UI Design for an Engineering Process: Programming Experiments on a Liquid Handling Robot

Farzad Nejatimoharrami[1], Andres Faina[1]
[1]Robots, Evolution, and Art Lab (REAL)
IT University of Copenhagen
Copenhagen, Denmark
Corresponding author: fnej@itu.dk

Andrea Jovanovic[2], Olivier St-Cyr[2], Mark Chignell[2], Kasper Stoy[1]
[2]University of Toronto
Toronto, Canada

***Abstract*—In this paper we describe the development of a user interface for a liquid handling robot. We report on the user interface design process for the interface, beginning with requirements analysis and rapid prototyping followed by heuristic evaluation. We then demonstrate the resulting cloud interface where the robot is controlled remotely in a teleoperation mode to program common protocols in chemistry, and biology, laboratories. We describe the main characteristics of the web interface, focusing on solution strategies used to expose features for programming experiments, remotely. We also report System Usability Scale (SUS) scores obtained by testing the interface with 15 experts in the fields of chemistry, and biology.**

Keywords-component; Human-robot Interaction, Robot Software Architecture Design, Cloud Computing for Distributed and Real Time Robotics, User Interface Design, Usability Research.

I. INTRODUCTION

We have developed a liquid handling robot, called EvoBot [1, 2] which performs typical experiments in chemistry and biology laboratories. EvoBot is designed in three layers as can be seen in Fig. 1. A labelled view of some of the main components in EvoBot is shown in Fig. 2. The head (top layer) moves in x, and y directions, and experiment specific modules, such as single or multi channel syringes, a pH probe, or an OCT Scanner can also be mounted on it. The chemical vessels, such as well plates, petri dishes, PCR plates, tube racks, and troughs are placed on the transparent experimental layer in the middle. The camera at the bottom is the sensing layer which analyses image frames and provides feedback to the ongoing experiment.

EvoBot has been designed to be affordable by taking advantage of open source hardware available in the 3D printing community, and building on top of it. EvoBot costs $1,500, disregarding assembly expenses. This enables even relatively small laboratories to use it. EvoBot has been developed as part of the EVOBLISS EU project, and is used for different applications in seven different countries. The applications include artificial chemical life experiments, microbial fuel cell experiments, visualizing biofilm structures by OCT scans, and routine liquid handling experiments [3,4,5].

EvoBot's application programming interface (API) allows researchers to build on top of the functionality of the robot. Developed in Python, it supports the use of different modules for different applications. EvoBot is an open source system. Examples of how to develop software and create new modules for EvoBot, using the API, may be found in the repository (https://bitbucket.org/afaina/evobliss-software.git.)

Like many complex engineering systems, liquid handling robots have traditionally had complex and difficult to use interfaces [6]. Thus in addition to providing a low cost and extensible open source system, the EvoBot team wanted to provide a good user interface that would be more intuitive and easier to use. It was particularly important to make the robot more efficient [7]. Furthermore, a good interface design was needed to reduce errors in setting up and programming experiments. When dealing with chemicals, errors are potentially dangerous, and a good user interface can improve safety by providing good feedback. Last but not least, the long-term costs of ownership and use of the robot should be lower with a good interface, reducing the need for user training, and will lead to greater efficiency and accuracy in performing experiments.

Advantageously, users can access the robot at any location of their choosing, and on any device. On the other hand, teleoperation can be a challenging task as the operator is remotely located, the operator's situation awareness of the remote environment can be compromised and resulting mission effectiveness can suffer [8]. This should be carefully addressed in the design of the interface.

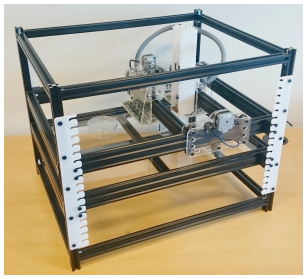

Figure 1. Overview of EvoBot .

II. PROTOCOL-BASED WEB INTERFACE

In this section we discuss the development of a user (protocol web) interface for EvoBot. The protocol interface does not require expert knowledge to program, and provides an intuitive interface for common tasks in a laboratory. The

following section then reports on a heuristic evaluation that was carried out on the user interface prototype.

