safety concerns are addressed. For this paper, we focus in on the following key point. In the absence of the traditional role of a system integrator, (1) the *app itself defines the system integration* by specifying the interface capabilities of the devices and IT systems upon which it depends and by defining the intended use of the platform constituted device, (2) the *platform ensures correct system integration* by only allowing the app to execute when the platform can satisfy the execution and communication needs of the app (this is achieved via dynamic schedulability analysis) and when the devices attached to the network satisfy the device functional and real-time capabilities required by the app.

IV. Requirements Engineering with Apps

There are a number of potential sources to appeal to when designing artifacts capturing requirements for ICE apps. Traditional Software Requirements Specification (SRS) guidelines such as IEEE-830 are general purpose guidelines and fall short of the methodology and insights needed when dealing with safety-critical systems. Our principle source of inspiration has been the US Federal Aviation Administration (FAA) Requirements Engineering Management Handbook (FAA-REMH) written by Rockwell Collins engineers David Lempia and Steven Miller [7]. FAA-REMH focuses directly on recommended practices for requirements engineering for safety-critical embedded systems and provides illustrations using three systems, including a medical system–an Isolette Thermostat for a neonatal incubator. We found it a reasonable resource as it met two key criteria: 1) it is targeted at safety-critical embedded systems, and 2) it was written by experts in the field.

FAA-REMH lists eleven steps that developers should take in order to "progress from an initial, high-level overview

[2]In previous work, we have referred to this as a *virtual medical device*.

of the system... to a detailed description of its behavioral... requirements." The steps are:

1) Develop the System Overview
2) Identify the System Boundary
3) Develop the Operational Concepts
4) Identify the Environmental Assumptions
5) Develop the Functional Architecture
6) Revise the Architecture to Meet Implementation Constraints
7) Identify System Modes
8) Develop the Detailed Behavior and Performance Requirements
9) Define the Software Requirements
10) Allocate System Requirements to Subsystems
11) Provide Rationale

These steps were designed for requirements engineering in monolithic systems, and in this paper we consider necessary changes for a compositional, app-as-integrator requirements process. The subsections below follow the structure of our proposed format. For illustration, we consider an app that implements two "smart alarms" for pulse oximetry (PO Smart Alarm App) monitoring in a clinical context. We present excerpts of the requirements document; the complete document can be found on the MDCF website [8].

A. System Overview

FAA-REMH describes this first activity as "Develop[ment of] an overview of the system that includes a brief synopsis, describes all contexts in which the system will be used, and lists the main goals, objectives, and constraints of the system." Our modification of this step is the addition of a significant stipulation: that the system overview should include a clinical background and statement of clinical need.

1) Clinical Background: The clinical background should provide the medical and clinical background necessary for a developer who is not a clinician to understand and interpret the requirements. Though the inclusion of this section may depart from the standard organization of software requirements specifications, it is necessary to orient app developers who will likely not be experts in the clinical domain targeted by the app. Note that this may not be all the information that is provided, but domain experts should aim to provide a concise summary of the relevant clinical context.

In the PO Smart Alarm App requirements, this consists of approximately 1.5 pages describing the concept of blood oxygenation, medical conditions that benefit from PO monitoring, physiological causes of low oxygen saturation (SpO_2), basic technology underlying SpO_2 monitoring, normal and abnormal ranges for SpO_2 readings.

2) Clinical Need: This section should describe the specific clinical problems that the app is trying to address. Ideally, this section contains references to articles in the clinical literature that motivate the need.

> Existing pulse oximiters raise an alarm when a patient's pulse rate moves outside of limits that are set by clinicians on the PO device. Unfortunately, nuisance alarms (alarm events that occur when there is no physiological condition of concern) are common. An example is when a non-invasive blood pressure (NIBP) cuff is used on the same arm as a finger-based PO probe (*e.g.*, a broken arm necessitates attaching equipment for both devices on the healthy arm. When the NIBP activates at regular intervals to take a reading, the inflation of the cuff deprives the finger of oxygenated blood, which triggers a low SpO_2 alarm on the PO. These alarms can theoretically be avoided, but are not general enough to warrant direct support for by manufacturers—that is, existing pulse oximeters are not context-aware.
> Utilizing context-awareness could also increase precision of alarming by alarming in hazardous situations where a normal on-board PO alarm would not be activated. One possible context-aware alarm (implementable as a MAP application) concerns a patient's use of supplementary oxygen. Since SpO_2 values will be artificially high (and thus a PO device would be less likely to alarm when potentially harmful desaturations may be occurring with the patient) when a patient is on supplementary oxygen, a context-aware pulse-ox would take this into account by automatically adjusting down the SpO_2 value. A second possible app would be one that watches for rapid drops in a patient's SpO_2 values over time. If a patient's blood rapidly lost SpO_2 saturation (but that saturation was still above some safe lower threshold) an alarm could be raised. This type of monitoring is often useful in monitoring neonates and patients with sleep apnea.

