Demonstration of a Medical Device Integration and Coordination Framework*

Andrew King, Sam Procter†
Dan Andresen, John Hatcliff‡, Steve Warren
Kansas State University
{aking,samuel3,dan,hatcliff,swarren}@ksu.edu

William Spees, Raoul Jetley
Paul Jones, Sandy Weininger
US Food & Drug Administration
{William.Spees,Raoul.Jetley,PaulL.Jones,
Sandy.Weininger}@fda.hhs.gov

Abstract

This tool demonstration presents a framework for integrating and coordinating the activities of medical devices. The framework uses a publish-subscribe framework for communicating with and controlling devices and a model-driven component-based development environment for rapid implementation of device coordination tasks. A multi-faceted graphical user interface supports activities such as device/driver registering and installation, model-based development of device integrations, and monitoring system activities/performance. The framework also includes a control/display environment that clinicians would use to (a) display integrated information pulled from multiple devices and (b) launch and interact with device coordinations that automate clinical workflows. The distribution of the framework includes a collection of mock medical devices, and instructions for integrating real devices. The codebase is freely available under an open source license.

1. Introduction

Medical devices historically have been monolithic units – developed, validated, and approved by regulatory authorities as stand-alone entities. Modern medical devices increasingly incorporate connectivity mechanisms that offer the potential to stream device data into electronic health records, integrate information from multiple devices into single customizable displays, and coordinate the actions of groups of cooperating devices to realize "closed loop" scenarios and automate clinical workflows. However, it is not clear what middleware and integration architectures may be best suited for these possibly numerous scenarios. More troubling, current verification and validation techniques used in the device industry are not targeted to assuring groups of integrated devices.

To facilitate industry, academic, and government exploration of these issues, we are developing an open Medical Device Coordination Framework (MDCF) for designing, implementing, verifying, and certifying systems of integrated medical devices. This framework uses publish-subscribe architectures and component-based model-driven development along with standards-compliant open-source code-bases to provide a full featured and scalable system for integrating both real and mock medical devices in realistic clinical applications. This tool demonstration will explain current challenges that are driving this research, describe the architecture of the framework, and provide live walkthroughs of the primary MDCF modules.

This tool demo paper supplements our ICSE 2009 Experience track paper [2]. The Experience paper gives a detailed discussion of benefits and challenges of medical device integration/coordination and describes our experience in using the MDCF in multiple configurations that correspond to realistic clinical deployments. By presenting this material at ICSE, we seek to (a) directly engage the software engineering community with initial experience and challenge problems associated with this emerging paradigm of medical systems and (b) overcome community barriers that have previously inhibited interactions between the software engineering researchers/practitioners, industrial medical device developers, and government regulatory authorities. The MDCF is available for public download at [3].

2 Examples

Our tool demonstration will illustrate how the MDCF can support the following examples of device integration.

Workflow Automation: Integration of a X-ray machine and ventilator is an example of useful coordination that addresses the problem of acquiring chest X-rays of patients on ventilators. 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. These risks can be minimized by automatically coordinating the actions of the X-ray imaging device and the ventilator: specifically, the ventilator can identify when the lungs are at full inhalation or exhalation (and thus experiencing minimal motion) so that the X-ray image can be automatically captured at the optimal point in time.

Integrated medical device displays: A typical hospital room in an intensive care ward hosts a number of stand-alone devices. Many modern rooms are integrated with an Electronic Health Record (EHR) database to log clinical activ-

*This material is based upon work supported by the National Science Foundation under Grants # 0454348 and 0734204 and by the Air Force Office of Scientific Research.

†Author's current affiliation: University of Nebraska, Lincoln
‡Corresponding Author.

ities, lab data, treatment plans, and information for patient billing. Connections to drug dosing databases may also be available to facilitate correct drug dispensing. In such contexts, a number of factors reduce efficiency, degrade the quality of the patient's encounter, and increase error likelihood. Each device in the room has its own user interface, and these interfaces are non-uniform – potentially leading to mental overload and confusion on the part of the caregiver. The MDCF allows traditional medical devices to be viewed as data producers that publish periodic or streaming data to several types of data consumers. An EHR database serving as a data consumer allows information from individual devices to be integrated directly into the EHR. A single "heads up" display serving as a data consumer takes information from multiple devices and an EHR database (in this case, acting as a data producer) and formats it on one or more large monitors near the patient bed.

