STEPSTONE: An Intelligent Integration Architecture for Personal Tele-Health

Sumi Helal*, Raja Bose, and Chao Chen
Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL, USA
sumi.helal@gmail.com, raja.bose@nokia.com, cchen@cise.ufl.edu

Andy Smith and Scott de Deugd
IBM, Research Triangle Park, Raleigh, NC, USA
andybs@us.ibm.com, dedeugd@us.ibm.com

Diane Cook
School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA cook@eeecs.wsu.edu

Abstract
STEPSTONE is a joint industry-university project to create open source technology that would enable the scalable, “friction-free” integration of device-based healthcare solutions into enterprise systems using a Service Oriented Architecture (SOA). Specifically, STEPSTONE defines a first proposal to a Service Oriented Device Architecture (SODA) framework, and provides for initial reference implementations. STEPSTONE also intends to encourage a broad community effort to further develop the framework and its implementations. In this paper, we present SODA, along with two implementation proposals of SODA’s device integration. We demonstrate the ease by which SODA was used to develop an end-to-end personal healthcare monitoring system. We also demonstrate the ease by which the STEPSTONE system was extended by other participants – Washington State University – to include additional devices and end user interfaces. We show clearly how SODA and therefore SODA devices make integration almost automatic, replicable, and scalable. This allows telehealth system developers to focus their energy and attention on the system functionality and other important issues, such as usability, privacy, persuasion and outcome assessment studies.

Category: Smart and intelligent computing; Ubiquitous computing

Keywords: Connected health; Health telematics; Open source healthcare IT; Service oriented device architecture

I. INTRODUCTION

In this paper, we present STEPSTONE, a joint effort between industry and academia to examine existing technological barriers that limit the commercialization and proliferation of wireless health concepts and urgently needed health care delivery alternatives. We follow a storyboard approach, in which we describe some aspects of the daily life of an elderly person living alone, to introduce STEPSTONE. We introduce and position STEPSTONE through the story lines, and invite other researchers working on similar areas of concern to join our effort and to use and improve what we have made publicly available to this research community. Charlie’s story below can be found in video production format on the web [1].

A. A Smart World for Charlie

This is a story about Charlie, a frail, elderly gentleman with a
serious heart condition. For quite some time - and with the help and encouragement of his son David - Charlie has been participating in a wellness program that tracks his weight, diet, activity level and important vital signs, including blood pressure. So far, Charlie has been supported by several devices at his “smart” home that features a weight scale, a barcode/radio frequency identification (RFID) food scanner and several devices that gauge occupant activity levels.

Until now, Charlie’s Smart House has provided for the sharing of information with select family members, so David can keep track of his father’s activities. However, Charlie has been experiencing changes in his health and, therefore, his medications, he was recently forced to travel to his local hospital twice a week for checkups, including blood-pressure monitoring.

These visits have become a challenge, since Charlie is no longer able to drive. They cause major changes in Charlie’s routines. They tie David down, since he must keep track of Charlie’s appointments, call his father to remind him about them and ensure that his father gets on the community bus that picks him up and transports him to the hospital.

Of course, there’s more. Once he has entered the hospital, Charlie must wait in long lines - waits that are often extended when Charlie must vacate his place for rest-room breaks. Once he’s had his appointment, including his blood-pressure checks, Charlie must wait again - this time for another bus to return him home. For Charlie, these aren’t simple trips - they are ordeals. Worse yet, the day goes by and David has no idea how his father is faring, either in terms of health status or whereabouts. With his father becoming more forgetful, the situation grows riskier.

Charlie and his loved ones are suffering. There’s too much room for things to go wrong, too little monitoring, and too much risk in pursuing treatments themselves. The family is too often left in the dark. Charlie doesn’t understand what good the healthcare delivery he receives is doing and can do for him - especially in the event of an emergency.

B. A Smarter World for Charlie

Clearly, Charlie’s situation and needs are shared by millions. For years, the building blocks of the solutions that could alleviate his problems have not only existed, but they have been deliverable on a one-to-one basis. What has been missing and what is urgently needed is driving commonality in community/ ecosystem and driving standards around these key points of interoperability with the home, family doctor’s office, providers, and healthcare enterprises.

Personal medical devices are highly available and affordable today. Their accuracy (compared to clinical equipment) is steadily improving. What they lack is an open and interoperable function/use model that allows their utility to easily extend to third party providers. The STEPSTONE project presented in this paper combines an open standard approach, wireless sensor middleware technology and commercial off-the-shelf personal medical devices - thereby creating new, repeatable, flexible possibilities and opportunities in Healthcare delivery.

STEPSTONE proposes the application of an open Service Oriented Device Architecture (SODA) as a framework to enable medical device manufacturers, as well as HealthCare information technology (IT), to reach out to each other in an interoperable, friction-free fashion. This will allow third party providers, such as wellness centers, Nurse-At-Home, or local hospitals, to develop applications and define services without having to expend effort in integrating devices to their “enterprise data buses”.

