Software platform design for personal service robots in healthcare

Chandan Datta [1] and Hong Yul Yang[1] and I-Han Kuo[1] and Elizabeth Broadbent[1] and Bruce A MacDonald[1]

Abstract— **This paper describes the software platform design of a personal service robot with application to healthcare scenarios. The platform was designed after exploration of both the service application and software architectural design spaces. The service application considered here is a comprehensive medication management service which evolved to be one of the more complex use cases in the Healthbots project. While algorithms and user studies of human robot interaction are reported in the literature, there is less attention given to related software frameworks and tools, which are addressed in this paper. The paper focuses and solves numerous design challenges to achieve the desired functionality of the medication management service with its complex set of workflow and contextual requirements which were not present in the software platforms we had designed. The robots were used in a real world deployment scenario at an Aged Care Facility. Scenarios and results from a field trial are presented to enable the research community to understand the technological and engineering challenges of deployment of the robot system and the level of fluid integration required to achieve the desired goals of functionality and robustness in the real world.**

I. Introduction and background

The service robotics industry around the world will play a key role in economic transformation of several knowledge-based economies which depend on innovation for growth. This makes it important to understand the tools which will be useful to bring about such benefits. Robotic software platform design is considered a consolidated and necessary discipline centered on the idea of reducing complexity in software development and evolution through abstraction and separation of concerns [1]. It is important to design software architectures that exhibit a good tradeoff between functional requirements and multiple quality attributes, and a key requirement is effective interaction between humans and service robots.

The current research in human-robot interaction can be classified into three major themes. The first theme relates to studies and field trials understanding the psychology of human-robot interaction, the second theme relates to algorithms for affective computing for driving the human-robot interaction such as gaze tracking or face tracking while the third theme concerns software engineering and programming language technologies for implementing the human-robot interaction design. While there is a body of literature dedicated to the first two themes, description of software architectures and platform challenges are not often published. This aspect is the theme of this paper.

[1]C. Datta, H.Y. Yang, I.H. Kuo, E. Broadbent and B.A. MacDonald are with the University of Auckland, New Zealand chandan.datta at auckland.ac.nz

Glas [2] presents a software architecture with an easy-to-use graphical interaction development environment called Interaction Composer for designing social robots by cross-disciplinary teams of programmers and interaction designers. The OPRoS (Open Platform for Robotic Service) software platform for network based intelligent robots aims to provide a comprehensive development environment with extensive functions to accommodate the increasing demands for developing a variety robot software applications [3]. Jung [4] provides a description of the service robot development approach used to develop core robot services used on Yujin robots. Touretzky [5] describes the Tekkotsu application development framework for the Sony AIBO robot to provide an intuitive set of primitives for perception, manipulation, and control. The Care-O-Bot 3 robot, developed by the Fraunhofer IPA has a hybrid control architecture for task planning and execution [6], and has demonstrated capabilities to perform mobile manipulation tasks using the Robot Operating System (ROS) [7].

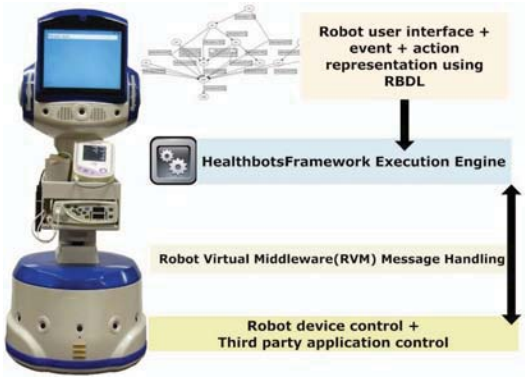

Fig. 1. *Healthbots Framework* platform used on Charlie

In comparison to the current state-of-the-art, our research group has been working on improving software design for human-robot interaction. Compared to the current research on programming languages and software platforms which are used on completely autonomous systems capable of intelligent behavior based on sensing and planning, Healthbot service design instead requires an additional layer which can use pre-built algorithms and use them for the desired human-robot interaction required in those services. The *Healthbots Framework* platform [8] raised level of abstraction and introduced an easy-to-use visual programming environment [8] for designing service applications by multi-disciplinary teams of programmers and Subject Matter Experts (SMEs). Fig. 1 highlights essentials of the *Healthbots*