Figure 2. PO Smart Alarm Clinical Need Summary

For the PO Smart Alarm example, we have 1.5 pages describing particular challenges of SpO_2 monitoring in certain clinical contexts. A high-level summary is given in Figure 2.

3) System Synopsis: Per FAA-REMH, this section provides a short textual synopsis of the app. The synopsis names the system, describes its purpose, and summarizes the system capability:

> This Integrated Clinical Environment (ICE) Pulse Oximetry Monitoring app provides Medical Device Data System (MDDS) displays of pulse oximetry device data, trend data for device readings, control of pulse oximeter device alarm settings, and derived alarms.
> The derived alarms enhance the functionality of conventional pulse oximeters by supporting more precise alarming for SpO_2 readings when a patient is on supplementary oxygen and by detecting rapid declines in SpO_2 that do not necessarily fall below the SpO_2 lower limit alarm provided by pulse oximeters.

This section should also include a diagram summarizing the clinician's view of the system. The diagram and supporting discussion should indicate how the ICE Platform would be configured to support (provide the run-time computational context) for the app. The diagram would indicate devices the app requires, and the specific sensors/actuators required for each device. The diagram would indicate what ICE

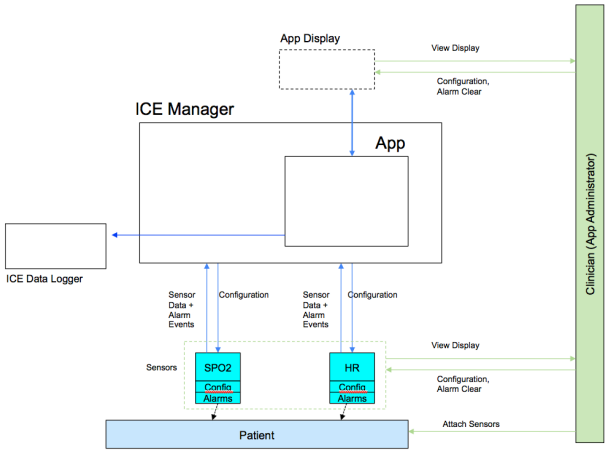

Figure 3. ICE instantiation for PO Smart Alarm app

external interfaces (*e.g.*, Electronic Medical Record, dosing databases) are required. The diagram would indicate what displays are supported by the app (*e.g.*, the ICE Supervisor display and any additional remote displays). The diagram would indicate at a rough level the basic interactions with the clinician and patient, though these will be spelled out in greater detail in the sections that follow.

Figure 3 is the corresponding diagram for our example. The bottom of the diagram indicates that the app requires a PO and the two primary physiological parameters from a PO (SpO_2 and heart rate). The app receives sensor data and alarm events for each parameter and can configure the device alarm limit parameters. There is a single display on the consumer console and the interactions that a clinician has with the display and attached devices are shown as well. In the full document, more details are provided concerning these interactions, as well as the interactions between apps and devices. This detailed discussion corresponds to the FAA-REMH step of identification of the external entities with which the system interacts and the nature of those interactions.

4) System Goals: Per FAA-REMH, this section offers a preliminary set of system goals that will be used to guide the specification of requirements. Goals are informal statements of the system stakeholders' needs. They are not requirements since they are not (generally) verifiable and do not provide enough sufficiently detailed information. However, they provide important guidance on why the system is being built and what is important to the stakeholders. *E.g.*, see Figure 4.

B. System Operational Concepts

Operational concepts are defined by the FAA-REMH as "scripts or scenarios describing how the system will be used." This step should: "[f]or all contexts in which the system will be used, define a black-box view of how the system will interact with its environment." This includes both normal and exceptional use cases.

- G1–warn clinician if patients using supplementary oxygen have low blood oxygenation
- G2–warn clinician if patient's blood oxygenation decreases rapidly
- G3–warn if blood oxygenation measurement is unreliable
- G4–display recent blood oxygenation measurements
- G5–display recent heart rate measurements
- G6–display current heart rate
- G7–display current blood oxygenation measurement
- G8–display parameters used to determine alarms
- G9–allow entry of parameters used to determine alarms
- G10–warn clinician if device detects fast or slow heart rate, by forwarding native alarms from the pulse oximeter device
- G11–warn clinician if device detects low SpO_2 by forwarding native alarms from the pulse oximeter device