Charlie’s smart home has become smarter under the STEPSTONE project. All Charlie’s medical devices and activity monitors have been made SODA compliant devices. Their integration has been enabled by the standards-based architecture proposed by STEPSTONE. This allowed Charlie’s activities and vitals, including the blood pressure measurements he takes by himself at home, to be made available remotely to his local hospital, other care givers, and of course to his son. David’s simple wellness indicator on his cell phone now alerts him to a text message on his phone, showing if his father’s blood pressure is high or if it is under control. Charlie and David can work more closely with healthcare providers, and finally regain peace of mind over the heart condition onset, and cost, time, effort, errors, and worries have been cut or significantly reduced.

C. Paper Outline

STEPSTONE is a case study in the domain of personal health of a wider effort to promote device integration and programmability, launched jointly by IBM and the University of Florida. In Section II, we describe the SODA framework proposed by what is now expanding to be the SODA alliance. We describe the SODA framework goals and elements, and then focus on one key impediment within SODA, which is device languages and integration infrastructure. We present two SODA proposals for Device integration: The DeviceKit presented in Section III, and the Device Description Language presented in Section IV. In Section V, we present the STEPSTONE project and its overall system architecture showing all of the architecture component layers and how it works. We present specific use cases centered on the use of blood pressure and actigraph devices. We describe the case studies and present their assessment consoles. Section VI presents related work. Conclusions and future work are presented in section VII.

II. SODA ARCHITECTURE AND FRAMEWORK

Monitoring and controlling a physical environment has long been possible through electronic devices, ranging from basic sensors and actuators to complex digital equipment and controllers. However, as the type of available devices continues to grow, the number of device interfaces, connections, and protocols are multiplying at a corresponding rate. System developers are finding that integrating devices into large-scale enterprise systems is both daunting and expensive. We proposed the SODA framework [2, 3] to eliminate much of the complexity and cost associated with integrating devices into highly distributed enterprise systems such as remote health monitoring systems. SODA utilizes concepts of a Service Oriented Architecture (SOA) and leverages existing and emerging standards from the embedded-device, healthcare and IT domains [4-6] to enable seamless cost-effective integration of sensing, monitoring, and actuation devices into the enterprise data bus.
STEPSTONE: An Intelligent Integration Architecture for Personal Tele-Health

Sumi Helal et al. 271 http://jcse.kiise.org

A. Making the Case for SOA

SOA provides the ability to map business processes and events across an open communication infrastructure, sharing data and functionality in an open, flexible manner. SOA aims at integrating business systems through a set of services that can be reused and combined to address changing business priorities. Services are software components with well-defined interfaces, and they are independent of the programming language and the computing platforms on which they run. We can apply SOA’s modular, componentized nature to traditional enterprise services and to events and services from all types of data sources and devices, ranging from low-level sensors and actuators to more sophisticated devices, such as diabetes and blood pressure monitors.

Conventional approaches to device integration often center on custom interface software communicating to enterprise applications through a variety of IT middleware and application program interface (API) technologies where applications encapsulate both instructions pertaining to specific device operations and the business logic that drives them. However, this kind of hard integration leads to massive re-engineering efforts when application features have to be updated regularly during the lifecycle of the system or the specific devices being utilized are changed over time.

Conversely, SOA can provide the basis for a softer integration approach, where low-level device-specific operation logic is separated from application-specific business logic. This coupled with SOA mechanisms for dynamically discovering and accessing services, building and sharing service interfaces and loosely coupled messaging models, can enable the creation of enterprise systems that are inherently more flexible and adaptable to change both from the perspective of applications and devices.

B. SODA Approach

The SODA approach to designing and building distributed software, utilizes SOA concepts to integrate a wide range of physical devices into distributed IT enterprise systems. At the most basic level, as Fig. 1 shows, SODA bridges the embedded and business IT domains by enabling application developers to interact with sensors and actuators, just as they do with business services in today’s enterprise SOAs. For example, a reading from an RFID reader providing user credentials via RFID tags could interact with a login event, just as a client application can provide credentials.

SODA focuses on the boundary layer between the physical and digital realms. A sensor device, such as a digital thermometer or a blood pressure monitor, can translate a physical phenomenon into corresponding digital data. Conversely, an actuator, such as a medication reminder or alarm beacon, can translate a digital signal into a physical phenomenon. Sensors and actuators are combined either physically or conceptually in a large scale distributed enterprise system to create complex devices and services, such as remote at-home health monitoring systems or indoor climate control systems.

SODA aims to:
- Provide higher-level abstractions of the physical device.
- Insulate enterprise system developers from the ever-expanding number of device interfaces, protocols and low-level device-specific operational details.
- Bridge the device hardware and software realms, representing devices as high-level software services that can be dynamically discovered, accessed and controlled.

Conversely, SOA can provide the basis for a softer integration approach, where low-level device-specific operation logic is separated from application-specific business logic. This coupled with SOA mechanisms for dynamically discovering and accessing services, building and sharing service interfaces and loosely coupled messaging models, can enable the creation of enterprise systems that are inherently more flexible and adaptable to change both from the perspective of applications and devices.

