Software Interface Design for Home-Based Assistive Multi-Robot System

Patrick Benavidez, Mohan Kumar, Berat Erol, Mo Jamshidi, Ph.D and Sos Agaian, Ph.D
Department of Electrical and Computer Engineering
The University of Texas at San Antonio
San Antonio, TX, USA
patrick.benavidez@utsa.edu, mkumar2301@gmail.com, berat.erol@utsa.edu, moj@wacong.org, sos.agaian@utsa.edu

Abstract - *In many assistive robotic systems, the interface to the user is simply a tablet computer or a monitor attached to a single robot. Missing from approaches are the system extensibility made possible with a tablet computer and a division of work between multiple agents. In this paper we present the design for a software interface to connect users to an assistive robot system for the disabled and elderly. The system is comprised of heterogeneous low-cost assistive robots, a home management portal and a cloud computing backend. The system is designed with the premise that all components do not need to be present for the system to function, but it will be improved when expanded by addition of robots and expanded computing capabilities. This paper focuses on developing the interfaces necessary to connect the user to these systems in a simple and easy to comprehend manner for the target user population.*

Keywords: Robotics, Robot Operating System, assistive robot, cloud computing, user interface design.

1 Introduction

The authors are working on an assistive robotic system for assisting the elderly and disabled. In the assistive system, there exist multiple robots to aid a user, a home management portal for the robots, and a cloud computing backend. To facilitate design of a future- proof system, there is a need to make the system modular and extensible. A user, or their care physician, can pick and choose the components that will best benefit the user. The system can then be expanded at any time, as the user can actually afford, or now sees the need for the add-on.

With this in mind, the software interface has to work in different modes, i.e. direct to one or more robots, to the home management portal. The robots also have to be able to work in multiple modes, utilizing any local and/or global information available to them. Initial designs of graphical interfaces for the system applications focused too heavily on direct control of single agents in the system. Provided the fundamentals of systems of systems (SoS), we decided to provide a wholistic approach to controlling the system of robots. We soon sought out usability design software engineering tools to improve the designs.

The rest of the paper is structured as follows: Section 2 provides a short background on the assistive robot system. Section 3 describes the system interface design, and Section 4 details the user interface design.

2 System Interface Design

2.1 System Communication

To facilitate multiple connection paradigms, we utilize ROS as not only the middleware for the robots, but also as the communication standard for which applications and robots interact. ROS utilizes a network of applications connected by TCP/IP sockets. TCPROS is the transport layer for messages and services within ROS [1]. Connections between nodes are managed via the ROS master node. To provide the same level of support in mobile operating systems, a mobile-friendly ROS interface needs to be used. Luckily, developers made a Java port of the SDK, called ROSJAVA, for handling the basic features of ROS. Android applications can import the ROSJAVA SDK module for integration with the ROS network. For the Android application to connect to the system, it simply needs to be pointed towards the ROS master. Similar modules, packages, or SDKs will likely to be available in the near future for other mobile operating systems. The rosserial API is used to perform the same functionality on devices that communicate via serial communication devices. Particularly for Arduino, rosserial_arduino is used to implement ROS communication using the *rosserial* API.

In this complex SoS there need to be many potential communication links used in normal operation. In normal operations, the following situations need to be considered:

- Addition or removal of robots at any time

- Seamless system expansion at the will of the user
- Handle and withstand partial system outages (i.e. at the power utility, internet and cloud levels)

If the system is not able to handle these cases, then the flexibility and resiliency of the SoS are not being considered properly.

