Web2cHMI: A Multi-Modal Native User Interface Implementation for Accelerator Operations and Maintenance Applications

Reinhard Bacher
Deutsches Elektronen Synchrotron DESY, Notkestrasse 85, 22607 Hamburg, Germany
(Dated: January 2nd, 2017)

The advent of advanced mobile, gaming and virtual or augmented reality devices provides users with novel interaction modalities. Speech, finger- and hand gesture recognition are commonly used technologies, often enriched with data from embedded gyroscope-like motion sensors and 3D graphical visualization features. This paper discusses potential use cases of those native user interface technologies in the field of accelerator operations and maintenance. It describes the conceptual design and implementation of a native, single-user, multi-modal human-machine interface (*Web2cHMI*) which seamlessly incorporates actions based on various modalities in a single API. *Web2cHMI* belongs to the *Web2cToolkit* framework which is briefly introduced. Finally, an outlook to future development steps and potential consequences is presented.

I. NATIVE USER INTERFACE EXPERIENCE

Zooming applications by performing a pinch gesture at touch-sensitive displays, talking with the personal assistant to retrieve information from the internet, or controlling video games through a gaming console recognizing arm and body motions are all popular and intuitive interface features currently in common use. Moreover virtual or augmented (mixed) reality applications providing e.g. a real three-dimensional user experience are enjoying increasing popularity. These technologies, well known in the consumer market, have extremely enriched the way in which people interact with their devices.

Even in the rather conservative market of industrial applications, native user interface technologies are gaining in importance, e.g. to simplify quality assurance of manufacturing processes in the car or aircraft industry or to improve the efficiency of warehouse logistics or maintenance work in machine building industry.

In addition, a novel concept known as "App", optimized for these unique technological features, has been introduced, which has revolutionized the conceptual design, look-and-feel and handiness of graphical user applications. Prominent features of app-like user applications include replacing complex menu trees by intuitive navigation or browsing patterns and focusing solely on task-specific contents by hiding all information not relevant to these cases.

Hardware commissioning and maintenance use cases might profit from such novel interaction capabilities (modalities). For instance the alignment of mirrors mounted on an optical table to adjust a laser beam spot often requires a "third hand". Interacting with control applications via spoken commands could be an appropriate alternative. Likewise wearing rough and dirty working gloves during cooling water maintenance work is not adequate for touch sensitive devices. Interacting via hand or arm gestures might be a better choice. Accessing online documentation or viewing virtual 3D-models is often indispensable for efficient inspection work. Wearing see-through augmented reality glasses controlled by head movements, hand or arm gestures or spoken commands displaying routing schemes alongside with control applications or overlaying schematics could substantially improve measurement operations or maintenance work in the field.

Even control room work provides use cases for novel interaction modalities. Remote-controlling an overhead mounted screen showing some overview or control application panels might be considerably simplified by recognizing spatial gestures such as clenching a fist or snapping fingers. Beam steering requires uninterrupted eye contact with trend charts or other display updates. Controlling a virtual knob by recognizing hand rotation or moving a virtual slider providing tactile feedback could eliminate the risk of losing device focus which often happens with mouse-based operations.

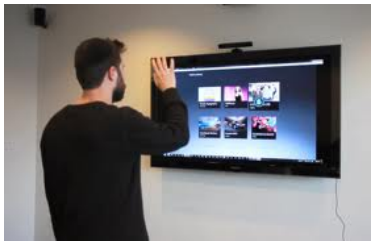
Todays youth are more than familiar with these novel interaction capabilities and app-like user applications. Providing up-to-date tools for future control room operators and maintenance technicians appears to be a must. It will foster the acceptance of native user interaction technologies for accelerator operations and maintenance and quickly widen the list of corresponding use cases.

II. USER INPUT DEVICES

With the advent of ubiquitous mobile communication and entertainment as well as considerably improved optical projection and graphics processing capabilities novel native interaction devices have found their way into the consumer and industrial market. This includes

- Single- or multi-touch sensitive displays available for desktop, notebook or tablet computers and smart phones,
- Optical sensors capable of tracking movements and recognizing gestures such as Leap motion controller (Fig. 1a) [1] or Microsoft Kinect motion controller (Fig. 1b) [2], wearables with muscle activity sensors like the Myo gesture control armband (Fig. 1c) [3],

(a) Leap motion controller



(b) MS Kinect motion controller

(c) Myo gesture control armband

(d) Ultrahaptics UHDK5

(e) Vuzix M100

(f) MS HoloLens

FIG. 1: Native input devices

or multi-axis gyro sensors embedded e.g. in smart phones or head-mounted displays,