B. SODA Approach

The SODA approach to designing and building distributed software, utilizes SOA concepts to integrate a wide range of physical devices into distributed IT enterprise systems. At the most basic level, as Fig. 1 shows, SODA bridges the embedded and business IT domains by enabling application developers to interact with sensors and actuators, just as they do with business services in today’s enterprise SOAs. For example, a reading from an RFID reader providing user credentials via RFID tags could interact with a login event, just as a client application can provide credentials.

SODA focuses on the boundary layer between the physical and digital realms. A sensor device, such as a digital thermometer or a blood pressure monitor, can translate a physical phenomenon into corresponding digital data. Conversely, an actuator, such as a medication reminder or alarm beacon, can translate a digital signal into a physical phenomenon. Sensors and actuators are combined either physically or conceptually in a large scale distributed enterprise system to create complex devices and services, such as remote at-home health monitoring systems or indoor climate control systems.

SODA aims to:
- Provide higher-level abstractions of the physical device.
- Insulate enterprise system developers from the ever-expanding number of device interfaces, protocols and low-level device-specific operational details.
- Bridge the device hardware and software realms, representing devices as high-level software services that can be dynamically discovered, accessed and controlled.
- Enable multiple rich applications to share the same set of deployed devices.
- Empower IT developers to modify and distribute application-specific business logic any time without dealing with or affecting device-specific operations.

A SODA implementation comprises four main components, as shown in Fig. 2. Device adapters talk to device interfaces, protocols, and connections on one side and present an abstract services model of the device on the other. The application layer interfaces with this services model providing a developer the capability to distribute application-specific logic to the edge of the network before forwarding the messages to the bus adapter. The bus adapter moves device data over network protocols by mapping an application service to the specific SOA binding mechanism used by the enterprise. The device service registry provides for the discovery and access of SODA services. In the next few sections, we go into further details by describing two

![Fig. 1. Modeled as services, device access, and control can be made available to a wide range of enterprise applications using service-oriented architecture mechanisms.](image1)

![Fig. 2. Service-oriented device architecture (SODA) implementation.](image2)
alternative real-world implementations of the SODA framework, comparing and contrasting their various features and showcasing some of the wireless remote health monitoring applications they have enabled.

III. DEVICE INTEGRATION USING THE DEVICEKIT

The first approach leverages an Eclipse open source project, DeviceKit, which provides tooling and runtime capabilities enabling rapid prototyping and development of Java code to interface with any hardware device. It acts as an isolation layer between the hardware device and application and is a key accelerator in the implementation of SODA built on top of the Open Services Gateway initiative (OSGi) SOA framework [7]. DeviceKit has a platform-independent markup language-driven specification model, where each device is described using the XML-based DeviceKit Markup Language (DKML). DKML allows the specification of hardware signals and protocols required for interfacing with a device, as shown in Fig. 3. The DKML description is converted into appropriate Java code by DeviceKit that automatically generates adapters to access and control the device as a software service.

A. DeviceKit Abstraction Layers

The DeviceKit runtime abstracts a physical device into several software service layers: STEPSTONE leverages the Connection Layer, Transport Layer, and Device Layer, as shown in Fig. 4. The layers are loosely coupled with mutually exclusive sets of functionalities, thereby allowing an application developer to choose appropriate service options from each layer that can then be dynamically linked together at run-time using OSGi services mechanisms.

The Connection Layer is the lowest layer in the device abstraction hierarchy and is responsible for reading and writing byte streams of data to the hardware device over a specific connection interface. DeviceKit supports dynamic installation of connection adapters and provides native support for the following connection interfaces: IP, Bluetooth, File, URL, and RS-232 Serial Port.

The Transport Layer deals with the specific messaging requirements of the device and corresponds to the Protocol Adapter component of the SODA framework. It is responsible for multiplexing byte streams received from the Connection Layer into messages and de-multiplexing messages into byte streams.

B. Integrating a Device Using DeviceKit

Integrating a device by converting it into a software service using DeviceKit (Fig. 5) consists of a number of steps, some of which are automated, whereas other require manual customization.

First, the device protocol and operational logic is specified in a DKML file. The DKML file is then parsed by DeviceKit that generates two service objects (or bundles in OSGi terminology) namely, the Device Bundle that implements the Device Layer logic and the Transport Bundle that implements the Transport Layer Logic. The Connection Layer Bundle has to be hand coded by the application developer, depending on the type of network/connection interface used to communicate to the device hardware. Typically, the hardware device driver is provided by the manufacturer and the application developer writes Java code to interface with that driver. However, DeviceKit currently provides native support for device connections over IP, Bluetooth, file, URL, and RS-232 Serial Port to ease this aspect of the integration process.
Finally, the code for the Transport Layer service bundle, which was automatically generated by DeviceKit, might need to be manually customized, depending on the type of Connection Layer service used. However, customization is not required when one of the protocols mentioned above is used. Conventional device drivers are not required, and once this process is complete, the hardware device is ready to be dynamically discovered, accessed and controlled as a high-level software service by multiple applications.