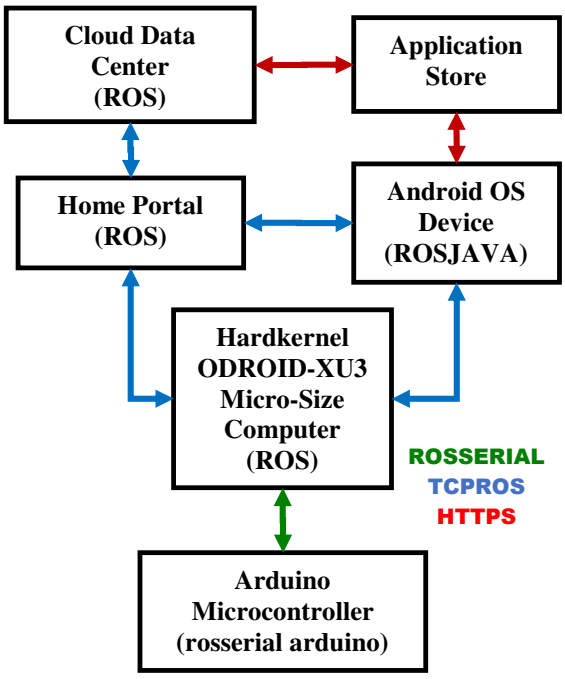

Figure 1: Communication Protocols in System

2.2 Automated Discovery of Nodes in ROS Network

Users should not have to perform multiple time-consuming, and potentially error prone tasks to associate a robot with a ROS network. A simple and efficient method should incorporate some level of autonomy in the connection process. Automatic registration with the ROS master method can be done in a variety of methods.

To generalize many of the approaches, there needs to be at a minimum a standard address or port for which a standardized communication exchange is to take place. Normally, ROS master nodes start a server on socket port 11311 [2]. This port can be changed to accommodate different needs. Multiple nodes on a single computer can be pointed to different ROS masters. The necessary options to be set to enable this type of operation is the same as a distributed ROS configuration, the ROS_IP and ROS_MASTER system environment variables [2]. If the environment variables are set differently between two terminals in Linux, then nodes launched from each terminal can be on different ROS networks.

On startup, the highest level node must broadcast its position as the master of the system. If changes to the master need to be changed, the system will need to change to match the desired hierarchy. Management services on each node will need to be developed to reassign hierarchy when changes are made to the ROS network. This would involve restarting of ROS nodes with the appropriate system environment variables being set to facilitate the re-configuration. As ROS nodes cannot exist without a master, so the choice of host process for the management service cannot be a ROS node. All nodes will poll all known addresses on a network for ROS hierarchy status and connect to the master of the system. If a current master node is at a lower status and is operating at a lower quality of service than an announced master of higher quality, the management service will change the hierarchy.

3 Redesigning the Graphical User Interface for the SoS

3.1 Introduction

There are different models, methods and practices for User Interface (UI) for any level of design projects, and they have been applied to development stages from very beginning. The most important part for development process- in other words, the foundation of successfully implemented UI- is gathering the requirements for the final product that meets the expected design. One may think we should talk about these methods and models, their objectives, structures and their differences for gathering requirements. Comparison of the methodologies and models can be considered for another detailed research, but this study. Simply, after surveying the literature for iterative design aspects, we believe that *User Centered Design* (UCD) concepts will bring better approaches for a fastest and reliable application development process, in particular for our UI design project [3].

During the development process of new graphical user interface (GUI), the procedure resulted with better solutions and mitigated the communication constraints, once applied in *Agile Methodology*. While studying on the GUI, which creates the link between the users and the system, it has realized that the most important drawback for any system is the capability of the application that primarily represents the system

[4]. Actually, this link answers crucial questions for the success of mobile applications, such as if the system or its application is understandable, easy to control and modify by its users [5].

Therefore, the U*sability* aspects for the mobile platforms, applications and *Quality attributes* for software systems have been studied [6]. Then, it is found that there are *Functional* and *Non-Functional Requirements* that evaluates the system designs and development steps based on the system's quality attributes. All of these iterative learning process and application development phases, forced us to study *Requirements Engineering* (RE) and its practices, such as *Paper Prototyping, Contextual Design* (CD) and *Contextual Inquiry* (CI). Furthermore, we applied our solutions, based on user centered objectives, into our project for gathering the requirements for building a better system [7]. This focus help us to understand the users in the first step, and to design a better UI that scales the interaction between the users and the system.

3.1.1 Using Requirements Engineering as a Tool for Developing a SoS

User requirements aim at describing the system functionalities. In other words, they explain what the system has to provide to users for accomplishing their tasks. User requirements include functional requirements, quality attributes based on the system, and other non-functional requirements and hardware environment as well. All of functional requirements mention in *use cases* as a documentation process, and illustrated in *Unified Modeling Language* (UML) diagrams. Following definition states best of our emphasize to imply RE approach into a SoS.

"*Requirements Engineering is the subset of systems engineering concerned with discovering, developing, tracing, analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction*.[8]"

Our approach for gathering the requirements is based on a combination of current frameworks, iterative and incremental development approaches with UCD rules. On the other hand, these steps and their interactions in the software life cycle as a requirement engineering process illustrated at Figure 2 [9]. This four steps process shows basic stages for requirements gathering as a part of RE. Following this process increase the users' involvement on the early steps of the project, and reveals their expectations from the SoS that aims to provide what they really need.

