An Efficient Programming Framework for Socially Assistive Robots based on Separation of Robot Behavior Description from Execution

Chandimal Jayawardena[1], I-Han Kuo[2] and Bruce A. MacDonald[2]

Abstract— **One of the main challenges in socially assistive robotics is providing flexible and easy-to-use programming tools for users. Unlike other robots, designing socially assistive robots includes the subject-matter-experts (SMEs) from non-engineering disciplines. Therefore, the provided tools should be suitable for users with less programming experience. On the other hand, socially assistive robotic research involves field trials and user-centric studies, in which user and subject matter expert comments are used to improve the robot applications. Therefore, field programmability and customizability are key requirements. This paper presents a programming framework for socially assistive robots, which satisfies the above requirements; programmability by non-experts, field programmability and customizability. The proposed framework has been successfully implemented, deployed, and tested. Some robots with the framework presented in this paper are already in the commercialization pathway.**

I. Introduction

Software plays a major role in robotics as it is the means of controlling the physical agent and embodying the intelligence in it. Robotic software development is always a difficult and complex task since robotic systems are typicaly concurrent, distributed, embedded, real time, and data intensive. Therefore, there have been many research attempts to develop various methods, frameworks, and tools for easing robot programming tasks [1][2][3][4][5].

In recent research, many software engineering issues of robotics have been identified [6] and various potential solutions have been proposed [7]. However, these attempts aim at solving mainstream robotic problems. There are some unique problems to be solved in the domain of socially assistive robotics [8][9].

Developing socially assistive robots is an emerging and important goal in robotics research [10] [11]. It is an interdisciplinary research area, which requires collaboration between a wide range of disciplines, including robotics, health sciences, psychology, gerontology, and human-computer interaction. This trend is a result of the increasing abilities of mobile service robots and the increasing needs of people for various kinds of help. In particular the median age of the world population is increasing [12]. On the other hand, there is an increasing shortfall in numbers of health professionals and caregivers [13].

There have been many attempts to find assistive robotic solutions to these socio-economic issues. Mobility aids [14], manipulation aids [15], therapeutic aids [16], surgical robots [17], physical and mental rehabilitation robots [18] [19], medication reminding robots [8] and elder-care robots [20] [21] are some examples . Among these solutions, 'socially assistive robots' belong to a distinct category.

Socially assistive robots are different from social robots and entertainment robots, which provide relatively simple human-robot interactions. In contrast, socially assitive robots are expected to provide a broad range of services to support daily activities of users. However, designing such robots poses new challenges, as individualized requirements to cater for the special needs of each user need to be considered [22], [23], [24].

In our previous research, it was found that socially assistive robot demand some special features; i.e. feature richness, rapid prototyping, customizability, involvement of subject matter experts, and end-user programmability [9].

- *Feature richness*: To be useful to the users, socially assistive robots should be capable of providing a range of service. Our previous studies showed that the users expect features which are similar to feature-rich desktop applications.
- *Rapid prototyping*: In socially assistive robot development, there is a focus on field trials with real participants with the objective of improving the design using user feedback. Therefore, the ability develop robot applications rapidly is very important.
- *Customizability*: This is related to the previous point. Customizability enables the inclusion of real-time feedback from the SMEs, pilot groups, end users, and other stake-holders, while reducing the introduction of new bugs and minimizing additional software testing. The software architecture should be flexible enough to accommodate new findings, suggestions, and new requirements, even during testing and deployment phases.
- *Involvement of SMEs*: SMEs are professionals with expert knowledge in a particular domain. For example, doctors, nurses, caregivers, health psychologists, and health care researchers are the SMEs in the health care domain. Inputs of SMEs are mandatory to provide a robotic solution to any application domain. In traditional software development approaches, SMEs were mainly involved in requirement gathering and validation phases, but largely excluded in the development phase. In agile approaches, SMEs are heavily involved in the software design and the development phase, but still the programming is done by programmers. In our approach we tried to extend the involvement of SMEs to the extent of doing actual application development.