3 Primary Functional Components

Device and Driver Database: A MySQL database stores information about device types (e.g., X-Ray, Ventilator, Pulse Oximeter, Blood Pressure Cuff) and specific models of devices supported by the framework. Each device model has a driver associated with it. Due to the safety-critical nature of the application, connection to the system is limited to approved devices with pre-installed drivers. The database stores a list of approved devices identified by unique identifiers based on, e.g., MAC addresses. Our demo will illustrate the MDCF console for maintaining this device information. In addition, we will overview the construction of drivers for both real and mock devices.

Middleware Layer: MDCF uses an open-source the Java Messaging Service (JMS) implementation to support publish-subscriber style communication. On top of JMS, we have developed interfaces to support common clinical data formats such as Health Level 7 (HL7) and the DICOM medical imaging format. [2] reports on a variety of experiments that confirm the effectivness of MDCF for communicating sample clinical data conforming to these formats. Our demo will show a monitoring console that allows one to observe the flow of events and data across the JMS infrastructure.

Model-based Coordination Development: We have built an integration scenario development environment[3] for eclipse based on our Cadena framework [1]. The demonstration of this environment will be the primary focus on the demo. Cadena provides component-based meta-modeling that enables us to define a domain-specific language of components for building device integration scenarios. Given a meta-model of the component language, Cadena generates a *component interface* editor that allows one to define component types and a *system scenario* editor that allows one to allocate and connect component instances to form an executable system. Cadena's rich type system allows one to define different type languages for component ports that capture specific properties of data communicated between components. Cadena provides a notion of "active typing" that continuously checks for type correctnesss as a system scenario is constructed in the graphical scenario editor.

We have also built a type system defining the abstract 'shapes' of different component types such as (*DeviceDriver*, *DataTransformer*, *DataSink*) found in integration scenarios. Given a Cadena type signature for an MDCF component, autocoding facilities generate a Java skeleton/container for the component. The skeleton contains all logic required by the framework to enable the component implementation to connect to the framework as a framework component (this includes automatically generating the subscription assignment and publishing logic). The component developer then only needs to implement the "business logic" – the code that processes medical information (such as a data transformer or rendering routine) or device access logic (interaction with actual device sensor hardware). Similar in spirit to CORBA Component Model (CCM)'s deployment and configuration infrastructure, MDCF can also analyze a Cadena coordination scenario model and generate a MDCF specificiation file. This XML-based file is used by the MDCF to locate the appropriate MDCF component class files and execute the coordination scenario.

Clinician Console: A caregiver chooses a coordination scenario to run from the coordination library using the Clinician Console. This console guides the caregiver in selecting devices of the appropriate type from among those that are currently connected to the framework. The console supports a variety of data panels that can serve to integrate data from one or more devices or the EHR database and allow the caregiver to guide device coordination activities. Our demo will follow the typical workflow of a caregiver as they select, execute, and oversee coordination scenarios.

4 Conclusion

We believe the MDCF framework provides a rich setting for experimenting with design, development, and verification techniques in a challenging real-world context. The contents of this paper should not be interpreted as an endorsement by the FDA of any particular technology, software infrastructure, or direction for regulatory policy. However, we expect experience with frameworks like the one presented here to provide science-based input to ongoing regulatory policy and standards development efforts.

References

[1] A. Childs, J. Greenwald, G. Jung, M. Hoosier, and J. Hatcliff. CALM and Cadena: Metamodeling for component-based product-line development. *Computer*, 39(2):42–50, February 2006.

[2] A. King, S. Proctor, D. Andresen, S. Warren, J. Hatcliff, 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 (ICSE 09)*, 2009. To appear.

[3] Medical Device Coordination Framework (MDCF) – Kansas State University. `http://mdcf.santos.cis.ksu.edu/`.