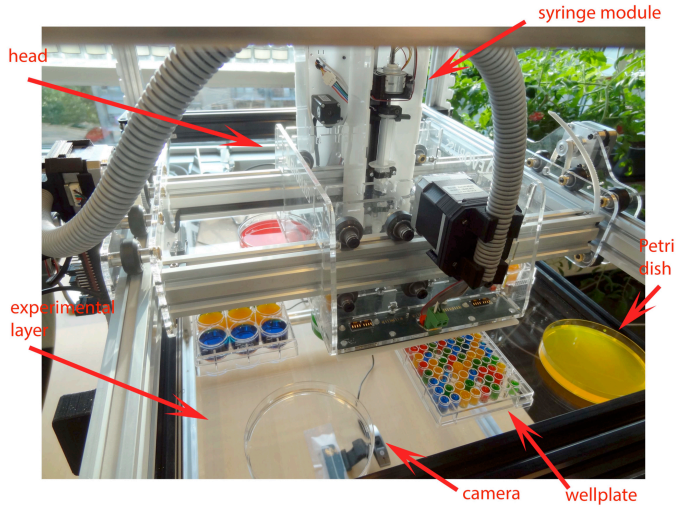

Figure 2. Robot head on top, experimental layer in the middle, and the sensing layer at the bottom.

A. *User Requirement Analysis*

The user interface of the robot was developed using the User-Centred Design (UCD) framework [9]. As a first step in the user centered design, we carried out field visits to interview and observe users and learn to understand their needs. The field visits were carried out in chemistry and biology laboratories at a large North American University. Users carried out real tasks, and their performance and reactions were observed, and recorded. We analyzed the goals users were trying to achieve, how users performed tasks without the interface, the parts they liked or disliked about the work, the difficulties they experienced along the way, and the workarounds they used.

We used these observations to create personas, user stories, and user journey maps. We then tried to uncover and describe users' mental models. We tried to develop the subsequent design based upon an explicit understanding of users, tasks and environments with respect to the domain of liquid handling experiments.

Our field visits with users showed us how tasks should be divided into logical steps that make sense to users. We learned that some experiments may require the syringe tip to be touching the liquid in vessels for better dispensing results. Due to the properties of surface tension, some liquids may need to be blown out while dispensing. We also learned that, when working with micro well plates, the edge wells should be excluded in some experiments, as their evaporation pattern is different from that of the inner wells. Another learning was that using air gaps, volume of air before aspirating any liquid, may be required (depending on the experiment) for better liquid handling performance.

B. *Low Fidelity Prototype*

A focus group was carried out with three experts in human factors and user interface design to determine the overall strategy for constructing the User interface for dilution experiments (a frequent task in a number of scientific areas relevant to EvoBot). The experts recommended a strategy where the user first configures the robot and then programs the experiment, using a menu selection approach. Fig. 3 and 4 show two screenshots for the resulting Balsamiq prototype developed for setting up and programming experiments with EvoBot. Fig. 3 shows the screen for setup and configuration and Fig. 4 shows a screen for programming the experiment.

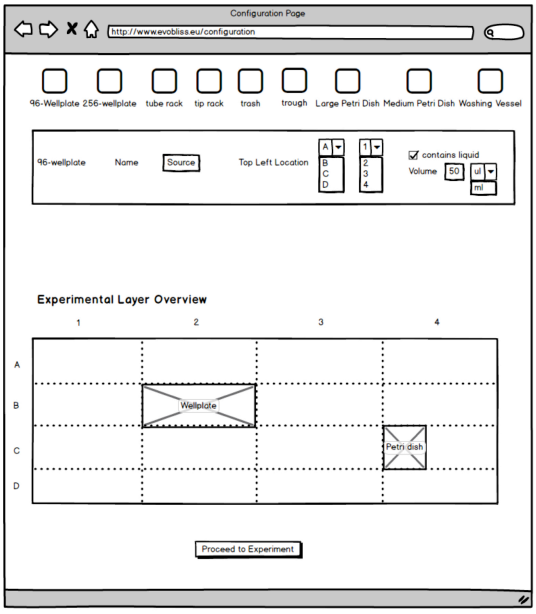

Figure 3. Interface Prototype for Layout Page.