978-1-4799-1201-8/13/$31.00

Framework architecture. We used Charlie for version one of a medication management [9] application. The robot's multi-modal dialog and behavior is specified in a set of states that are organized in a domain-specific language called Robot Behavior Description Language (RBDL) and represented in an XML form. The execution engine navigates through the states and the *Healthbots Framework* enables end-users to customize the robot behavior without requiring changes to the *Healthbots Framework* code itself. The complete design and architecture of the *Healthbots Framework* are presented elsewhere [8].

The *iFlex* platform, which is the theme of this paper and used on the iRobiQ-S robot evolved from what we had learnt from the *Healthbots Framework*. The challenges which led us to develop the *iFlex* platform are presented in section III. iRobiQ-S was chosen for the development of a more comprehensive medication management service. The aim of the medication management program is to safely improve medication adherence, helping older people to manage their complex medication regime at home. This healthcare aim informs the robotics research question: "*What are the precise functional and non-functional requirements for such a robot, and what development methods and software/hardware architecture can fulfil them?*"

The rest of the paper is structured as follows. Section II introduces the Healthbots project. Section III discusses the requirements and challenges for the iRobiQ-S platform. Section IV provides the details of the software platform design on the iRobiQ-S. Section V provides the field experiments and presents the results. Section VI concludes the paper.

II. Healthbot robots

Fig. 2. Charlie (left) iRobiQ-S (right) interacting with older participants

The robots used in the Healthbot project are a joint development of the University of Auckland in New Zealand, with ETRI and Yujin Robot Co. Ltd., in South Korea. The hardware was provided by Yujin Robot while the software and field trial were designed and developed by the University of Auckland. Robots interacting with older participants are shown in Fig. 2. Charlie is a differential drive mobile robot of 1.2-meter height. It has a rotatable touch screen, microphones, ultrasonic sensors, bumper sensors, and a laser range finder. iRobiQ-S is smaller in size measuring 45x32x32cm and weighing 7kg. It is capable of self navigation to any number of landmark targets and can intelligently avoid obstacles in its path. It has an Intel Atom processor based internal computer for running its software. Physically, the robot has 2 arms which are used mainly for getting attention, indicating emotions and gesturing. It is also equipped with a number of touch sensors at different locations on its body. This enables the programming of realistic responses when users pat, tap, touch, or nudge the robot. It has a touch-screen LCD display on its body.

III. Challenges for iRobiQ-S design

A. Taking advantage of the iRobiQ-S platform

The interaction design for the robot is beyond pure user interface design. It is important to take advantage of the robot's embodiment and its human robot interaction capabilities, especially its perception for situational awareness and its motion control for engagement during interactions. Can we design interactions and present the patient with the correct medication for the time and assist him/her through the intake? Thus, enabling the patient and provider working together to achieve a desired therapeutic outcome according to the required medication workflow in the Aged Care Facility including closed loop feedback enabling clinical supervision of patients [10]. There are a number of reasons why the interaction design of healthcare services on the robot is significantly more complex than a simple service application, specially when designing the medication management service. Some of the differences are due to the following requirements:

- The robot should be distinguish between organized (in a pillbox), loose and as required (PRN) medications.
- The robot's dialog should have provisions for exception handling (wrong/missing medications, trouble with instructions).
- The robot should ensure patient safety at all times. The robot must not at any time put the user at risk. If something seems to go wrong, someone must be informed about it, as soon as possible.
- The human-robot interaction should be engaging enough to avoid fatigue of performing mundane tasks such as medication reminders.

B. Multi-disciplinary aspect of the Healthbots project

A bulk of the design challenges are sourced from the "multi-disciplinary" nature of the Healthbots project. The iRobiQ-S design involved significant contributions from individuals having a background in Health Informatics, medicine and public health who wanted to evaluate a futuristic technology in terms of their domain goals. The robot is treated as a 'black box' which should work robustly in order to evaluate it in a scientific study.