Based on the data that received from the users, some system specifications and functionalities can be extracted from the SoS. We basically applied these steps and discussed requirements gathering in a different perspective by working with CD approach for SoS design. CD provided us new point of view for understanding the system requirements based on actual users, and helped us to understand what their real life requirements are, especially in their work and home environment during real work processes [10]. Moreover, we believe in that this aspect will help to engineers and developers to solve a very important problem, "What do users really need?"

3.1.2 Contextual Design and Contextual Inquiry

The answer for what the users need lies on gathering the requirements, in a correct way. From this point of view, we came up with an idea that describes how to gather requirements more properly and correctly before starting to build the system of systems and a mobile application for it.

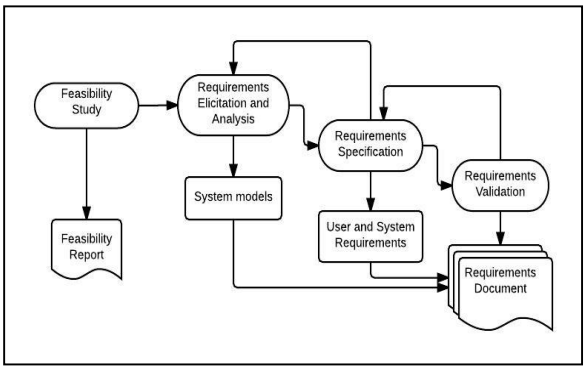

Figure 2: Requirements Engineering Process

Moreover, we also need to express how we can create an approach that allows the system evolve by requirement changes based on the predefined goals. If we make the user know what they really need and create such an environment that matches with their experiences, we can easily receive most correct data for gathering the requirements. Because of these reasons RE can be a useful tool in design stages of a SoS. This idea is applicable by using CI, and following the CD approach for building the system and its application around the users' environment. We simply applied this UCD approach into RE processes for requirements gathering during the design phase.

Furthermore, CI practices for gathering requirements, which is a practice of CD process, have decided to be applied. CI aims to collect requirements by observing the users in their work environment, following their practices, recording their work

processes, and interviewing with them. Briefly, it is a process for generating a background knowledge or documenting the field data from the users' environment for applying those data in requirements gathering process.

3.1.3 Paper Prototyping

Early prototyping is crucial for validating the requirements, testing and usability purposes. Also, prototyping helps us to receive user feedback for main design and implementation processes, which is priceless during the early design phases. Paper prototyping is the cheapest and easiest way for generating user comments for early design sketches. Moreover, we can also get the users involved testing early design ideas with extremely low cost [11]. Papers are affordable to make mistakes and do corrections, comparing with a designing digital and real prototypes after long work hours. We just need paper, pen and pencils, markers and tapes.

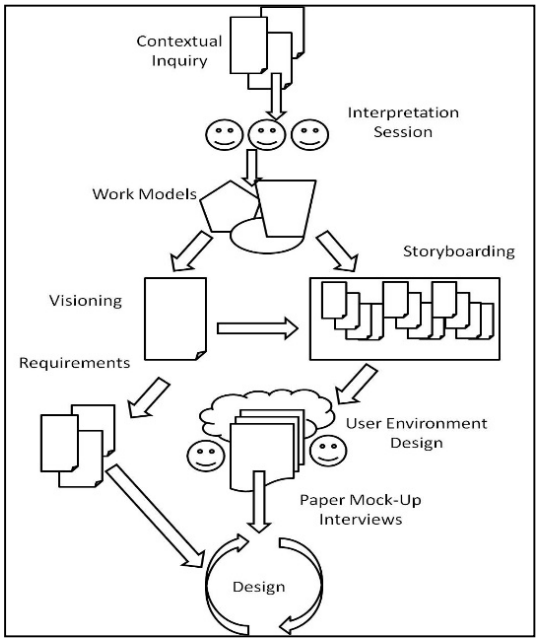

Figure 3:Contextual Design Process and Activities

In addition to these, paper prototyping is adjustable for any level of budgets, and it is an iterative process, new design ideas or big changes can simply add into last prototype by drawing or printing new papers [12, 13].