C. DeviceKit Availability

DeviceKit has been made available to all interested system and application developers as part of the Eclipse Open Health Foundation (OHF) and can be downloaded free from [8]. Furthermore, people interested in obtaining the STEPSTONE reference application built using DeviceKit should access [9], where detailed download and installation instructions are provided for a complete end-to-end remote health monitoring solution.

IV. DEVICE INTEGRATION IN SODA USING THE DEVICE DESCRIPTION LANGUAGE

The Device Description Language (DDL) is the second proposed implementation of the SODA framework developed by the University of Florida. Working in concert with the ATLAS Sensor Platform [10, 11] and the ATLAS middleware, DDL provides an easy and convenient solution to automatic device integration in a way that is distinct from DeviceKit. DDL is designed to incorporate complex devices, such as blood pressure monitors and GPS navigators, and primitive pinhead sensors and actuators. While it is possible to directly establish end-to-end connections between a complex device and a computer using serial or USB cables, it is practically impossible to do so in the case of primitive devices, as many of them only have digital/analog pins and do not have a networking capability. Therefore, DDL assumes that all devices are connected through sensor platforms that can handle data communications.

In this case study, we implemented device integration using the ATLAS Sensor Platform (Fig. 6). ATLAS provides physical nodes to connect heterogeneous devices with various interfaces and a runtime environment to access services and compose applications. The hardware of the ATLAS platform has a modular design consisting of three stackable and swappable layers: a device connection layer that provides various connectors to devices, a processing layer that runs the firmware that controls the operation of the node, a communication layer that handles data transfer over various networks, such as wired 10BaseT Ethernet, 802.11b WiFi, ZigBee, and Bluetooth. The ATLAS middleware uses OSGi as its basis, providing many service discovery and configuration mechanisms to support the creation of pervasive applications.

The software toolkit for DDL consists of two parts: a language schema that defines the syntax of the language, and a lan-
guage processor that converts DDL device descriptors to service bundles.

A. Device Description Language Schema

DDL is a markup language using XML encodings that provides high readabilities to both human and machines [12]. Beyond the basic syntactic constraints imposed by XML itself, we provide a DDL schema to further define the constraints on the structure and the content of a DDL document. Fig. 7a shows a snippet of the schema. The DDL language processor verifies each instance of DDL to ensure it conforms to the schema.

Within the SODA framework, a DDL document is a descriptor file for a single device (Fig. 7b). It contains information required for service registration and discovery, such as the name and model of the device and the description of its functions. It also describes operations of devices. DDL models a device from a data-oriented perspective, where each operation of a device is a collection of input/processing/output function chains. This view is consistent with the service-oriented architecture, within which each device is represented by a service that provides an interface to its properties and is shielded from its internal mechanism. Hence, the focus of DDL is describing the interface of devices.

DDL describes the communications between a device and its corresponding service entity using ‘Signals’. For example, a signal can be a voltage output from a pin, a number from an analog to digital converter (ADC), or a byte stream from a serial port, conforming to a certain well-defined protocol. Depending on the type of device, signals can be either unidirectional or bidirectional. ‘Readings’ are the meaningful information that the service entity receives from the associated device. It usually specifies a conversion or a parsing method that converts raw signals to semantically meaningful data. For example, a temperature sensor outputs a signal ‘0010000000’, a 10-bit value from the ADC. It is then converted to 325 degrees, a reading in the unit of Centigrade. In DDL, the combination of signals and readings together define the interface of a device.

B. Device Description Language Processor

In our current implementation, the DDL Language Processor consists of a DDL parser and an ATLAS service bundle generator. The parser checks if the DDL document conforms to the defined schema and retrieves information from the DDL descriptors. The bundle generator creates Java code for the device based on the information, and then builds the compile code into software bundles. Once a bundle is created, it will be uploaded to the ATLAS bundle repository and its description of the device service will be registered.

When a device comes online, the ATLAS node that connects the device will negotiate with the middleware. After the middleware is notified of the device’s presence, it will activate the corresponding device service bundle from the bundle repository. Applications are then able to locate and use the service to interact with the device.

1) DDL Availability: The DDL schema and source code of the language processor have been released and are accessible online [13] to invite and encourage discussions and participation from the research communities. A DDL Programmer Manual, which documents the detailed specifications of the language and provides programming instructions to software developers, is also available.

V. STEPSTONE

Fig. 8 shows the architecture of the STEPSTONE system. Starting from the bottom, personal medical devices are connected through available physical interfaces to Atlas nodes (e.g., through RS-232, USB, IR, Bluetooth). Atlas nodes communicate directly or through a gateway to the Internet based on a wireless or wired IP based connection. Atlas nodes represent the devices and provide a standardized messaging and program-
matic interface to the rest of the system. On the network side, a backend system consists of Atlas middleware server that is able to communicate with Atlas nodes, along with an application server through which web and other portals interact with the Atlas device network. The Atlas middleware is shown with both DeviceKit and DDL frameworks. Currently web and mobile phone portals are supported. Other portals such as automated voice response (AVR) can be easily added.