- Camera-based eye-tracking and attentive environments analysing video and audio recordings,
- Ultrasonic actors providing tactile feedback, e.g. Ultrahaptics UHDK5 (Fig. 1d) [4],
- Miniaturized classical optical projection devices (e.g. pico projectors), and advanced projection systems integrated into personal eyeglasses such as Zeiss Digital Smart Lens [5],
- See-through augmented reality devices with embedded camera and multi-axis gyro sensor (e.g. Goggle Glass [6], Epson Moverio BT-200 [7], Vuzix M100 (Fig. 1e) [8], Microsoft HoloLens (Fig. 1f) [9]), and
- Powerful cloud-based speech recognition systems used by Apple Siri [10], Google Now [11], or Microsoft Cortana [12]) and stand-alone speech recognition systems such as the CMU Sphinx-4 toolkit [13].

III. BASIC CONSIDERATIONS

Today's users of accelerator control applications have developed intuitions based on click-like interactions. In an accelerator control room the mouse is still the standard user input device to interact with graphical applications. Being well accepted by the operators it provides a very accurate pointing capability and standardized user actions normally associated with graphical widgets. Mouse-based or single-finger touch-based interactions are highly reliable unambiguous single-user actions. They are best suited for complex applications containing a wealth of graphical widgets or tree-like menu structures.

Thus the introduction of any new interaction capability will be accompanied by a serious paradigm change regarding how software programmers design graphical operations and maintenance applications and how operators or maintenance personnel interact with them.

A. Gesture-Based Interaction

Spatial hand- and arm gestures, as well as multi-finger touch-based gestures, provide only a rough pointing capability, and experience shows that the users arm tends to fatigue quickly, a phenomenon known as "gorilla arm". In addition head gestures such as turning or nodding might also be considered. However, not all head gestures are equally suitable and might cause symptoms similar to travel sickness. The handiness of gestures might also profit from tactile feedback, e.g. simulating a virtual knob.

In practice only a very limited number of gesture types are available which are partially standardized and not a priori associated with graphical widgets. Consequently the design and look-and-feel of applications must accommodate these restrictions.

In general gestures are less precise and might be performed slightly differently by varying persons. They tend to be less reliable and might require a specific arming and disarming procedure to prevent the user from unwanted interaction.

Spatial gestures are not limited to single-user interaction only. It depends on the technology of the gesture recognition device used how many gestures from different persons can be tracked individually. The Myo gesture

control armband worn at the users forearm represents a true single-user device while Microsofts Kinect motion controller is capable to handle various gestures even from different individuals at the same time.

B. Speech-Based Interaction

In contrast to other interaction modalities the recognition of spoken commands does not provide any pointing capability at all.

On the one hand the huge word pool of human languages is a clear plus factor. On the other hand the context-dependent ambiguity of the vocabulary (e.g. the term mouse describes both an animal and a computer peripheral device) and the accent- or dialect-dependence of human speech pose a big challenge for the recognition algorithms involved. The latter might even require an individual training of the speech recognition system involved.

Video conferencing experience shows that from the audio technical point of view the recognition of spoken commands as well as the reliable identification of the speaker might suffer from ambient noise and the interference of multi-user inputs. Wearing high-quality microphones based on bone conduction or specific noise filtering and cancelation technologies might be preferable. Advanced augmented reality glasses such as Microsoft's HoloLens provide already excellent speech recognition capability.

Limiting the allowed vocabulary or grammar structure is preferable to achieve reliable spoken command recognition ability and ensuring an unambiguous word-to-action mapping. Similar to gesture recognition a specific arming and disarming procedure is capable of preventing the user from unwanted interaction.

C. Application Design Criteria

Due to the limited number of available gestures or spoken commands, a multi-page application design consistent with the app concept, where each page provides a well-confined and standardized functionality with an unambiguous input-to-action mapping, appears to be best suited. In addition, both a clearly arranged option to launch applications and a handy and intuitive procedure to browse between pages and applications have to be provided.

Supporting and combining different input modalities at the same time such as tracking of head or eye movements and speech-based interactions might considerably enhance the performance and reliability of native user interfaces.

D. Sociological Issues

Daily experience shows that many individuals tend to use their mobile devices without any inhibitions in a public but anonymous environment. However, control room or maintenance work is typically performed by small groups of individuals. Consequently, it is expected that the wish to preserve individual privacy and the potential impact on the communication between team members might become an issue for using native user interface technologies.

IV. IMPLEMENTATION

This section reports ongoing R&D work and describes a Web-based native user interface implementation (*Web2cHMI*) for accelerator operations and maintenance applications in the context of the *Web2cToolkit* Web service collection [14].