[1]C. Jayawardena is with the Department of Computing, Unitec Institute of Technology, New Zealand cjayawardena at unitec.ac.nz

[2]I-Han Kuo and B. A. MacDonald are with the Department of Electrical and computer Engineering, The University of Auckland, New Zealand

- *End-user programmability*: Socially assistive robots are meant for helping people. Therefore, providing tools for end-users to customize the robot behavior without requiring changes to the code is another important consideration.

In order to cater these requirements, effective programming frameworks and tools are necessary. In this paper, one such programming framework, which was implemented and used for real-world robot deployment is presented.

II. Overview of the proposed framework

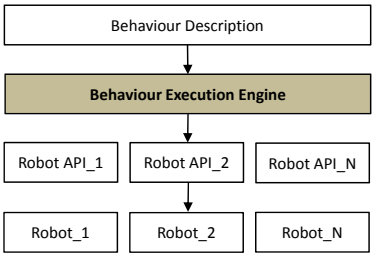


Fig. 1. Overview of the proposed framework

Figure 1 gives a simplified view of the proposed framework. It has 3 key features:

1) Separation of robot behaviour description and the behaviour execution
2) Combining robot actions, user inputs and the graphical user interface (GUI)
3) Behaviour description language for describing robot behaviour as a finite state machine

A. Separation of robot behaviour description and the behaviour execution

The robot behaviour can include anything that the robot is supposed to do. In our implementations, we considered the following behaviours:

- Things displayed on the screen (Components of the GUI such as text, buttons, images, movies, etc.).
- Speech.
- Events that the robot can receive and corresponding actions, such as network events and actions.
- Background actions. i.e. things that the robot can do transparent to the user.
- Events.

In the proposed framework, the Robot Behaviour Description (RBD) is completely isolated from the behaviour execution. The Behaviour Execution Engine(BEE) is responsible for generating the robot behaviour as per the RBD. The BEE is the core software and it contains all the code. However, the BEE does not contain any information about the robot behaviour; i.e. what the robot is supposed to do in a given application scenario. On the other hand, RBD contains all the information pertaining to a given application scenario. Therefore, to develop a new application to suite a given scenario, or to customize an existing application, changes to the core software (BEE) are not necessary. This approach provides the following benefits:

- Since modifications to the execution engine are not necessary to develop a complete new robotic application, the execution engine can be well tested and made highly reliable
- The same execution engine can be re-used on multiple robots to deliver completely different applications, without much software development effort.
- Since the RBD is much simpler than a computer program, it can be authored and edited by someone without any programming knowledge (for example, by an SME)
- New behaviors can be defined rapidly, since programming is not involved
- Changes to the robot behavior can be introduced at any time (even after the deployment) just by editing the RBD

As per the Figure 1, conceptually BEE can support multiple robot APIs through different robot middleware and the bahaviour may be executed on different robots.

The syntax of the RBD, implementation of the BEE and the details of the robot APIs are implementation specific and do not constitute the core concept described above. The implementation specific details are explained in Section III.

B. Combining robot actions, user inputs and the graphical user interface

Usually, in robotic applications, the GUI is not included in the robot behaviour design. Instead, the focus is on robot behaviour such as path planning, navigation, and other actions. However, in most service robot applications, the GUI is a dominant part of the robot behavior, since the user experience highly depends on the audio visual output and the interactions with the robot. In most service robots available in the market, touch screens are used as the main mode of interaction [25].

Therefore, to have effective human-robot interaction, robot actions, user inputs, as well as changes in GUI, should be synchronized. This feature is especially needed in designing a service robot for the healthcare domain [9].

C. Behaviour description language for describing robot behaviour as a finite state machine

As mentioned above, RBD is created as a finite state machine (FSM). Fig 2 depicts a state. An state description includes visual output (GUI), expected events, robot actions, and speech. The structure of RBD is described in Section III.

III. Implementation Details

A. Robot hardware

The mobile platform of HealthBots robot is the Cafero mobile robot developed by Yujin Robot Company, Korea (Fig. 3). The robot hardware consists of a differential drive mobile platform, two single board computers, sonar sensors, microphone, speakers, touch screen mounted on an actuated head, camera, and USB ports.