Fig. 9 shows a commercially available Blood Pressure monitoring device connected using a serial interface to an Atlas sensor platform node. The sensor node connects to the back end system through an Ethernet interface and a router. The device has been integrated using both the DeviceKit and DDL.

A. DeviceKit and DDL: Pros and Cons

DeviceKit is a powerful modeling tool for complex devices. Its three abstraction layers dissect separate key aspects of the device interfaces, allowing dynamic integration and reconfiguration of device support as the needs of the healthcare monitoring applications change. Functions, such as filtering, preprocessing and alerts, can be distributed to the appropriate node in the network. Hence, the Java code automatically generated by DeviceKit is clearly structured based on function. However, the integration process is not completely automated, since customization is required by the system developers to match the business rules, or when specific transport protocols and connection interfaces are added.

DeviceKit allows end-to-end connections to be established directly between a complex device and a backend server, without routing via intermediate agents, such as sensor platforms. While direct connectivity is a convenient and economic solution for device integration, it does require the device to have some level of wire protocol. Many primitive sensors and actuators simply do not have the interface support required to communicate with a computer. In this case, DeviceKit would require some type of hardware input/output adapter. Even if a sensor platform adapter is utilized, the programming overhead of using DeviceKit must be weighed against the benefit - programmers will still need to write or generate three layers of descriptor files for any pinhead sensor or actuator, regardless of how simple its functional mechanism is.

DDL [12] takes a different approach by basing itself on a service-oriented sensor platform to accommodate devices of all scales (complex devices as well as pinhead sensors and actuators). As the sensor platform handles signal transmissions and networking issues, it frees DDL to concentrate on higher level tasks, such as modeling protocols and data processing. In DeviceKit, writing a DKML file for a device requires both high level knowledge, such as signal format and protocol syntax, and low level information, such as connector model and IP configurations. In cases where system designers, or device manufacturers need to avoid building such low level components, DDL separates the two sides with a clean-cut distribution of responsibilities: system integrators need only to investigate how to connect devices to sensor platforms, while the high level details are already covered in DDL descriptors provided by the device manufacturers.

In addition, DDL achieves a complete self-integration of devices using service discovery utilities provided by the ATLAS middleware. DDL is lightweight compared to DeviceKit, as the sensor platform relieves its burden by taking care of the connection layer. The main major drawback of DDL is perhaps due to its strength, and that is the reliance on a provided sensor platform. While significant advantages are realized by that reliance, they must be weighed against efficiency and cost considerations.

We invite the research community to participate by writing commentaries and suggesting modifications or even alternatives. However, we strongly believe that real-world events are critical to the next generation of healthcare systems and that device integration is a hotspot of research in urgent need of open solutions and standardization efforts.

It is important to mention that neither DDL nor DeviceKit is mutually exclusive. In fact, one of the major value propositions of SODA is that you can easily swap out a different implementation of a device without impacting the application or service interface. For example, an Atlas enabled blood pressure cuff could be swapped with a DeviceKit enabled Bluetooth implementation. Similarly, a future third proposal of a SODA device
integration could be used alongside DDL and the DeviceKit.

B. STEPSTONE Interfaces and Portals

Family members and care providers will often not have the time or expertise to interpret large volumes of events that are generated by individual monitoring devices. Fortunately, authorized individuals, such as Charlie’s son and care provider, can view graphical summaries of the data. One interface we created for STEPSTONE is a mobile client application that provides a quick visual of Charlie’s health (Fig. 10), in addition to drill down functions to know more details about his health status. The interface could be designed as content utilizing mobile browsers, but a full mobile client provides a more controlled experience. Android, iPhone, J2ME all provide equally powerful capabilities to represent the STEPSTONE data.

Combined with analytics [14] and complex event processing subsystems [15], additional realizations of the STEPSTONE data can provide further insights that are important in understanding Charlie’s wellbeing. Such interfaces are better implemented as web based dynamic content. Fig. 11 shows our current web based portals that enable tracking of information, such as Charlie’s daily activities, his blood pressure, and other pertinent information that can be collected at the device layer.

In a National Institute of Health (NIH) funded study, STEPSTONE was used to monitor behaviors leading to obesity in older adults. In this study [16], body weight (true sensor), energy consumption and energy expenditure were monitored to provide assessment and caloric information to the user to aid in self-management of obesity.

As an illustration of the extensibility and openness of STEPSTONE, researchers at Washington State University integrated an Actigraph data device that measures caloric energy expenditure (Fig. 12) into STEPSTONE [17]. The captured information is valuable for patients and caregivers who wish to monitor their exercise levels, as part of a regular health regimen. We wanted to determine how easily the Actigraph device and data collection could be integrated into STEPSTONE and its results displayed via a web portal. We were able to quickly complete the integration and provide visualizations of the data with minimal effort.