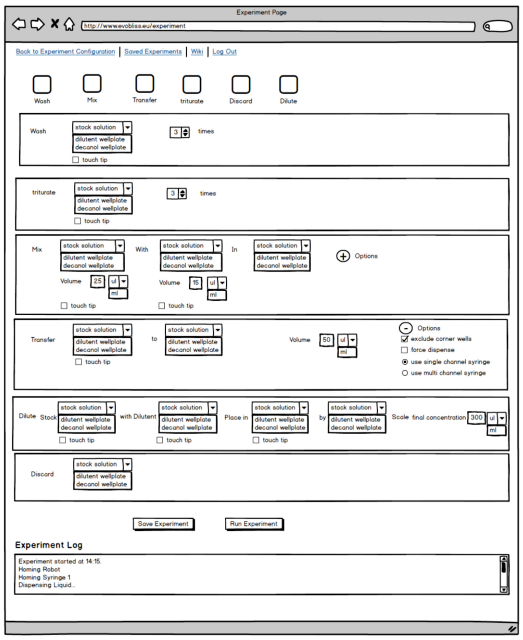

Figure 4. Interface Prototype for Protocol Page

C. *Heuristic Evaluation*

Once the low fidelity prototype was constructed, it was tested with Heuristic evaluation, a method of discount usability engineering that allows efficient formative evaluation of user interface designs based on the assessments of expert reviewers,

guided by a set of heuristics [10]. Clarkson and Arkin (2007) [11] examined a number of lists of heuristics that were subsequently developed, and then integrated them into a new list in the special context of human-robot interaction:

1. *Sufficient information design*
2. *Visibility of system status*
3. *Appropriate information presentation*
4. *Use natural cues*
5. *Synthesis of system and interface*
6. *Help users recognize, diagnose, and recover from errors*
7. *Flexibility of interaction architecture*
8. *Aesthetic and minimalist design*

An expert evaluator used the Clarkson and Arkin list to identify usability problems in the low fidelity prototype of the EvoBot user interface. Each problem identified was labelled with a severity based on the original ratings provided by Nielsen (1995) [12] which vary from 1-4, where 1 is a cosmetic problem only, and 4 is "usability catastrophe".

TABLE I. HIGH SEVERITY PROBLEMS

Interface	Evaluator Observation	Heuristics	Severity
Set up environment	The meaning of the "location" input field is unclear.	3, 6	4
	The experimental layer representation does not clearly reflect the real-world.	3, 4, 5	4
Program Experiment	Terms and phrases do not reflect users' language.	4	4
	Units are manually input from a keyboard.	3. 6	4
	Units are hardcoded into the interface.	3, 4, 6, 7	4
	Required input parameters are missing.	1, 3	4
	Tasks cannot be removed or reordered.	7	4
	The "run experiment" function is unclear.	2, 6	4

Table 1 shows the eight high severity problems that were identified. Two of them were found in the configuration component of the interface and six of them were found in the experiment programming interface. Seven (1-7) of the 8 heuristics listed by Clarkson and Arkin were involved in defining the problems (numbers provided in third column of Table 1).

D. *Final Protocol-based Web Interface*

The protocol-based web interface is a webpage, which enables users to perform common liquid handling tasks. The users first define the layout of experiment to be performed. The experimental layer of the robot is partitioned in rows and columns. The users will choose the type of vessels for the experiment and set their location on the experimental layer as can be seen in Fig. 5. Users will also choose a name for the vessel to refer to in the next step. When the layout of the experiment is set, users proceed to the protocol setup (see Fig. 6) Users can select the task they need to perform, choose from the vessels they have named in layout setup page, and define the parameters for the task. When a series of tasks are set in a protocol, the users can either run the experiment on the robot or save it for later use.

The experiment log at the bottom of the screen provides live feedback from the robot while the experiment is happening. It informs users of the start of experiments, the experiment's progress step by step, and will let them know when the experiments finish. The robot will also warn users if there are any issues during the experiments, such as if the user has forgotten to mount a syringe, or power on the robot, or if a vessel is running out of liquid. The status indicator at the top of the screen informs users if the robot is ready to be used or not.