Figure 4. PO Smart Alarm System Goals

- Related System Goals: G2
- Primary Actor: App
- Preconditions:
  - device on
  - sensor connected
  - enough SpO_2 measurements to form baseline from which current SpO_2 measurement is compared
- Postcondition:
  - patient's SpO_2 stabilizes
  - monitoring continues
- Main Success Scenario
  1) app calculates moving average of SpO_2 measurements
  2) app compares current SpO_2 measurement with moving average
  3) if current SpO_2 is below moving average at least as much as the rapid SpO_2 decline limit, app announces rapid SpO_2 decline both visually and audibly through the supervisor user interface
  4) clinician responds to patient need taking medically-necessary actions
  5) patient's SpO_2 stabilizes
- Alternate Exception Scenarios
  - clinician changes rapid SpO_2 decline limit
  - clinician detaches sensor and turns off device

Figure 5. PO Smart Alarm Exceptional Use Case

There is no specialization to MAP apps to be done for this step. An example of an exceptional use-cases for the PO Smart Alarm app (there are two normal and six exceptional use cases) is given in Figure 5.

C. App Interfaces

This section addresses the compositionality of MAP apps by identifying the device and IT system interfaces on which the app depends. The most important interfaces are the device (sensor/actuator) interfaces. The device interface requirements include a precise statement of the physiological parameters—ideally, in terms of standard medical nomenclature.

There is no direct correspondence between this section and the FAA-REMH. However, there are many similarities between this section and the FAA-REMH Section 2.4.1 on "Identifying Environmental Assumptions" in which the

data type, range, precision, and units of the monitored and controlled variables are defined. The FAA-REMH notes that "Every system makes specific assumptions about the environment in which it will operate." These environmental assumptions are defined as "the assumptions about the environment on which a system depends for correct operation." It continues, explaining that "These can be specified as a mathematical relationship between the controlled and monitored variables... [which] can be as simple as [their] types, ranges and units."

An example of these requirements for the PO Smart Alarm app would be the following declared dependency on the SpO_2 sensor value.

Blood Oxygenation Saturation (SpO_2)		Physical Interpretation
Data Type:	Integer	Fraction of hemoglobin that is oxyhemoglobin expressed as percentage
Units:	Percentage	
Range:	0 to 100	
Period:	1 second	

Note the need to capture real-time constraints (*e.g.*, period). In general we anticipate ICE device interfaces including additional constraints such as temporal ordering (*e.g.*, type state constraints) on device commands, role-based security access controls of data steams and command ports. In more sophisticated apps that perform control of devices, other contract information would specify how the device should change its state to move to a fail-safe mode upon accidental disconnect from the Network Controller.

D. Functional Architecture and Requirements

The FAA-REMH notes that "To be usable, a requirements specification for a system of any size must be organized in some manner." To that end, it states that "To enhance readability of the requirements and to make them robust in the face of change, the requirements [should be] organized into functions that are logically related with minimal dependencies between functions."

There are no significant changes between the FAA-REMH version of this step and the one we eventually settled on. Our notion of app development is based on a component architecture, so the suggested functional decomposition aligns nicely with the organization of the app into components. Our example app has a number of components, as well as incoming and outgoing channels (see Figure 6 for a high-level summary of the structure).

This section also includes the detailed functional requirements for the app. For example, here is a fragment

> When SupplOxyAlarmEnabled is true, the SupplOxyAlarm function shall generate an unlatched alarm when the SpO_2 device parameter decreased by SpO2LevelAdj is less than the device parameter SpO2LowerLimit.

An important element of this section is the presentation of requirements for the app Supervisor User Interface (SUI) display that the clinician sees on the Supervisor Clinical Console. The intent here is that the requirements writer provides a mock-up of the display (*e.g.*, Figure 7)—The mock-up shows what information must be shown, but does not constrain the actual implementation of the display. The various numbers/labels in the display mockup refer to an even deeper layer of requirements. These are enumerated, and include references to the element(s) affected. For example:

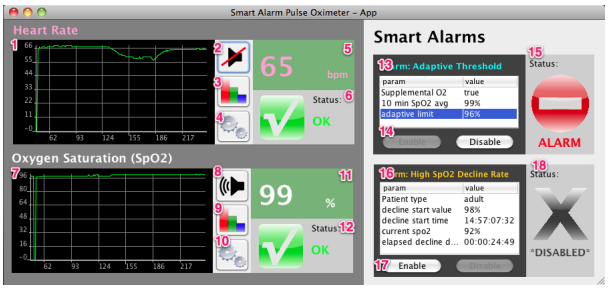

Figure 7. App SUI Mockup

> - The SUI shall display a visual cue reflecting the current state of both the Heart Rate Lower Limit and the Heart Rate Upper Limit Alarm (element 6).
> - The SUI shall display a visual cue reflecting the current state of the SpO_2 Lower Limit Violated Alarm (element 12).
> - The SUI shall display a visual cue reflecting the current state of the Supplemental Oxygen-Adjusted Low SpO_2 alarm (element 15).