As the project expands to include other devices and vital data, such as daily activities and medicine intake, STEPSTONE will facilitate the easy integration of the devices, the data collection, event processing, and the visualization into a comprehensive monitoring package.

VI. RELATED WORK

A. Tele-Health Deployment and Practices

There is a plethora of literature on tele-health projects integrating a variety of medical devices to support applications for health monitoring and disease management. In this section, we select a number of important projects and compare the approaches in the design and development of their systems.

Sanchez et al. [18] describe the implementation of a telemedicine backend system that utilizes medical processes modeled on the HL7 Reference Information Model (RIM). It provides mechanisms for storage and presentation of medical data from remote patients but does not provide any means for deployment and self-integration of personal medical devices into the medical portal backend. This paper also does not provide any insight into ways of presenting the medical data to different types of stakeholders for example, attending physicians, as opposed to family members.

Perez et al. [19] propose a basic framework for integration of sensors as software services in an Ambient Assisted Living
Environment. The authors utilize two main technologies to enable integration namely, Device Profile for Web Services (DPWS) and OSGi. DPWS is utilized to represent low-level devices as web services and has access to discovery, eventing and security mechanisms. OSGi is utilized as a framework to provide an integrated application development environment that promotes development of applications as compositions of available device services. While the basic philosophy and some of the underlying mechanisms (such as OSGi) of this system are the same as STEPSTONE, there are a number of important differences. First, this system requires the resource constrained devices to be represented as web services; something that may not be feasible on low-level primitive sensor platforms that do not have the processing, memory or networking resources to run HTTP. Second, despite the use of OSGi, unlike STEPSTONE, the authors fail to leverage its use in providing a unified software service interface to the myriad heterogeneous types of sensors and actuators out there. Moreover, their system lacks any tools, such as STEPSTONE’s DDL to represent sensor and actuator information and to automatically generate device service drivers without manual programming. All this results in an application development framework that requires the programmer to have knowledge about the low-level aspects of specific sensing and actuation devices that he or she wants to utilize in his/her application.

Sun et al. [20] propose a user-centered hybrid approach called Mutual Assistance Community that attempts to combine care services provided by humans with applications provided by assistive devices. The model proposed by the authors recognizes the importance of utilizing Service Oriented Architecture (SOA) in integrating assistive device services into Ambient Assistive Living environments (AALs). They propose that both human and device functionalities should be published as services in a SOA, the former as web services and the latter as OSGi services. However, their system also has an interesting concept, called the “Participant Model”, which is utilized to encourage group activities amongst the elderly. While the overall concept and philosophy of the authors’ system is very similar to ours, they do not look at the problems facing self-integration of assistive devices in AALs and how one can enable seamless installation and use of new devices in an existing AAL without requiring intervention on the part of the resident or any disruption in their routine due to presence of external service personnel. Moreover, the interaction process described in the paper still seems to require extensive manual involvement of the user and is perhaps infeasible when the target audience is elderly.

Martinez et al. [21] describe a system that utilizes ISO/IEEE11073 and EN13606 standards to achieve end-to-end connectivity between personal health monitoring devices and healthcare backend portals. While the goals of this system are similar to our goals, it does not take into consideration mechanisms that enable medical devices to be automatically self-integrated into existing health portals (as opposed to integration through manually programming). For example, there is no discussion on how interfacing software, such as drivers, can be automatically generated on-demand for different medical devices without expert human intervention. Furthermore, they do not provide any middleware mechanisms for plug-and-play deployment of medical devices that become immediately available to application developers as programmable services.

Raad and Yang [22] describe a smart home system that utilizes multiple sensors to monitor the condition of the resident, typically an elderly person, and sends emergency calls to first responders in the case of an accident or impending health problem. A web interface is also provided for remote management of this system, including specific access control for different users. The main issue with this system is that even though it provides a working configuration of sensors, communication channels and servers that can be deployed in a smart home, it provides no mechanisms that would ensure that such a configuration can be easily modified or re-programmed in the future without running into integration issues at both the hardware and software levels. Furthermore, no capabilities are provided for automatic self-integration of any sensor in the smart space. Hence, any addition of a new sensor will require significant re-programming and testing of the entire system to ensure that nothing is broken.

B. Standardization

In addition to research efforts in academia, the industry is showing a growing interest in tele-health and is becoming a driving force in the standardization of tele-health products and services. Most of these standards address at least one of the following aspects of tele-health: 1) network and communication, 2) device interface and description, and 3) medical information and healthcare records.