One of the biggest challenges is communication within a multidisciplinary team. SMEs in the team aid in generating most of the software requirements. Such requirements documents are written in a narrative style either in descriptive documents or in presentation software. While software designers can communicate specific requirements with UML and its extensions, requirements documents from SMEs have missing information. Either functional information may be missing in informal design artifacts or the structured design artifacts may not always contain all non-functional requirements [11].

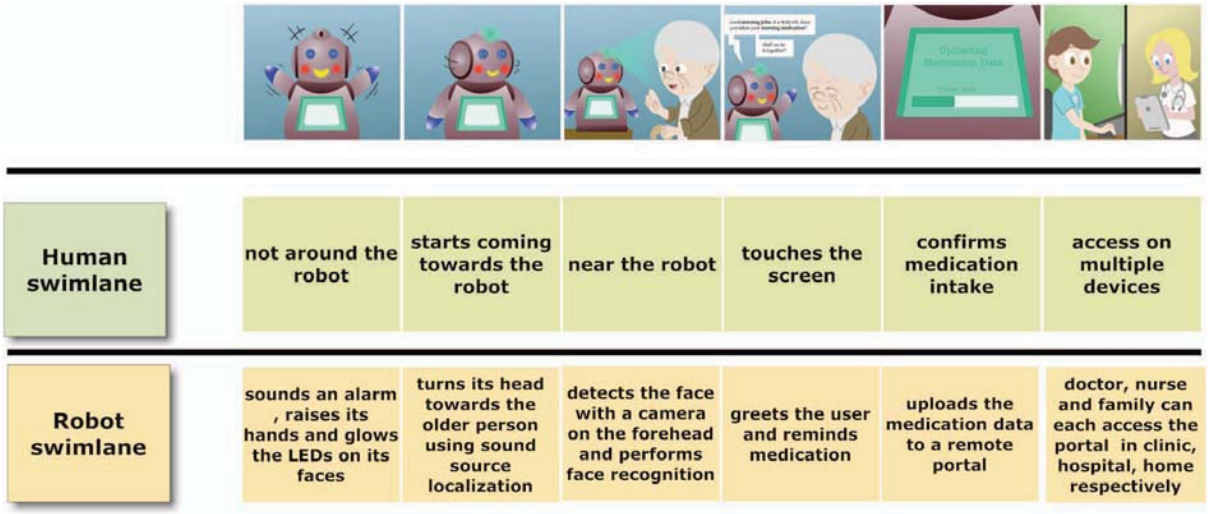

Fig. 3. Human robot interaction design swimlane storyboard used to clearly communicate design goals for the software platform

C. Full autonomous operation over a period of time

The trial where medication management was used on Charlie [9] was only focused on a single, user-initiated interaction. There was no element of autonomous scheduling and operation, and no closed-loop medication workflow provisions, whereas iRobiQ-S required all of these features. In contrast to Charlie, the iRobiQ-S robot had a closed loop medication workflow [10] enabled, which required a richer and more expressive software framework to be designed with constraints which required full autonomous operation without researcher intervention.

D. Limited scalability with respect to robot behavior definition

The behavior representation used on Charlie enables flexibility of quick changes to the robot behavior, but is a flat finite state machine (FSM). Due to the inherent flatness and sequential nature, the FSM representation inflicts repetition. What this means is, many events have to be handled identically and repeatedly in several states. The FSM model doesn't have a mechanism to factor out the common behavior which needs to be shared across states. This situation leads to ***state explosion***. For example, we faced the problem while trying to model the measurement of blood pressure using an advanced commercial medical device (PulseCor R6.5, New Zealand [12]) with the robot. The application logic is sequential with respect to delivering the instructions to setup the cuff, processing the measurement and finally displaying it. However the blood pressure device has a set of output events which need to be handled by the robot in order to display the output measurement or any error caused while taking the measurement. In the RBDL description, we have to define states which can sequentially process the events related to the measurement process or the error events. So, in each stage of the measurement the RBDL definition has to contain the definition of the events (and actions) related to the process of measurement (first stage involves measuring the systolic and subsequently the diastolic measurement to compute feature points [12]) and also definition of the error events which might happen in those stages.