V. Conclusion

We have summarized one possible requirements document organization for ICE apps. Our primary driving example has been the PO Smart Alarm app. We are using the same format to capture requirements for more challenging apps such as the patient-controlled analgesic closed loop safety interlock that we have previously implemented [3]. In contrast to the PO Smart Alarm app, these other apps have significant safety requirements.

In ongoing work on the MDCF project, we are building the next generation of our app development environment based on use of the Architecture Analysis and Design Language (AADL) [9], including an Eclipse-based requirements tool supporting the AADL Requirements Definition and Analysis Language (RDAL) Annex [10][3] developed by researchers at Université de Bretagne Sud (France) and Taif University (KSA). The Eclipse tool for RDAL provides a number of capabilities that allow RDAL requirements to be traced to AADL models, and we are looking forward to exercising these capabilities in the very near future.

There are several other envisioned development artifacts related to ICE app requirements including clinical workflow

[3]Adoption of RDAL as a standard is expect in late 2012 as an annex designation to SAE AS5506B.

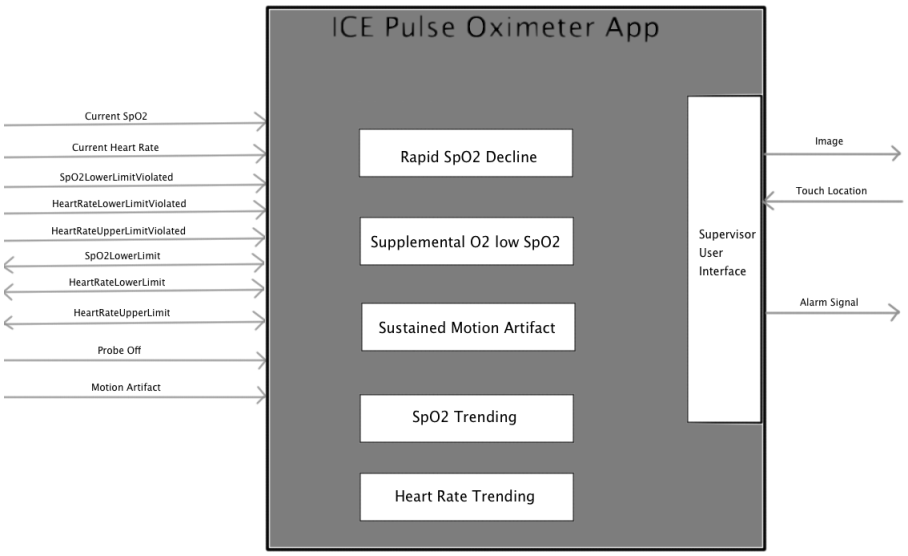

Figure 6. App Sub-Functions

descriptions that specify workflows that clinicians are expected to follow when interacting with ICE apps, infrastructure, and devices. These workflows could be formalized in process description languages like Little JIL [11], which has been used to capture non-MAP clinical workflows.

Finally, we continue to work on other artifacts such as hazard analysis, assurance cases, and mock regulatory submissions associated with these apps.

ACKNOWLEDGMENT

This authors thank Andrew King, FDA engineers Paul Jones and Sandy Weininger, and NIH/NIBIB Quantum Intranet team members Mike Robkin and Ed Ramos for their extensive comments on the PO Smart Alarm ICE App requirements. This work is supported in part by the National Science Foundation under Grants #0734204, 0932289,1065887, and by the NIH/NIBIB Quantum program.

REFERENCES

[1] *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.

[2] 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.

[3] 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*. ACM, 2010.

[4] "The Medical Device Coordination Framework project website," http://mdcf.santos.cis.ksu.edu, 2012.

[5] J. Rushby, "The design and verification of secure systems," in *8th ACM Symposium on Operating Systems Principles*, vol. 15(5), 1981, pp. 12–21.

[6] J. Hatcliff, A. King, I. Lee, A. Fernandez, A. McDonald, and E. Vasserman, "Rationale and architecture principles for medical application platforms," in *Proceedings of the 2012 International Conference on Cyberphysical Systems*, 2012.

[7] D. Lempia and S. Miller, "DOT/FAA/AR-08/32. requirements engineering management handbook," Federal Aviation Administration, 2009.

[8] J. Hatcliff and B. Larson, "ICE pulse oximiter smart alarm app requirements," http://mdcf.cis.ksu.edu, SAnToS Research Laboratory, Kansas State University, 2012.

[9] "Architecture Analysis & Design Language," http://www.aadl.info, 2012.

[10] D. Blouin, E. Senn, and S. Turki, "Defining an annex language to the architecture analysis and design language for requirements engineering activities support," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2011*, Aug. 2011, pp. 11 – 20.

[11] A. Wise, "Little-JIL 1.5 language report," University of Massachusetts, Amherst, MA, USA, Language Report, October 2006.