E. *Implementation of Cloud Web Interface*

To implement this interface, we needed to make some modifications to EvoBot's hardware. We replaced the dedicated computer with a cheap (roughly US 35-dollar) Raspberry Pi 3, which is a single board computer. The Raspberry Pi 3 is connected to the Arduino controlling the robot through serial communications with a standard A-B USB Cable. EvoBot's API resides on the Raspberry Pi 3, and the experiment code is executed on the Raspberry Pi 3. EvoBot uses an extended version of Marlin firmware to add support for control modules on the head. It resides on an Arduino Mega, and is the link between software and hardware. It interprets commands from the G code file and controls the motion accordingly. The Raspberry Pi 3 is connected to the internet through the wireless network adaptor on the board.

On the software side, we use the MEAN stack (MongoDB, Express, AngularJS, Node.js), and web sockets, to realize our interface. Our software architecture can be seen in Fig. 7. The backend for our user interface has been implemented using web sockets, instead of common RESTful APIs. The first reason is that accessing the robot behind a Local Area Network (LAN) requires port forwarding which is not always possible, as it is not allowed on all networks. In addition, leaving a port open all the time while port forwarding exposes the network to security risks. Secondly, traditional restful APIs were not useful in our situation, as the link is terminated after the connection. In our situation, the robot and the interface need to actively listen for events, and respond accordingly.

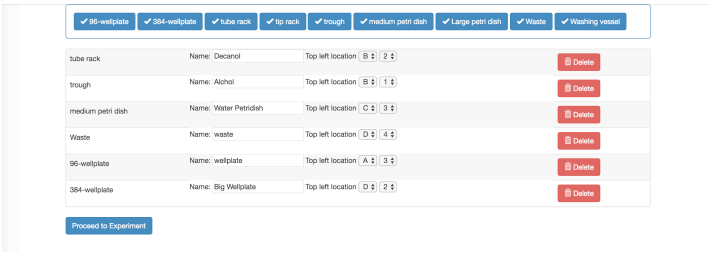

Figure 5. Experiment Layout Page.

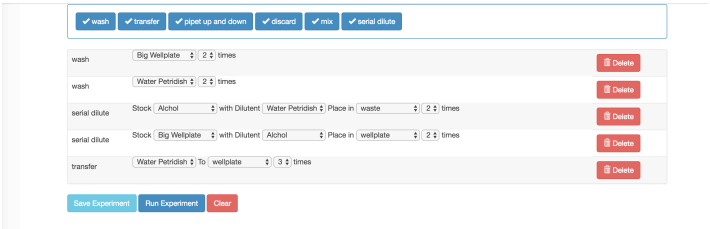

Figure 6. Protocol Setup Page

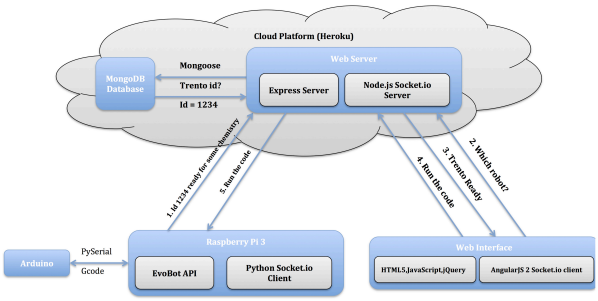

Figure 7. Software architecture of the user interface

The protocol-based web interface allows multiple users to control their corresponding robots simultaneously. This is because each robot has a unique API Key. Therefore users will login with their credentials, and access their own robot. This is useful as multiple copies of EvoBot are used in seven laboratories in different countries.

The advantages of a web user interface includes the fact that it is usable on multiple platforms, software setup is facilitated, and there is an increase in affordability. The users can access the user interface on any device, such as a tablet, a desktop, a mobile phone, or different operating systems, such as OSX, Windows, or Linux. In addition, users don't have to deal with the cumbersome task of installing the numerous packages for the robot as it is taken care of on the Raspberry Pi 3. Last but not least, the price of EvoBot decreases drastically as a dedicated desktop device is no longer required.