Similarly, for the medication management service, de-serializing complex medication data returned as objects from web-services requires several states. Errors in the xml are common when it become large and having to fix them spontaneously during the field trial detracts user experience during the trial.

IV. The *iFlex* platform

The *iFlex* software platform was used to implement the iRobiQ-S robot's scenarios taking into account the various design requirements and the challenges discussed in the last section. The human robot interaction [13] was crystallized using a storyboard. A part of the storyboard used to visualize the iRobiQ-S scenario is shown in Fig. 3. We proposed the design of storyboards to design the scenarios of the robot's usage beyond graphical user interface designs to overcome the issues of communication and comprehension mentioned in section III-B. The storyboard is in form of a swimlane to describe what actions the user and the robot will take under the context of medication reminders. Scenario description swimlanes visualize the activities of multiple actors (the human and the robot) in a flow of events. This addresses the requirements related to multi-disciplinary design challenges mentioned in section III-B.

A major part of the *iFlex* software platform is written in Adobe's Actionscript 3 (AS3) object oriented scripting language and extends several capabilities of Adobe's flex framework. The *Healthbots Framework* engine is also written in AS3, however, there are clear differences between the *Healthbots Framework* and *iFlex* designs, specially for representing the robot's behavior. In the *iFlex* platform, Adobe MXML [14] instead of RBDL is used to represent view states which are conceptually different from robot behavior states in RBDL. Developed by Adobe, MXML is

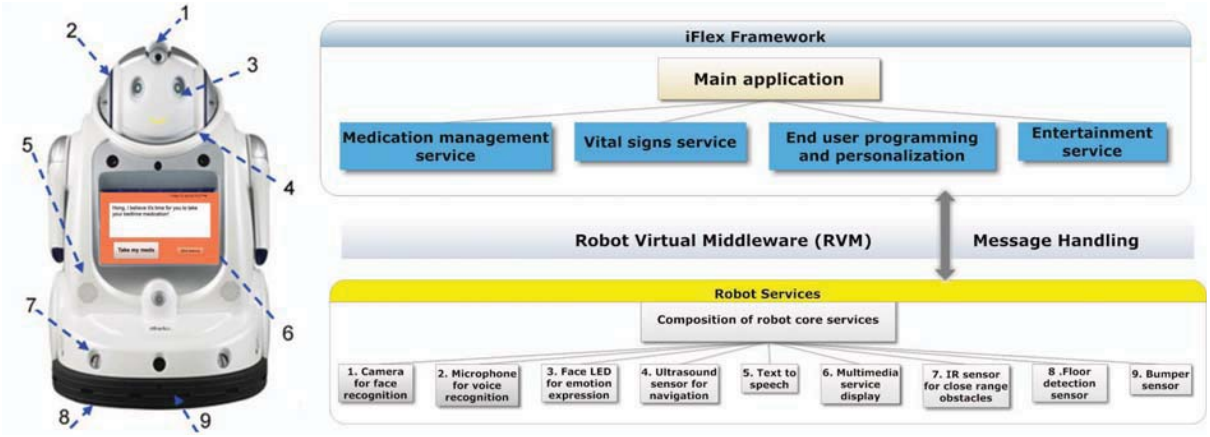

Fig. 4. Interaction of the *iFlex* platform with robot's services using the middleware

a domain-specific language and obeys the xml validation rules. Adobe briefly defines it as "an XML language that you use to lay out user-interface components for Adobe Flex applications" [15]. However, XML tags in MXML can be used to declare things such as the containers, controls, effects, formatters, validators, and web services in addition to declaratively laying out the user interface [14]. Hence, MXML is not appropriate for describing what the robot does when a user clicks a button on the screen, for example. For that, AS3, a procedural language, offers executable methods, various types of storage variables, and flow control such as conditionals and loops.