1) Network and Communication: Tele-health is the use of electronic information and communication technology to deliver health and medical information and services over large and small distances [23]. The network of sensors, medical monitors and handheld devices, along with a variety of wireless coverage available today, enables the convenient and pervasive delivery of tele-health services. Several network standards are in use or are being considered for use in wireless healthcare monitoring. Bluetooth is one of the most widely used wireless standards for home medical devices [24]. Compared to IEEE 802.11 standards, Bluetooth provides a lower bandwidth with much lower power consumptions, more suited to medical devices and on-body sensors operating by battery. Bluetooth has become the basis of IEEE 802.15, a new family of standards that specializes in Wireless Personal Area Network (PAN). IEEE 802.15.4 defines physical layer and data link layer standards that address communication with low data rate but very long battery life and very low complexity [25, 26]. ZigBee, running over IEEE 802.15.4, is a popular industry standard for Wireless home area networks (HANs), which provides a low-cost and low-power solution to connect medical devices.

Wireless Medical Telemetry Service (WMTS) is a standard for wireless data transmission related to patients’ health (medical telemetry) [27-29]. It designates the bands 608-614, 1395-1400 and 1427-1432 MHz to avoid interference with other consumer electronic devices, such as televisions. WMTS is established by the Federal Communications Commission (FCC) and therefore, the assignment of its frequencies has not been agreed internationally. Another limitation of WMTS is that FCC does
not allow the home use of WMTS equipment. The medical implant communication service (MICS) [30, 31] is a standard widely used for implanted wireless transmitters. It uses the 402-405 MHz band.

The heterogeneity of networks imposes a great challenge to the integration of multiple tele-health devices communicating using different network protocols. In STEPSTONE, this challenge is addressed by adopting Atlas, a sensor platform with interchangeable interface boards and network modules. An Atlas node serves as a gateway between the Internet and the local network of medical devices. By changing the interface board, the node can connect to devices using a variety of communication protocols (such as Bluetooth, ZigBee, USB, IR, and RS-232).

2) Device Interface and Description: Two classes of standards are in use for describing medical devices and interfaces: 1) dedicated medical device standards for tele-health; 2) general sensor/device standards. The Continua health alliance standard [4] is an example of the first class. It is a set of specifications on two interfaces: PAN-IF - interface to Personal Area Network health devices, and xHRN-IF - interface between Disease Management Services (DMS) WAN devices (xHR Senders) and Electronic Health Record (EHR) devices (xHR Receivers). The specifications ensure interoperability of devices, and were specifically written for device manufacturers that intend to undergo the Continua Certification process with their devices, companies that integrate (integrators) Continua devices in systems and subsystems, and test labs that certify (certifiers) compliance to Continua specifications [32, 33]. The ISO/IEEE 11073 PHD (Personal Health Data) standard family is another prominent example of dedicated standards for health devices. It includes a group of standards addressing the interoperability of a variety of personal health devices, such as weighing scales (IEEE 11073-10415), blood pressure monitors (IEEE 11073-10407), and blood glucose monitors (IEEE 11073-10417). On top of these device specialization standards, IEEE 11073-20601 provides a framework standard that defines the generic data types, message types and communication model shared by these devices.

The other class of standards is designed for general devices and sensors, and could be applied in tele-health systems to support personal health devices. These standards include Energy Conservation and Homecare Network (ECHONET) [34], IEEE 1451 [33], and SensorML [35]. The ECHONET standards specify an open system architecture that enables the integration of a variety of home appliances and sensors from multiple vendors. ECHONET provides a device specification that explicitly defines their properties and access methods to describe the interface of these devices. Therefore, vendors must create the same interface as specified in the standard to earn an ECHONET certification to get a device approved and incorporated into the home network. ECHONET originated in Japan, and ECHONET devices are available primarily for the Japanese market. The IEEE standard for smart transducer interfaces (IEEE 1451) describes a set of open and network-neutral interfaces to connect sensors and actuators to communication networks and processors. The standard uses Transducer Electronic Data Sheet (TEDS) to specify information, such as transducer identification, calibration, correction data, and measurement range. TEDS typically resides in the embedded memory of a transducer, such as an electrically erasable programmable read-only memory (EEPROM). IEEE 1451’s objective is to create a standardized interface that sensor or actuator manufacturers can follow to create cost-effective sensors and actuators that interface with a variety of networks, systems, and instruments. The Sensor Model Language (SensorML) is a product of the sensor web enablement activity, a research initiative in the geospatial community that mostly uses remote sensors and sensor systems, such as satellites and radar stations. However, since it provides a generic model for sensor data modeling and processing, it also has the potential to be used for tele-health devices. SensorML could be seen as the sensor standard most useful for sensor data interpretation and preprocessing to bridge the gap between low-level, hard-to-use data and higher abstractions.

Many aforementioned standards (most of the dedicated standards and the ECHONET standard) stress restricting device manufacturers, by defining a fixed set of devices and their attributes. As new devices and new features emerge, integration of these devices becomes difficult without constantly updating the standards. In addition, the strict regulations on device specifications could discourage manufacturers from complying with the standards. In contrast, STEPSTONE adopts the open SODA standard and imposes minimal restrictions on devices. Instead, it encourages manufacturers to participate by specifying their products using descriptive languages, such as DKML and DDL, which are capable of describing a richer variety of devices.