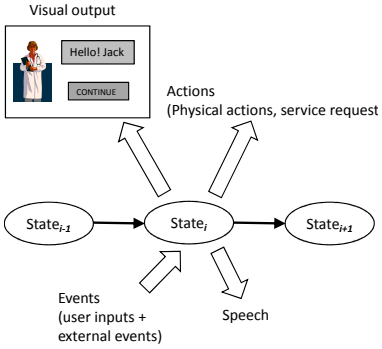


Fig. 2. State behaviour

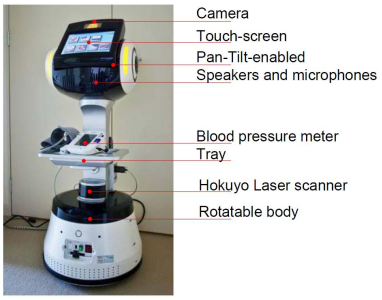


Fig. 3. Healthbot Robot.

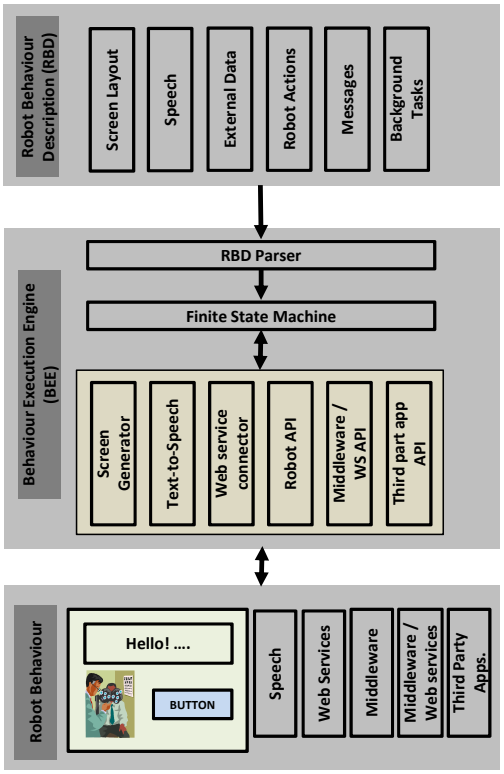


Fig. 4. Relationship of BEE and RBD.

B. Robot behaviour description (RBD)

The relationship between RBD and BEE is shown in Fig. 4.

In the current implementation, an XML based method was used to describe the RBD. However, as pointed out above, the format of the RBD is not important for the key concepts proposed in this paper (Section II). Therefore, any other suitable representation may be used for RBD, with modifications to RBD parser shown in Fig. 4.

RBD describes the robot behaviour as a finite state machine. Fig. 5 shows the format of RBD in the current implementation. Screen, background actions, and expected events are defined for each state. The complete RBD is a collection of states defined in this way, thus representing a finite state machine.

Fig. 6 shows the transition through some sample states with corresponding screens. This may help the reader understand what happens during the execution of robot behaviour. This figure depicts a simplified finite state machine, showing state transition from the starting state (robot's default state) to 'vital signs measurement.'

C. Behaviour execution engine (BEE)

The BEE was developed using C++ and ActionScript 3.0. BEE parses RBD and generates the corresponding finite state machine. This finite state machine interacts with several sub-modules of BEE; screen generator, text-to-speech, web-service connector, robot API, middleware API, web services API, and thirs-part application API. As a result, the robot executes the behaviour, which is a combination of GUI, speech, web service calls, control messages sent through the middleware, and third-party applications (Fig. 4). Specifically, BEE was designed to,

- render the screen layout,
- generate text-to-speech,
- access external web-services to get information,
- control robot movements,
- send and receive messages with back-end systems and various sub-systems,
- save data to databases using web-services, and
- invoke third-party applications

pertaining to the current state.

IV. Results and Conclusion

Overall, the robot was capable of doing vital signs measurement (blood pressure, pulse oximetry, blood glucose strip testing), resident schedule reminding, medication management, communication (with relatives, friends, staff), brain fitness games, entertainment and user authentication.