III. USER INTERFACE EVALUATION

We evaluated our user interface by 15 users with diverse expertise in chemistry, biology, artificial chemical life, etc, from universities in North America, and different countries in Europe. The user interface was constantly improved by feedback from users. One example of feedback was difference in technical terms. For instance, we replaced absorb with aspirate as they refer to different tasks in laboratories. The term triturate was also replaced by pipet up and down, as not all users were familiar with this expression. Dilution was changed to serial dilution, and the parameters for the task were modified to reflect how chemists perform the task in practice.

We used the System Usability Scale (SUS) to evaluate our user interface. SUS is a highly robust and versatile tool for usability testing [13]. We got an average score of 87 in our usability testing. Another interesting observation was that the users with experience using a liquid handling robot interface gave a higher usability interface to the protocol-based user interface.

IV. CONCLUSIONS

In this paper we describe the development of a user interface for a liquid handling robot, and highlight the strategies that allowed us to develop a web interface for remote control of EvoBot, as well as the UCD process that we followed to design the cloud interface. Using a special version of heuristic evaluation designed for robot interfaces we were able to identify key usability problems in the prototype and develop a web user interface that is currently being implemented on EvoBot and that represents a significant advance of prior user interfaces for liquid handling robots.

ACKNOWLEDGMENT

E.U. Future and Emergent Technologies supported this work through EVOBLISS grant no. 611640, and members of the EVOBLISS consortium evaluated the robot user interface. Florian Blauert, Pavlina Theodosiou, and Silvia Holler provided feedback on the interface. Naveen Venayak, Brian Nguyen, and Jenna Blumenthal at the University of Toronto also provided constructive feedback on the user interface

REFERENCES

[1] A. Faina, F. Nejatimoharrami, K. Stoy, P. Theodosiou, B. Taylor, and I. Ieropoulos, ": An open-source, modular liquid handling robot for nurturing microbial fuel cells," in ALIFE16 Conference Proceedings. MIT Press, Cambridge, MA, USA, July 2016, pp. 626–633.

[2] A. Faina, F. Nejatimoharrami, and K. Stoy, "Evobot: An open-source, and modular liquid handling robot," Accepted for publication in The IEEE Robotics and Automation Magazine.

[3] F. Nejatimoharrami, A. Faina, J. Cejkova, M. Hanczyc, and K. Stoy, "Robotic automation to augment quality of artificial chemical life experiments," in ALIFE16 Conference Proceedings. MIT Press, Cambridge, MA, USA, July 2016, pp. 634–635.

[4] F. Nejatimoharrami, K. Stoy, and A. Faina, "An open-source, low-cost robot for performing reactive liquid handling experiments," in SLAS2016 Conference, January 2016.

[5] P. Janska, Z. A., F. Nejatimoharrami, A. Faina, K. Stoy, and J. Cejkova, "Collective behaviour in droplet systems," in 43rd International Conference of SSCHE, May 2016.

[6] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey," Found. Trends Hum.-Comput. Interact., vol. 1, no. 3, pp. 203–275, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1561/1100000005

[7] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common metrics for human-robot interaction," in Proceed- ings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Inter- action, ser. HRI '06. New York, NY, USA: ACM, 2006, pp. 33–40.

[8] J. Y. C. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 6, pp. 1231–1245, Nov 2007.

[9] J. Preece, Y. Rogers, and H. Sharp, Interaction Design: Beyond Human-Computer Interaction. New York, NY, USA: John Wiley & Sons, Inc., 2015.

[10] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in SIGCHI Conference Proceedings, ser. CHI '90. ACM, 1990, pp. 249–256. [Online]. Available: http://doi.acm.org/10.1145/97243.97281

[11] E. Clarkson and R. C. Arkin, "Applying heuristic evaluation to human-robot interaction systems," in Flairs Conference Proceedings, 2007, pp. 44–49.

[12] J. Nielsen, "Severity ratings for usability problems," Papers and Essays, vol. 54, 1995.

[13] H. A. Yanco, J. L. Drury, and J. Scholtz, "Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition," Human–Computer Interaction, vol. 19, no. 1-2, pp. 117–149, 2004.