3.1.4 Summary

According to the enlightening results of literature review, and looking for a better approach to mobile application design for the SoS, previous GUI has been redesigned. For reducing complexity and ambiguity of the application interface, and increasing the quality attributes, the *abilities of the SoS, fundamental usability testing questions applied into the new design process. For the beginning, the main page and the transactions between the menus has been changed. Illustrations for the SoS components added into the GUI to clarify each subsystems. Moreover, for mapping the work environment of the SoS option, very useful specification has embedded into the application, the users can identified every object with geometrical shapes that mapped by the subsystems. As we mentioned before, usability concerns brought up a new point of view on to the design phases; therefore, we improved the navigational tools for the SoS and developed better buttons that works flawlessly with the main view of the application.

4 User Interface Design

The GUI has been redesigned in order to reduce the complexity and increase the usability. In Figure 4 the original Main activity and Clean-E dropdown are represented in (a) and (c). The Main activity helps the user to navigate through different options available for all the robots (Clean-E and Walk-E) in our system, 'MAP' option gives the visual representation of the environment in which robots are performing the jobs and 'SETTINGS' is for changing different system defining parameters. All of these new arrangements are the conclusions of early design and prototyping stages. Since paper prototyping provide faster and cheaper design cycles, by using sketching tools, color pens, papers and web applications, the design changes and new features were easy to apply. Therefore, developers and users were able to consider the improvements and provide a feedback immediately.

From Figure 4 (a) we can observe that the main activity layout has not been properly utilized and the user cannot understand the function of the buttons unless he is familiar with all the terminology related to the project. Both the activities displayed in Figure 4 (a) and (c) have been redesigned in such a way that the layout is properly utilized and more important information is displayed.

In Figure 4 (b) we have a mock up for the redesigned version of Figure 4 (a). Instead of just using the name of the activity for buttons, we have used an illustration of the robot along with its name, which helps user to easily determine the robot that currently chosen. Coloring the menu frames differently and separating the fields not only decreased the usability concerns, also improved the quality attributes of the GUI design.

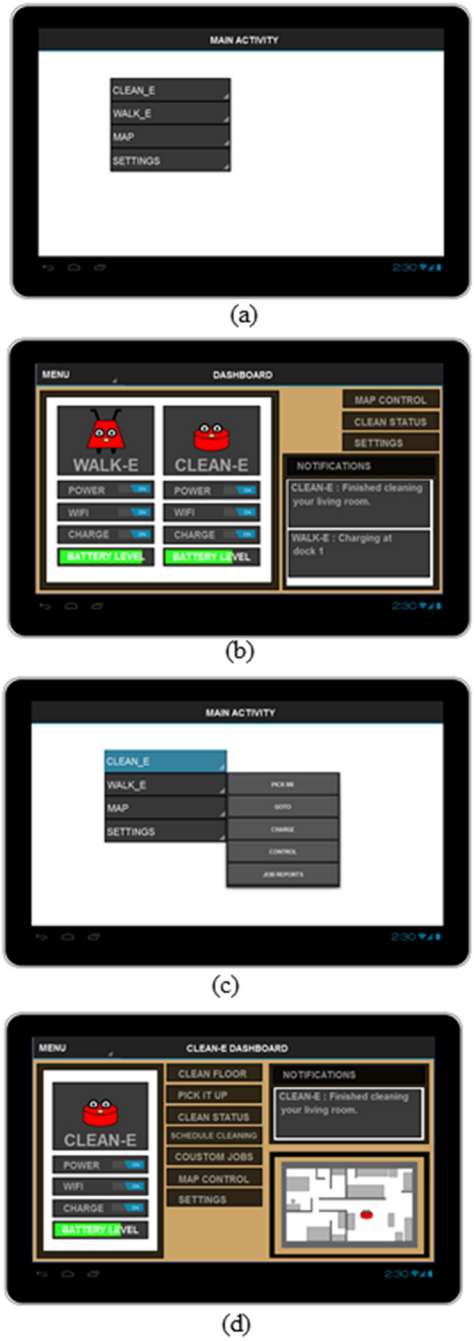

Figure 4: Original components of the GUI are (a) and (c), while (b) and (d) are the respective re-designed components

Options such as turn on/off robot (POWER) or Wi-Fi, command the robot to charge itself (CHARGE), battery level, and cleaning status of the home (CLEAN STATUS) are most used when dealing with the system, providing the control for these operations in dashboard helps the user to access them with ease. The 'NOTIFICATIONS' display the current state of all the robots. Figure 4 (d) is the redesigned version of (c) in which the layout is properly utilized to enable user to have more control over robot and get information from it.