A. Field trials

Following three field trials were conducted at Selwyn Village retirement centre in Point Chevalier, Auckland, New Zealand to collect psychological results related to human robot interaction.

- Study 1: in public spaces (in independent apartment building common areas and rest home common rooms)

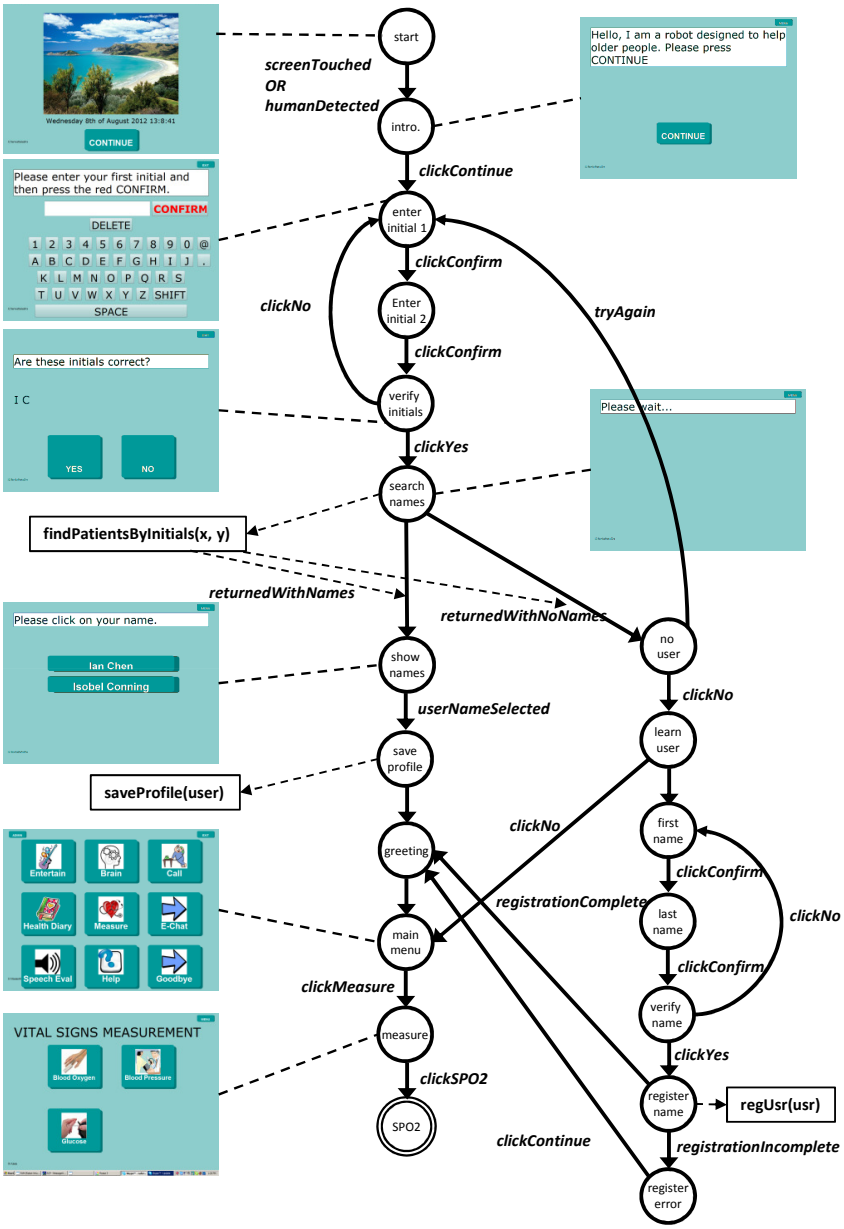


Fig. 6. State transitions: an example

- Study 2: in private spaces (independent living apartments and rest home rooms)
- Study 3: monitoring studies with falls monitoring, wandering and activity monitoring in the rest home.

