WearSkill: Personalized and Interchangeable Input with Wearables for Users with Motor Impairments

Ovidiu-Andrei Schipor
MintViz Lab, MANSiD Center
Stefan cel Mare University of Suceava
Suceava, Romania
ovidiu.schipor@usm.ro

Laura-Bianca Bilius
MintViz Lab, MANSiD Center
Stefan cel Mare University of Suceava
Suceava, Romania
laura.bilius@usm.ro

Radu-Daniel Vatavu
MintViz Lab, MANSiD Center
Stefan cel Mare University of Suceava
Suceava, Romania
radu.vatavu@usm.ro

ABSTRACT
We introduce WearSkill, a web application that implements personalized and interchangeable input for wearable computing. We outline functional and quality requirements for WearSkill and present the engineering details of its implementation using web technology. We emphasize the interchangeability of input modalities with various wearables, e.g., touch input on a smart ring vs. mid-air gestures of the hand wearing a smartwatch vs. voice input detected by the microphone from a pair of smartglasses, towards personalized input for users with various motor abilities. Our findings, from a study involving twenty-one people with motor impairments, show that WearSkill can provide accurate recommendations for personalized input modalities that match 85.3% with users’ own preferences.

CCS CONCEPTS
• Human-centered computing → Accessibility technologies;
• Software and its engineering;

KEYWORDS
Wearables, software architecture, event-based processing, touch input, gesture input, voice input, motor impairments, accessibility

ACM Reference Format:

1 INTRODUCTION
Wearable devices have become increasingly available in the context of Ubicomp and IoT environments. With a global market of USD 40.65 billion in 2020 [3], wearables are becoming mainstream due to their many valuable functions, e.g., health and fitness tracking, and integration with other mainstream devices, such as smartphones. As end users who are part of this market, people with motor impairments experience challenges interacting with wearables [22] that were not designed to be accessible in the first place. The type and severity of the motor condition determine the specificity of such challenges [9,13,16,31] with impact on the feasibility of the input modalities that can be effectively used to interact with off-the-shelf devices, such as touch input [26], hand gestures [8], foot gestures [6], or movements of the head [24]. For example, Mott et al. [13] reported barriers regarding the physical accessibility of VR devices, such as putting HMDs on and taking them off, adjusting the head strap, or maintaining view of the controllers, and Malu et al. [9] reported challenges regarding button, swipe, and tap-based interactions for smartwatch input. A workaround is system design that features alternative input modalities that best match individual users’ abilities [33]. Personalization and interchangeability of input modalities are also useful for interactions performed in conditions characterized by situational impairments [1,5,20].

We are interested in this work in software architecture design for wearables featuring personalization and interchangeability of input modalities. Our contributions are as follows: (1) a set of software design requirements for input personalization and interchangeability with wearables; (2) WearSkill, our web-based application that implements our set of requirements towards personalized, multi-device, multi-modality input with wearables; and (3) a study with N=21 people with motor impairments showing 85.3% accuracy for predicting users’ preferences for wearable devices and input modalities that best match their motor abilities.

2 RELATED WORK
Prior work has documented challenges experienced by people with motor impairments during interactions with computer systems of many kinds. From mouse [32], keyboard [4], and remote control [25] input to interactions with mobile devices [12,27,30] to large touchscreen displays [14] to wearables [9,10,31], a wide range of devices and input modalities have been scrutinized for accessibility. Among these, accessible interactions with wearables have been addressed to a lesser extent: in their systematic literature review of wearable interactions for users with motor impairments, Şiean and Vatavu [22] reported that hand gestures have been disproportionately favored (41.6%) compared to other input modalities, e.g., head gestures (23.4%) or voice input (13.0%). Based on their findings, Şiean and Vatavu proposed the WISE framework, a set of recommendations to increase the accessibility of wearables: (W) exploring diverse designs of wearables, (I) new input modalities and techniques for accessible wearable interactions, (S) more user studies and evaluations, and (E) extending wearable interactions with other devices. Due to its direct connection to our scope, we adopt the WISE framework as the conceptual foundation for WearSkill.
Several unique characteristics of wearables make them suitable for personalized and interchangeable input: diverse form factors, placement close to or in contact with the body, and varied sensing options to detect user input, including taps and touch input, movements of the fingers and hands, voice, eye gaze, and others; see [21,22]. Also, multiple devices can be worn together, e.g., a smartwatch and a smart ring, enabling the use of one or another according to context. This flexibility, however, raises several technical challenges for software architecture designs and for the algorithms employed to recognize user action across multiple devices.

