Software Architecture Based on Web Standards for Gesture Input with Smartwatches and Smartglasses

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

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

ABSTRACT
We employ the ISO/IEC 25010 standard for systems and software quality requirements and evaluation to present a proposal of a software architecture design for gesture input with two types of wearables: smartwatches, which have already entered mainstream, and smartglasses, forecast to become a growing market due to increased adoption of augmented reality applications. We present the technical details of our software architecture and describe practical implementation aspects that employ web standards.

CCS CONCEPTS
• Human-centered computing → Ubiquitous and mobile computing; • Software and its engineering → Software design engineering; Requirements analysis.

KEYWORDS
Software architecture, software quality, gesture input, wearable computing, web standards.

ACM Reference Format:

1 INTRODUCTION
Gesture input for interactive computing systems has been extensively examined in the scientific literature in terms of gesture taxonomies, models of user performance, gesture recognition algorithms, interaction techniques, and methods to study user input [10,12,15]. Gestures also represent a preferred input modality for wearables that embed touch and motion sensors [4–6,8,17]. However, wearables come in many flavors in terms of hardware and software platforms and, thus, dedicated software architecture is needed to support interactions among such heterogeneous devices while offering extensibility and personalization [11]. In this work, we employ the Software Product Quality Model from the SQuaRE ISO/IEC 25010 standard [7] to present a proposal of a software architecture that facilitates gesture input acquisition for wearable devices with touch and/or motion sensing capabilities. We focus on smartwatches, a category of wearables that has already entered mainstream [16], and smartglasses, a growing consumer market [9].

2 SOFTWARE ARCHITECTURE DESIGN
The ISO/IEC 25010 standard [7] defines a comprehensive software quality model with requirements that are grouped in eight categories—functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability—and several properties, e.g., time behavior, resource utilization, and capacity for performance efficiency. From these properties, we select nine directly relevant for gesture input with wearables (Figure 1), which we set as requirements for our software architecture.

Our architecture design features three components: Hub, Device, and Link. The Device component runs on the wearable that acquires gesture input. The Hub provides the user interface with our software architecture, e.g., specification of wearables and definition of gesture commands. For instance, the Hub allows adding, removing, and configuration of smartwatches and smartglasses devices as...
well as acquisition, processing, and recognition of touch and motion gestures collected via the Device component. The Link implements full-duplex communications between the Hub and instances of the Device component running on each wearable. This decoupling enables a distributed architecture that can process gestures from multiple devices, e.g., to enable cross-device input [1, 2, 4].

Starting from our quality requirements, we adopted an engineering approach centered on the following six pillars:

1. **Web-based user interface.** The presentation layer of the Hub rests on HTML elements, which enables portability across many devices (requirements R1, R2, R7, and R8).
2. **Web-based languages.** The three components use JavaScript to enable portability across a variety of platforms, while the Link runs as a node.js server (R5 and R6). This design choice follows similar approaches from the HCI community that employ JavaScript to implement interactions on mobile and wearable devices [3, 13, 14].
3. **Web-based communication standards.** Both half-duplex and full-duplex communication among the software components are implemented via web technologies via the HTTP and WebSocket protocols (R4-R6, R8-R9).
4. **IoC-DI.** We implement inversion of control through dependency injection for the decoupling of creation and usage of software objects and of high and low-level classes (R1 to R3).
5. **MVM-C.** By following the Model-View-ViewModel Coordinator structural design pattern, we implement the principle of separation of concerns for software classes (R1-R3).
6. **Open-source.** Our architecture is available under an open-source license (Figure 2) to foster further extension (R2-R7).

This engineering approach makes our software architecture suitable for collecting gesture input from any smartwatch or smartglasses device that supports WiFi and WebSockets. The Device component needs to be implemented for each wearable to send gestures to the Hub that implements the recognition algorithm and informs the Link about the result. As long as the wearable device supports JavaScript, either natively for devices running Tizen or via a web browser, touch and motion gestures are straightforward to collect.3

3 **RESOURCES AND FUTURE WORK**

Our source code is freely available on the web; see Figure 2. The web page features (i) an overview of the architecture, (ii) the source code of our software with a demonstrative implementation of touch gesture and acceleration motion gesture acquisition for Samsung Galaxy smartwatches and Vuzix Blade AR smartglasses, and (iii) a walk-through video showing how to use the Hub user interface to configure wearables in our architecture. Future work will address applications for cross-device input with smartwatches and smartglasses supported by our software architecture.

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

**REFERENCES**