The robot spent approximately two weeks in an independent living building and approximately two weeks in a rest home. At scheduled times the robot visited apartments or rooms (study 2). The remainder of the day was spent in a public place (study 1). When the robot was in the public place, anyone could approach the robot and interact with it. For study 3, a ZigBee sensor network and other systems has been implemented to receive falls events from wearable accelerometer devices. Once a fall event is received, it was relayed to the robot and the robot reacted by going to the falls location and by starting a remote monitoring session.

In total 67 people interacted with the robot. There were 42 participants in study 1 (public spaces), 25 in study 2 (private spaces), and five in study 3 (falls monitoring). There were two main objectives of these field trials; collecting technical results and psychological results. This paper is focused on technical aspects and psychological results are published elsewhere. Table I shows some statistics collected during these field trials.

B. Software development

For these scenarios, it was required to develop several software versions for the robots. Since the requirements of the healthcare robotics domain was little understood [9], these software versions had to undergo several revisions, both in laboratory environments as well as in the field.

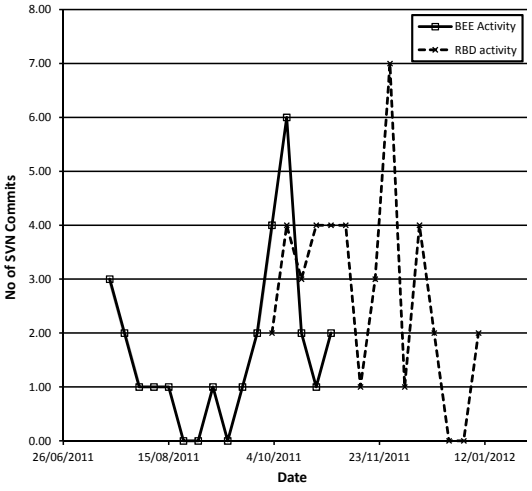
```
<application>
  <state no="...">
    ...
  </state>
  <state no="...">
    <timeout>...</timeout>
    <screen>
      <components>
        ...
      </components>
    </screen>
    <backgroundactions>
      ...
    </backgroundactions>
    <expectedevents>
      <event name="...">
        ...
      </event>
      <event>
        ...
      </event>
    </expectedevents>
  </state>
  <state no="...">
    ...
  </state>
```

Fig. 5. Robot behaviour description (RBD).

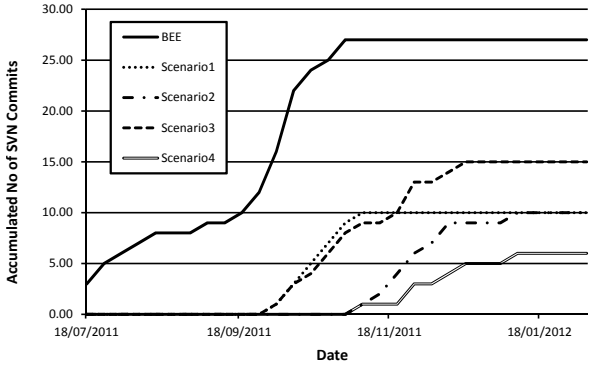
Service	Times used (%)		Avg. rating (%)		Overall rating (%)
	Pub.	Pvt.	Pub.	Pvt.	
Blood pres.	61	59	59	72	65.5
Blood oxy.	31	49	73	80	76.5
Entertainment	48	38	81	79	80
Medication	-	61	-	78	78
Calling	24	11	74	55	64.5
Brain game	32	54	74	74	74

TABLE I

APPLICATION STATISTICS.



(a) SVN activity.



(b) Accumulated SVN activity.

Fig. 7. SVN activity of BEE and RBD development.

All these versions were developed using the programming framework described above. As a result of the architecture, which comprised of RBD and BEE, software development process was managed smoothly, enhancing the involvement of SMEs.

The effect of the architecture on the software development process can be understood from the data presented in Fig. 7. These results explains the advantage of the architecture stated in Section II-A.

Fig. 7(a) shows the number of commits to SVN repository of both BEE and RBD, against time. This clearly shows that at the beginning of the development phase, there were many changes to BEE and eventually it stabilized. RBD development started much later and the latter part of the development phase is entirely dedicated to RBD development, with no changes to BEE.