AS3 is an object-oriented procedural programming language, based on the ECMAScript (ECMA-262) edition 4 draft language specification. MXML as a convenience language gets compiled into AS3 classes in the background during the compilation process before being compiled to swf byte-code [14]. This gives Adobe Flex frameworks an unique architecture to incorporate these two programming languages which are compiled into a combination of files that work together [16]. The following sub-sections describe the features of the *iFlex* platform used to tackle the many challenges described in section III.

A. Service-to-robot interface

As shown in Fig. 4, the *iFlex* platform is used to develop the healthcare service behaviors. There is a main application, a Flash swf which loads other services such as blood pressure measurement as Flex modules [17]. The design of the *iFlex* platform allows developers to extend the Flex framework by adding new classes derived from Flex, as well as expand the framework by adding new classes that are not derived directly from Flex. The *iFlex* platform has clearly defined interfaces to the robot's core services which often are written in C/C++ and run natively. A component called the Robot Virtual Middleware (RVM) [4] allows the marshaling and demarshaling of messages in order for AS3 and C++ code to inter-operate. The combination of Flash Player and the *iFlex* design exploits the maturity of Flex framework for modular component based service development.

B. Asynchronous event handling and dynamic view state generation

The *Healthbots Framework* execution engine essentially processes events and performs the robot's actions based on the behavior definition in the RBDL code. Due to the data complexity in the medication management application as will be described in sub-section IV-C as well as the event handling complexity, MXML view state based definitions were chosen for the *iFlex* platform. Conceptually, MXML enables the abstraction of the multi-modal view definition on the robot from the application logic through dynamic view state generation by the core classes in the *iFlex* platform. Events let the robot know when something has changed within a service. They can be generated by the attached medical devices, such as the blood pressure or blood oxygen devices, or other external input, such as the return of a web service call. Any user interaction with the robot's service can generate events. Events can also occur without any direct user interaction, such as when data finishes loading from a remote server or when an attached camera detects a face. The robot's service modules can handle these events by using event handlers or event listeners. When an event is handled, the platform might generate a view dynamically based on the rules of the handled event.

This powerful feature lets the interface change based on the task the user is performing without explicitly defining the view, but defining the components of the view based on an event rule. Since much of the robot's events are asynchronous in nature, it is easier to define which components appear in a view based on an event than to define multiple views in sequence. Thus, the language features used in the building of the *iFlex* platform modules, lets us avoid the problems of state explosion mentioned earlier in section III-D and made it easier to construct complex branching application logic present in the medication management service. This was a reason for fewer errors and faults with the application logic in the *iFlex* platform compared to the *Healthbots Framework* reported in earlier field trials [18].

C. Complex data binding

Fig. 5. Data binding in Adobe Flash Builder

In the medication management service, the medication information is to be provided correctly as organized, loose and PRN (as required) medications. In this case, it would be ideal to define a single view state which can handle the data un-marshaling from a remoting web-service and let the service application logic bind a place-holder for a complex data object. Data binding is the process of tying the data in one object to another object. It provides a convenient way to pass data between the different layers of the application. We use data binding to tie properties of user interface controls to a middle-tier data model, and to bind that data model's fields bound to a data service request from a web-service. MXML provides language features such as meta-tags and a special curly braces syntax to enable this and is easy to author in Adobe's Flash Builder programming environment as shown in Fig . 5. Since the complex branching of dialogs of the medication management service required generation of states dependent on the data layer, the problem was it led to many states which almost had the same dialog, but only differed in the data presentation. RBDL used in the *Healthbots Framework* did not deal with the problem as efficiently as the language features used in the *iFlex* platform. This way it avoids the problems of scalability mentioned earlier in section III-D.

V. Field Trials and Results

During a long term field trial conducted at an Aged Care Facility for 4 months in 2011-2012, a total of 6 iRobiQ-S were deployed as personal robots to independent residents in the village. In this paper, we report the results from the iRobiQ-S platform. Only iRobiQ-S robots were used for medication management, Charlie robots were used for other functions.