The small map towards in right bottom corner shows the location information of the robot. Options for scheduling cleaning, adding custom jobs, getting status of cleaning job, robot specific settings and commanding the robot to perform tasks like cleaning floor and pick an object are provided in this activity. Having separate dashboard for each of the robots not only allows us to avoid the confusion regarding options and settings but also help proper functioning of the SoS. In Figure 5, original map activity and redesigned map activity are shown. Initial design of 'MAP' activity was intended to provide visual representation of location of robot. In the redesigned version of it, 'MAP CONTROL' more useful option have been added to enhance the user experience.

The 'MAP CONTROL' activity can be directly used to control the robot manually with help of touch control pad towards left bottom, using the location information of robot from the map. When robot is stuck at a location because of an unavoidable or unpredictable obstacle the manual control feature is really handy to get it out of that situation. This activity has an option to switch the manual control between robots by just pressing on the icon of the robot.

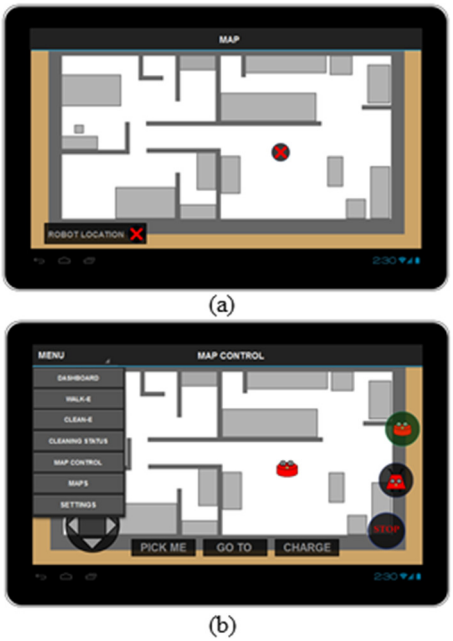

Figure 5: Map activity: (b) is the redesigned version of Map activity shown in (a).

Most used operations such as pick the user from a location, go to a location, charge yourself at docking

station and stop options have been provided to increase the ease of access for the user. More features such as cleanliness report for house and drag and drop option for providing the robot the target location are added to increase the capabilities of the system to provide better service to the user. Figure 6 depicts these features.

Figure 6: Features in redesigned GUI

5 Conclusions

In this paper we proposed an improved software interface for a multi-robot assistive system for the disabled and elderly. We found that a well-constructed requirements gathering process provides key variables, if it applied correctly in early stages. On the contrary of the general assumption, using fundamental Requirement Engineering methods and applying intensive tools, such as Contextual Design and Contextual Inquiry, for gathering the user requirements during the design stages improves representing the SoS and its quality attributes on the users side. In addition to these, interface designs for touch-enabled map navigation, remote viewing and control, and administration of the assistive robot SoS were presented. Important to design of these interfaces is the absolute necessity for ease of control when considering the overall complexity of the system. Improvements to the original software interfaces proved to be highly informative and in line with modern software designs.

References

[1] O. S. R. F. (OSRF). (2013). *ROS/ TCPROS*. Available: http://wiki.ros.org/ROS/TCPROS

[2] O. S. R. Foundation. (2014). *ROS/Troubleshooting - ROS Wiki*. Available: http://wiki.ros.org/ROS/Troubleshooting

[3] H. Beyer, "User-centered agile methods," *Synthesis Lectures on Human-Centered Informatics,* vol. 3, pp. 1-71, 2010.

[4] J. Nielsen, "Why WSJ Mobile App Gets ** Customer Reviews," 2015.

[5] J. Nielsen, "Agile User Experience Projects," 2015.

[6] J. Nielsen, "Usability 101: Introduction to Usability," 2015.

[7] D. C. Gause and B. Lawrence. User-driven design: incorporating users into the requirements and design phase. *Software Testing and Quality Engineering Magazine*. 23-27.

[8] E. Hull, K. Jackson, and J. Dick, "Requirements Engineering. 2005," ed: Springer.

[9] I. Sommerville and P. Stevens, "Software Engineering: AND Using UML, Software Engineering with Objects and Components," 2007.

[10] H. R. Beyer and K. Holtzblatt, "Apprenticing with the customer," *Communications of the ACM,* vol. 38, pp. 45-52, 1995.

[11] J. Nielsen, "Paper prototyping: Getting user data before you code," *Last Reviewed on September,* vol. 22, p. 2007, 2003.

[12] S. Medero, "Paper Prototyping," 2015.

[13] C. Snyder, *Paper prototyping: The fast and easy way to design and refine user interfaces*: Newnes, 2003.