Various technical solutions have been proposed for input with heterogeneous devices [18,19]. One example is Euphoria, a scalable, event-driven software architecture for implementing interactions across heterogeneous devices in smart environments. Euphoria consists of five software layers: Producers, Emitters, Engine, Receivers, and Consumers. The outermost components, represented by Producers and Consumers, implement the specific details required by different platforms, operating systems, and APIs. In the center of the architecture, Emitters and Receivers abstract the strategy for exchanging messages with all devices sharing the same interface and communicating via the same protocol. The business logic for the communications resides at the Engine level, where messages are routed between devices. In this work, we reuse the open-source Euphoria [18] as the middleware for implementing WearSkill.

3 WEARSKILL

We introduce WearSkill, our web-based application that implements personalized and interchangeable input with wearables. We start by presenting functional requirements for WearSkill in accordance with the WISE framework [22]. We then leverage the SQuaRE [7] model to select six quality requirements, which we implement with three technological and three design patterns approaches.

3.1 Design Requirements

Based on the directions set by the WISE framework [22], we formulate four functional requirements (F1 to F4) for WearSkill:

F1. **Wearables.** WearSkill must be flexible to integrate a variety of wearables and their integration must require only minimal software changes at the outer layer of the software architecture. We consider wearable devices that have built-in Wi-Fi connection and support communication protocols for the web, such as WebSocket.1

F2. **Interactions.** WearSkill facilitates execution of system functions on various output devices, e.g., a PC or a TV set, with input performed with wearables. To this end, users perform a command with the wearable they prefer in a given context. WearSkill stores associations between commands performed with wearables and system functions of output devices, e.g., a wave of the hand wearing a smartwatch turns on the TV.

F3. **Studies.** WearSkill offers out-of-the-box support for logging input data to enable more studies to learn about how users employ wearables and how they personalize input.

F4. **Extension.** The main goal of WearSkill is to enable personalized and interchangeable input with wearables as the middleware for a distributed user interface [28].

![Figure 1: Design requirements for WearSkill (left) and an overview of its software components (right) for personalization and interchangeability of input with wearables.](image)

We choose six quality requirements from the ISO/IEC 25010:2011 SQuaRE [7] model (Systems and Software Quality Requirements and Evaluation) that are directly relevant to wearables [17] and to the scope of the WearSkill application, respectively:

Q1. **Modularity** refers to splitting the software system into discrete units that interact with each other through interfaces. This requirement acknowledges the heterogeneity of wearables (F1) and keeps the adapter layer as thin as possible.

Q2. **Reusability** indicates the degree to which a specific asset can be reused in another configuration of the system. Since WearSkill enables input with heterogeneous devices (F1), developers that build on WearSkill can reuse existing functionality to accommodate new wearables and output devices.

Q3. **Interoperability** is the ability of two or more modules to communicate with each other based on a common standard. WearSkill integrates different types of wearables (F1) by means of dedicated modules that employ standard communication protocols to enable a variety of interactions (F2).

Q4. **Replaceability** refers to the possibility to replace a component with another, equivalent one. The interchangeability of wearables enables the use of different input devices (F1, F2) to execute a given system function on an output device.

Q5. ** Appropriateness** indicates the degree to which the software system meets end-user needs in a personalized way. This property connects to the functional requirements F3 and F4.

Q6. **Learnability** specifies the capacity of the system to help the user transition from novice to expert mode. Also, better understanding end-user needs for wearable interactions (F3) can lead to better future versions of WearSkill (F4).