When the trial started, each participant was given an introduction to the robot and was also taught how to operate the robots including how to use a touch-screen, adjust audio volume and turn the robot on/off. Other than this short introduction, no supervision or technical support was given on the sites during the whole trial. Whenever participants needed help or assistance, there was a 'Help' button on the robot and a duty phone number which they could call. When the 'Help' button was pressed a SMS text message was sent to the support team. If required, an assistant attended the robot to fix the problem. All the issues reported by the participants were recorded and they demonstrate how robust the robots were and what were the main issues that the participants experienced.

Issues	**Number times reported**
3G / Wireless network issues	23
Application froze	21
Blood pressure device related errors	7
SSD related issues	2
Battery and charging related issues	2
Robot not reminding medication	2
Robot missed medication	2
Robot reminded medication incorrectly at 4am	1
Change of medication not updated on the robot	1
Total:	61

TABLE I

System issues reported during the field trials

A sub-classification of issues based on their sources would be useful for the reader to understand the nature of the errors reported. The errors can be classified as follows:

- availability of the network
- efficient integration of the medical devices with the robot
- design of the robots hardware
- underlying software platform from the robot manufacturer and Adobe

The issues reported by the participants are shown in Table I and explained as follows. To provide medication reminders accurately, the iRobiQ-S robot always synchronises with a portal 'RoboGen' [10] before issuing a reminder. Whenever the iRobiQ-S robots lose access to the Internet through either 3G or wireless network, they stop reminding to prevent outdated medication information from being given to their users. This was part of the requirements for the field trial ethics application. However, this preventive measure caused several issues to the usability of the robots since the Aged Care Facility sometimes experiences wireless network outage. This was the most reported issue by the participants. The second major source of issues is related to the stability of the robot's software. During the trial, the robots often operated for multiple days without stop and there was a few times when the software stopped responding after continual use of a considerable amount of time.

The rest of the issues reported are mostly related to the design of the robots' hardware and software. For example, iRobiQ-S robots automatically turn their LEDs eyes and touch-screens off to save battery power after a certain period of inactive time, but the control of this feature was not available to the medication reminder application in the iFlex framework. This sometimes created issue when iRobiQ-S robot needed to issue a new reminder after a long period

of inactive time; iRobiQ-S robot would move its arms and speak out to attract its user's attention, but with its eyes and touch-screen off. For participants who were not familiar with technology, this sometimes caused confusion and was therefore reported.

Overall, the robots operated robustly for the period of time deployed and most of the issues reported could be fixed by research assistants by restarting the robot and showing the residents how to do this themselves. Other fixes included the following: reconnecting loose cable leads, repairing folders, removal of a non-functional robot, reinserting loose USB cables, reconnecting wireless network, moving the robots to the windows for stronger wireless network connections and providing a smaller blood pressure cuff.

These results reflect the errors or faults reported as well as the usability issues caused to the participants due to them. While *iFlex* platform met all the requirements for the robot's services, it is only the software platform designed for service application development. The criticality of some of the issues also show the need for a holistic and well integrated hardware-software development approach where the system is developed upfront with the target human-robot interaction scenario in mind, especially the constraints posed by the service complexity, environmental context and the target user group.

VI. Conclusions

In conclusion, our team put a special emphasis on the technical aspects of robot software platform development for the specific application domain of healthcare and achieved all the goals required of the software platform. We learned that stringent non-functional requirements made it necessary to work in fault-tolerance mechanisms directly into the dialog of the robot for the *iFlex* platform. The overall development process highlighted some key learnings. To the SMEs and domain experts, the functionality progression was "linear", i.e. medication 'X' times a day to a continuous mode throughout the day. But, in technology and engineering terms, the progression was comparatively a big leap – user-initiated control to autonomous control and fault-tolerance. The lessons learned by developing both the *Healthbots Framework* and *iFlex* platforms and their testing through real world field trials will inform future designs.