Fig. 7(b) shows the accumulated number of SVN commits. BEE development starts before RBD development as BEE does not require the exact scenarios of application requirements. Therefore, developers can start with a set of features and focus on making BEE stable. As and when scenario requirements are developed, RBDs can be developed independently. This figure shows the accumulated SVN commits of four scenarios (RBDs). RBD development times are relatively shorter since the development requires authoring an XML file and testing only.

C. Conclusion

During the development, deployment and testing, several observations were made and lessons were learned, related to the above presented software architecture.

a) Increased involvement of SMEs: Since the entire robot behaviour was confined to RBD, which was much easier to understand and edit than a programming language, SMEs were closely involved in the development process. Some SMEs learned the syntax of RBD (XML based) quickly and were able to author and modify robot behavior themselves, with little help from software engineers.

b) Rapid prototyping: During testing and requirement analysis, it was required to develop several robot behaviours for demonstrations and discussions. Software engineers were able to rapidly develop these since changes to BEE (source code) were not required.

c) Increased stakeholder participation: During pilot testing and even during field trials, valuable suggestions and comments were received from the stakeholders such as end-users, caregiver, nurses, and other helth professionals. Most of suggested changes were incorporated to robot behaviour with minimal effort, just by modifying RBD.

d) Improved testing: In robotic applications, usually it is difficult to resolve all software issues in the laboratory and a substantial period of field testing is required to correct all errors. Due to the separation of RBD and BEE, it was possible to handle errors related to robot features and robot behaviour separately. Most errors related to features were resolved in the laboratory since those errors were confined to BEE, which does not contain any behaviour specific data. On the other hand, behaviour specific errors were resolved during field trials. Fixing behaviour specific errors in the field was quick as it did not involve any changes to the source code.

e) Software integration: BEE is a complex distributed software, which was built integrating several research software modules. Therefore, a considerable time and effort were spent on BEE development and testing. However, during the development of BEE software engineers did not have to all end-user requirements, since BEE did not contain robot behaviour specific code.

f) Cost of software errors: In software development, it is generally true that errors introduced at the beginning of the specification phase are likely to only be detected during the use of the product [26]. This is quite true in socially assistive robot design as well. Specially, since the application domain is not well understood, errors in requirements are discovered only during field trials [9]. However, in the proposed architecture, since robot behaviour (RBD) is separated from execution (BEE), user requirements are not required at the beginning and therefore, user requirement errors are not introduced to the core software (BEE). User requirements are required for the development of RBD and errors of these requirements can be identified through field testing and trials and can be quickly fixed without touching the source code.

REFERENCES

[1] D. Brugali, *Software Engineering for Experimental Robotics*. Verlag Berlin Heidelberg: Springer, 2007.

[2] T. Abbas and B. A. MacDonald, "Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 3816–3821.

[3] K. Watanabe, C. Jayawardena, and K. Izumi, "Intelligent interface using natural voice and vision for supporting the acquisition of robot behaviors," in *Proc. 5th IEEE Conference on Sensors*, Oct. 2006, pp. 374–377.

[4] C. Jayawardena, K. Watanabe, and K. Izumi, "Posture control of robot manipulators with fuzzy voice commands using a fuzzy coachplayer system," *Advanced Robotics*, vol. 21, pp. 293–328, 2007.

[5] ——, "Controlling a robot manipulator with fuzzy voice commands using a probabilistic neural network," *Neural Computing Applications*, vol. 16, no. 2, pp. 155–166, 2007.

[6] D. Brugali and E. Prassler, "Software engineering for robotics," *IEEE Robotics Automation Magazine*, vol. 16, no. 1, pp. 9–15, March 2009.

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

[8] I. Kuo, C. Jayawardena, P. Tiwari, E. Broadbent, and B. MacDonald, "User identification for healthcare service robots: Multidisciplinary design for implementation of interactive services," in *Social Robotics*, ser. Lecture Notes in Computer Science, S. Ge, H. Li, J.-J. Cabibihan, and Y. Tan, Eds. Springer Berlin Heidelberg, 2010, vol. 6414, pp. 20–29.