The first two quality requirements, Q1 and Q2, set the condition for low-level software modules, requirements Q3 and Q4 refer to high-level software components, and Q5 and Q6 specify the characteristics of the interaction between users and WearSkill. To implement these quality requirements, we adopt three technological (T1 to T3) and three design patterns (D1 to D3) approaches; see Figure 1, left. Due to the high portability of web applications, we choose web technology (T1) for the development of the WearSkill user interface. We also choose JavaScript as our programming language of choice (T2) since it runs on a variety of platforms, either in a web browser or a runtime environment. Communications between the software components are implemented with Euphoria [18] and, thus, employ

HTTP and WebSocket protocols (T3). In terms of design patterns, we establish WearSkill on the SOLID principles [11]. We use dependency injection, i.e., injecting of dependencies rather than creating them inside the object, to obtain a strong separation between object creation and utilization (design pattern D1). Also, WearSkill is divided into several layers (models, views, view models, presenters, controllers, repositories, and gateways) that communicate through interfaces (D2). To simulate usage scenarios (D3), we employ high coverage end-to-end testing. To implement the functional requirements F1 to F4, we designed six software components for WearSkill (C1 to C6 in Figure 1), presented in the next subsection.

3.2 Implementation

The software components of WearSkill play specific roles in achieving input personalization and interchangeability. The interaction flow, from the specification of the user profile to runtime monitoring of the system, as follows:

C1. Profile enables users to enter personal data (in accordance with the functional requirement F2) if they represent their motor symptoms, e.g., slow movements, low strength, etc., which we adopted from [2]; see Figure 2, left.

C2. Preferences employs recommendations generated by a machine learning model that uses information from Profile to suggest wearable and input modalities (according to functional requirements F1 and F2). In the current version, we implemented support for three types of wearables (smartwatches, smartglasses, and smart rings) and four input modalities (touch input, hand motion, head motion, and voice input), but extensions can be further integrated.

C3. Registration of input and output devices is performed under the Devices section (requirements F2 and F3). Each device is identified by its MAC address and associated with one of the following categories: smartwatch, smartglasses, smart ring, or a generic output device, on which system functions will execute following input with the first three types of devices.

C4. The Input component is in charge of detecting and recognizing touch, motion, and voice input (functional requirements F1 and F2) that users can particularize by providing training samples; see Figure 2, right. Voice input consists in word commands recognized using the Google Speech-to-Text API. Touch, stroke-gesture, and motion input are implemented with the SP point-cloud gesture recognizer [29].

C5. Input and output devices are associated in the Commands section (requirement F4). Commands run on output devices, e.g., turn on the TV. The same command can be entered with different input modalities to implement interchangeability.

C6. Users can supervise the entire process from the Runtime Monitor component (functional requirement F3). For example, if input from a given device is not recognized or system functions are not executed on the output device, warnings help the user in identifying the cause.

We developed the user interface on top of Vue.js, a progressive JavaScript framework that enables clear separation between views and view-models. Back-end models run on the Node.js platform and communicate with the front-end and the input and output devices via HTTP and WebSocket protocols. For end-to-end testing, we employed Cypress, and specified more than 200 tests covering usual scenarios. To test WearSkill, we developed client web applications for three wearables: Samsung Galaxy Watch 3, Gear Fit 2 (which we mounted on a custom 3D printed support to be used as a smart ring), and the Vuzix Blade smartglasses. Next, we focus on the recommender implemented under the Preferences component to match wearables and input modalities to users’ motor abilities.

3.3 Personalized Recommendations for Input Modalities with WearSkill

The Preferences software component integrates a machine learning model that employs eleven self-reported motor symptoms, entered by the user as yes/no responses when setting their WearSkill profile (Figure 2, left), to suggest recommendations for wearables and input modalities that best match users’ motor abilities. The symptoms are: slow movements, spasm, low strength, tremor, poor coordination, rapid fatigue, difficulty gripping, difficulty holding, lack of sensation, difficulty controlling direction, and difficulty controlling distance, which we adopted from [2]. We conducted a user study to collect the data needed to train this software component.