In addition, none of the surveyed standards has achieved a complete solution to enable automatic end-to-end integration that incorporates devices of all scales. Some standards require partial manual efforts to write driver software that connect applications to the devices; and most standards lack verification tools to check if the hardware and software components comply with the standard. In contrast, STEPSTONE promises a seamless integration process that is completely automatic. Both DeviceKit and DDL provide software tools that automatically verify and convert device descriptor files into software service bundles. While DeviceKit employs a three-layered model to provide a complete abstraction of the communication between devices and applications, DDL leverages the Atlas Platform’s capability to handle low-lever tasks and focus on the message layer. Both approaches achieve true end-to-end connections.

3) Medical Information and Healthcare Records: It is desirable to have a standard way of expressing, exchanging, and storing medical information and healthcare records to ensure the interoperability among medical devices and tele-health applications. This typically requires a standardized language that defines a common nomenclature, data types, message syntax, and encoding rules. Most standards can be classified into two categories: EHR and PHR.

EHR provides structured content to store and exchange clinical information. The most important EHR standards include CEN prEN 13606 EHRcom [36], HL7 CDA [37], and DICOM SR [38]. A study [39] shows all these standards use very similar concepts and models, and promise similar capabilities. The major difference among them is the progress achieved in terms of the standardization process.

PHRs are individually maintained records that provide a
complete and accurate summary of the health and medical history of a person. An individual patient has complete control in accessing, managing and sharing the information stored in his/her PHR. PHRs differ from EHRs controlled by health care providers [40]. The most successful PHR implementations include Google Health [41] and Microsoft HealthVault [42]. Both PHR platforms allow users to store and manage health information in one central place securely, privately, and free. Despite the differences in end-user features, study [43] shows that no definite one central place provides richer functionalities.

The STEPSTONE portal is also a PHR platform. In addition to the common PHR services, such as retrieving and managing health records, the STEPSTONE portal provides a variety user interfaces using voice, text message and graphical visualization to deliver tailored information to various mobile clients and PCs. As an ongoing project, we are developing the next version of the STEPSTONE portal (known as STEPSTONE 2.0) to exploit social networks and robots as enhanced interfaces to manage and share healthcare information [44].

VII. CONCLUSION AND CALL FOR PARTICIPATION

We presented the STEPSTONE project whose objective is to develop a community based standard framework for device and sensor integration. The SODA specifications let programmers deal with devices, such as sensors and actuators, as business services in today’s enterprise Service Oriented Architectures. SODA converts hardware devices to software services with well-defined interfaces, independent of the programming language and the computing platforms to which they are connected.

SODA is intended to close the gap between system integrators and device and sensor manufacturers. It accomplishes the “Integrate once, deploy everywhere,” vision and lets developers focus on holistic solutions, instead of low-level device adaptations.

SODA provides a significant business opportunity for real-world health and wellness services to be developed and launched with significant reduction in cost and risk in the connected/wireless health domain. Researchers and developers working in wireless health are highly encouraged to use STEPSTONE as a base system that can be further specialized to cater for specific needs.

ACKNOWLEDGMENT

This research is partially supported by an IBM Emerging Standards Division and an NIH Grant number 1R21DA024294.

REFERENCES

20. H. Sun, V. De Florio, N. Gui, and C. Blondia, “The missing ones: key ingredients towards effective ambient assisted living


Sumi Helal

Sumi Helal is a professor in the Department of Computer and Information Science and Engineering at the University of Florida. He directs the Mobile and pervasive computing laboratory. His research interests include pervasive computing, mobile computing, and their applications to health telematics. Helal received a PhD in computer sciences from Purdue University. Contact him at sumi.helal@gmail.com
STEPSTONE: An Intelligent Integration Architecture for Personal Tele-Health

Raja Bose

Raja Bose is a senior researcher at Nokia Research Center Palo Alto. His research interests focus on the theme of mobile interoperability, including creating new devices, systems, and services that provide innovative user experiences by leveraging interoperability of mobile devices with other smart environments. Bose received a PhD in computer engineering from the University of Florida. Contact him at raja.bose@nokia.com.

Chao Chen

Chao Chen is a PhD candidate in the University of Florida's computer information science and engineering department. His research focus is on programming models and safety engineering for pervasive systems. Contact him at cchen@cise.ufl.edu.

Andy Smith

Andy Smith is a software engineer at IBM focusing on emerging standards and technologies to build highly interactive and collaborative applications. His expertise bridges a number of domains including application development, integration and deployments across vertical market solutions. Contact him at andybs@us.ibm.com

Scott de Deugd

Scott de Deugd is a senior engineer with IBM's Software Standards group. His current work is focused on standards and open source initiatives for the integration of sensors, actuators and other devices into enterprise systems. Research interests include healthcare standards, Web 2.0 application models, embedded systems, object-oriented languages and tools, partner and university relationships, and embedded Java. Contact him at dedeugd@us.ibm.com

Diane Cook

Diane Cook is a Huie-Rogers Chair Professor at Washington State University. Her research interests include machine learning, data mining, graph algorithms, and smart environments. Contact her at cook@eecs.wsu.edu.