Acknowledgment

This work was jointly supported by the Robot Pilot Project program of the Korea Ministry of Knowledge and Economy (MKE), Korea Institute for Robot Industry Advancement (KIRIA) and the New Zealand Ministry of Business, Innovation and Employment IIOF (13635). We thank Electronics and Telecommunications Research Institute (ETRI) for their valuable contributions and help with the research. We would also like to thank Yujin Robot for their technical support and our colleagues of University of Auckland HealthBots research team for their ongoing support.

References

[1] D. Brugali and P. Scandurra, "Component-based robotic engineering (part i)[tutorial]," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 4, pp. 84–96, 2009.

[2] D. Glas, S. Satake, T. Kanda, and N. Hagita, "An interaction design framework for social robots," *Proceedings of robotics: science and systems, Los Angeles, CA, USA*, 2011.

[3] S. Han, M. Kim, and H. S. Park, "Open software platform for robotic services," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 3, pp. 467 –481, july 2012.

[4] H. Chul, J. Ryul, and K. Stonier, "The study of robot platform for orchestrating and reusing services," in *Advanced Robotics and its Social Impacts (ARSO), 2010 IEEE Workshop on*. IEEE, 2010, pp. 162–164.

[5] E. Tira-Thompson and D. Touretzky, "The tekkotsu robotics development environment," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 6084–6089.

[6] U. Reiser, C. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hagele, and A. Verl, "Care-o-bot 3-creating a product vision for service robot applications by integrating design and technology," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1992–1998.

[7] F. Weisshardt, U. Reiser, C. Parlitz, and A. Verl, "Making high-tech service robot platforms available," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*. VDE, 2010, pp. 1–6.

[8] C. Datta, C. Jayawardena, I. Kuo, and B. MacDonald, "Robostudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012. Vilamoura, Algarve, Portugal*. IEEE, 2012.

[9] P. Tiwari, J. Warren, K. Day, and C. Datta, "Comprehensive support for self management of medications by a networked robot for the elderly," in *Health Care and Informatics Review Online, www.hinz.org*, ser. HINZ '11, 2011.

[10] C. Datta, P. Tiwari, I. Kuo, and B. MacDonald, "End user programming to enable closed-loop medication management using a healthcare robot," in *Proceedings of the 2011 Australasian Conference on Robotics & Automation, Melbourne, Australia*, 2011.

[11] M. Haesen, J. Meskens, K. Luyten, and K. Coninx, "Supporting multidisciplinary teams and early design stages using storyboards," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part I: New Trends*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 616–623.

[12] A. Lowe, W. Harrison, E. El-Aklouk, P. Ruygrok, and A. Al-Jumaily, "Non-invasive model-based estimation of aortic pulse pressure using suprasystolic brachial pressure waveforms," *Journal of Biomechanics*, vol. 42, no. 13, pp. 2111–2115, 2009.

[13] C. Datta, P. Tiwari, H. Y. Yang, E. Broadbent, and B. MacDonald, "Utilizing a closed loop medication management workflow through an engaging interactive robot for older people," in *e-Health Networking, Applications and Services (Healthcom), 2012 IEEE 14th International Conference on*. IEEE, 2012, pp. 313–316.

[14] Adobe, "Using ActionScript," http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf61c8a-7fff.html, 2012, [Online; accessed Aug 2012].

[15] ——, "MXML syntax," http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf69084-8000.html, 2012, [Online; accessed Aug 2012].

[16] ——, "Compiling MXML to SWF Files," http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf5f39f-7fff.html, 2012, [Online; accessed Aug 2012].

[17] ——, "Flex Modular applications overview," http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf69084-799a.html, 2012, [Online; accessed Aug 2012].

[18] P. Tiwari, J. Warren, K. Day, B. MacDonald, C. Jayawardena, I. H. Kuo, A. Igic, and C. Datta, "Feasibility study of a robotic medication assistant for the elderly," in *Proceedings of the Twelfth Australasian User Interface Conference-Volume 117*. Australian Computer Society, Inc., 2011, pp. 57–66.