3.3.1 Participants. A number of N=21 people with motor impairments (spinal cord injury located at various vertebrae, spina bifida, traumatic brain injury), aged between 28 and 59 years old (M=43.3, SD=8.2 years), were recruited via a non-profit organization providing technical assistance to people with disabilities.

3.3.2 Method. We used an online questionnaire to elicit preference ratings for various combinations of wearables and input modalities using 5-point Likert scales with items ranging from 1 (not suitable for me) to 5 (very suitable). Our experiment was a within-subjects design with two independent variables: input modality (nominal variable with four conditions—touch input, hand motion, head motion, and voice input) and wearable (nominal variable with three conditions—smartwatch, smartglasses, and smart ring).

3.3.3 Dataset. We collected a total number of 21 (participants) × 11 (symptoms) = 231 records representing predictors and 21 (participants) × 3 (wearables) × 4 (input modalities) = 252 preferences representing recommendations for the Preferences component.

3.3.4 Results. We employed Scikit-learn [15] to evaluate various classifiers on our dataset using hyperparameter tuning with grid search optimization [23] and leave-one-out cross-validation. Our choice of algorithms included linear models and discriminant analysis, Nearest Neighbor classifiers, decision trees, ensemble methods, semi-supervised models, neural networks, and Support Vector Machines models. During training, we grouped participants’ ratings obtained with the 5-point Likert-scales in two classes: do not recommend (corresponding to the Likert scale items 1 and 2) and recommend (items 3, 4, and 5), respectively. Hyperparameter tuning enabled us to cover a wide range of configurations for the classifiers consider in our evaluation, such as regarding the number of neighbors to use for the K Nearest Neighbors classifier or the best solver

---

3https://vuejs.org
4https://www.cypress.io
Table 1: Accuracy rates for predicting users’ preferences for wearables and input modalities from their motor symptoms.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Smartwatch</th>
<th></th>
<th></th>
<th></th>
<th>Smartglasses</th>
<th></th>
<th></th>
<th></th>
<th>Smart ring</th>
<th></th>
<th></th>
<th>Mean accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Touch input</td>
<td>Hand motion</td>
<td>Head motion</td>
<td>Voice input</td>
<td>Touch input</td>
<td>Hand motion</td>
<td>Head motion</td>
<td>Voice input</td>
<td>Touch input</td>
<td>Hand motion</td>
<td>Head motion</td>
<td>Voice input</td>
</tr>
<tr>
<td>Decision Tree [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>57.1%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>Gradient Boosting [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>57.1%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>Ada Boost [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>52.4%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>KNeighbors [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>52.4%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>Label Propagation [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>52.4%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>Radius Neighbors [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>52.4%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>57.1%</td>
</tr>
<tr>
<td>Logistic Regression [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>76.2%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>47.6%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>61.9%</td>
</tr>
<tr>
<td>Multinomial Naïve Bayes [ref.]</td>
<td>85.7%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>52.4%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>47.6%</td>
</tr>
<tr>
<td>Label Spreading [ref.]</td>
<td>71.4%</td>
<td>90.5%</td>
<td>81.0%</td>
<td>81.0%</td>
<td>90.5%</td>
<td>57.1%</td>
<td>90.5%</td>
<td>95.2%</td>
<td>100%</td>
<td>95.2%</td>
<td>100%</td>
<td>47.6%</td>
</tr>
</tbody>
</table>

*The base estimator was a Random Forest [ref.] classifier.

Figure 2: Snapshots of the WearSkill application. On the left, symptoms of the motor condition enable generation of personalized recommendations of input modalities for wearables. On the right, a personalized gesture is provided for touch input.

4 CONCLUSION AND FUTURE WORK

We presented WearSkill, a web-based application that implements input personalization and interchangeability for wearables. In this paper, we focused on the engineering details of WearSkill, but future work is envisaged to evaluate its usability. We also plan to continue the development of WearSkill (http://www.ceed.usv.ro/mintviz/projects/WearSkill) towards an open-source solution on the web that integrates with IoT devices.

ACKNOWLEDGMENTS

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI-UEFISCEDI, project no. PN-III-P2-2.1-PED-2019-0352 (276PED/2020), within PNCDI III.