[9] C. Jayawardena, I. Kuo, C. Datta, R. Stafford, E. Broadbent, and B. MacDonald, "Design, implementation and field tests of a socially assistive robot for the elderly: Healthbot version 2," in *Proc. 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 1837–1842.

[10] L. Boccanfuso and J. M. O'Kane, "Charlie : An adaptive robot design with hand and face tracking for use in autism therapy," *International Journal of Social Robotics*, vol. 3, pp. 337–347, 2011.

[11] Y. Yamaji, T. Miyake, Y. Yoshiike, P. R. S. Silva, and M. Okada, "Stb: Child-dependent sociable trash box," *International Journal of Social Robotics*, vol. 3, pp. 359–370, 2011.

[12] W. Lutz, W. Sanderson, and S. Scherbov, "The coming acceleration of global population ageing," *Nature*, vol. 451, no. 7179, pp. 716–719, 2008, 10.1038/nature06516.

[13] *Establishing and Monitoring Benchmarks for Human Resources for Health: the Workforce Density Approach*, ser. Spotlight on Statistics, no. 6, World Health Organization, Department of Human Resources for Health, WHO, 2008.

[14] Y. Hasegawa, J. Jang, and Y. Sankai, "Cooperative walk control of paraplegia patient and assistive system," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 4481–4486.

[15] J. Kofinan, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, Oct. 2005.

[16] K. Kiguchi, T. Tanaka, and T. Fukuda, "Neuro-fuzzy control of a robotic exoskeleton with emg signals," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 481–490, Aug. 2004.

[17] W. Shin and D. Kwon, "Surgical robot system for single-port surgery with novel joint mechanism," *IEEE Transactions on Biomedical Engineering*, vol. —, no. —, pp. —.

[18] H. Krebs, B. Volpe, D. Williams, J. Celestino, S. Charles, D. Lynch, and N. Hogan, "Robot-aided neurorehabilitation: A robot for wrist rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 3, pp. 327–335, 2007.

[19] A. Di Nuovo, D. Marocco, A. Cangelosi, V. De La Cruz, and S. Di Nuovo, "Mental practice and verbal instructions execution: A cognitive robotics study," in *Proc. The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–6.

[20] I. H. Kuo, C. Jayawardena, E. Broadbent, R. Q. Stafford, and B. A. MacDonald, "Hri evaluation of a healthcare service robot," in *Social Robotics*, ser. Lecture Notes in Computer Science, S. Ge, O. Khatib, J.-J. Cabibihan, R. Simmons, and M.-A. Williams, Eds. Springer Berlin Heidelberg, 2012, vol. 7621, pp. 178–187.

[21] I.-H. Kuo, C. Jayawardena, E. Broadbent, and B. MacDonald, "Multidisciplinary design approach for implementation of interactive services," *International Journal of Social Robotics*, vol. 3, no. 4, pp. 443–456, 2011. [Online]. Available: http://dx.doi.org/10.1007/s12369-011-0115-x

[22] A. Tapus, M. Mataric, and B. Scasselati, "Socially assistive robotics [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 35–42, 2007.

[23] C. Jayawardena, I. H. Kuo, U. Unger, A. Igic, R. Wong, C. Watson, R. Stafford, E. Broadbent, P. Tiwari, J. Warren, J. Sohn, and B. MacDonald, "Deployment of a service robot to help older people," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 5990–5995.

[24] R. Stafford, E. Broadbent, C. Jayawardena, U. Unger, I. H. Kuo, A. Igic, R. Wong, N. Kerse, C. Watson, and B. MacDonald, "Improved robot attitudes and emotions at a retirement home after meeting a robot," in *Proc. IEEE RO-MAN*, 2010, pp. 82–87.

[25] K. Tsui, M. Desai, H. Yanco, and C. Uhlik, "Exploring use cases for telepresence robots," in *Proc. 6th ACM/IEEE International Conference onHuman-Robot Interaction (HRI)*, 2011, pp. 11–18.

[26] B. Cohen, *The specification of complex systems*. Reading, MA: Addison-Wesley, 1986.