A. Web2cToolkit Web Service Collection

The *Web2cToolkit* is a collection of Web services, i.e. servlet applications and the corresponding Web browser applications, including

- *Web2cViewer*: Interactive synoptic live display to visualize and control accelerator or beam line equipment,
- *Web2cViewerWizard*: Full-featured graphical editor to generate and configure synoptic live displays,
- *Web2cArchiveViewer*: Web form to request data from a control system archive storage and to display the retrieved data as a chart or table,
- *Web2cArchiveViewerWizard*: Full-featured graphical editor to generate and configure archive viewer displays,
- *Web2cToGo*: Interactive display especially designed for mobile devices embedding instances of all kinds of *Web2cToolkit* Web client applications,
- *Web2cGateway*: Application programmer interface (HTTP-gateway) to all implemented control system interfaces,
- *Web2cManager*: Administrators interface to configure and manage the *Web2cToolkit*.

Web2cToolkit provides a user-friendly look-and-feel and its usage does not require any specific programming skills. By design, the *Web2cToolkit* is platform independent. Its services are accessible through the HTTP/HTTPS protocol from every valid network address if not otherwise restricted. A secure single-sign-on

user authentication and authorization procedure with encrypted password transmission is provided. The Web 2.0 paradigms and technologies used include a Web server, a Web browser, HTML5 (HyperText Markup Language), CSS (Cascading Style Sheets), AJAX (Asynchronous JavaScript And XML) and WebSocket.

The *Web2cToolkit* client applications are coded in native JavaScript or in Java within the Google Web Toolkit framework [15] finally compiled to HTML- / CSS-tags and JavaScript code. They are running in the clients native Web browser or in a Web browser embedded in a mobile app or desktop application. This approach is compatible with almost all major browser implementations including mobile versions. The server-side *Web2cToolkit* services are provided by Java servlets running in the Web servers Java container. The communication between client and server is asynchronous. All third-party libraries used by the *Web2cToolkit* are open-source.

The *Web2cToolkit* provides interfaces to major accelerator and beam line control systems including TINE [16], DOOCS [17], EPICS [18], TANGO [19] and STARS [20]. The toolkit is capable of receiving and processing video frames or a continuous series of single images.

B. Web2cToGo

The *Web2cToGo* Web client application embeds *Web2cToolkit*-compliant Web client applications. Each application may consist of multiple pages. At most fifteen applications can be launched at a single time. The user can switch between the different applications to select the active application as well as between the different pages of the active application to select the active visible application page. All inactive applications or application pages are hidden.

Web2cToGo has three different views. The Explorer View (Fig. 2) displays a set of icons to select a user application. It provides launching or displaying of already launched but currently hidden user applications. In addition it indicates the currently available modalities and the corresponding arming status.

Switched to Operation View (Fig. 4) the *Web2cToGo* client application displays the active application and allows interacting with the graphical widgets of the application.

The Navigation View (Fig. 3) previews the selected user application. It allows switching between user applications, switching to Explorer or Operation View, closing the current user application, browsing between application pages, and zooming, resizing or scrolling pages of user applications.

C. Web2cHMI

Web2cHMI is a Web-based, platform-neutral, single-user human-machine interface which seamlessly combines actions based on various modalities provided by input devices commonly available from the consumer market. All input modalities supported can be used simultaneously including

- 1D/2D flat gestures including single- and multi-finger gestures
- 2D/3D spatial gestures including hand- and arm-gestures
- 3D head movements including yaw, pitch and roll
- English spoken commands.

The *Web2cHMI* API comprises all actions needed to control *Web2cToolkit*-compliant Web applications such as application browsing, display zooming or executing commands associated with interactive graphical widgets.

Web2cToGo implements and provides a test environment for identifying intuitive and handy user actions as well as investigating the proper structure, design and operability of multi-modal accelerator operations and maintenance applications.

1. Supported User Input Devices

Web2cHMI supports various user input devices including

- Mouse
- Touch-sensitive display
- Leap motion controller
- Myo gesture control armband
- Epson Moverio BT-200 smart glass
- Vuzix M100 smart glass
- Microphone

The Leap motion controller and the Myo gesture control armband are connected locally with their corresponding host device (desktop, notebook or tablet computers) through USB and Bluetooth, respectively. The raw sensor signals are processed by a local Web server which communicates with the *Web2cHMI* via Web Sockets. Both smart glasses supported provide gyroscope-based head tracking capability. CMU Spinx-4 includes a speaker-independent continuous speech recognition engine and is entirely written in Java programming language.

All gesture-capable devices provide orientation data being used to position a virtual cursor label at an application window. Unlike mice or touch-sensitive displays gesture recognition devices such as Leap motion controller, Myo gesture control armband or smart glasses with head tracking capability do not allow an accurate positioning of the cursor label.

FIG. 2: *Web2cToGo* (Explorer View)

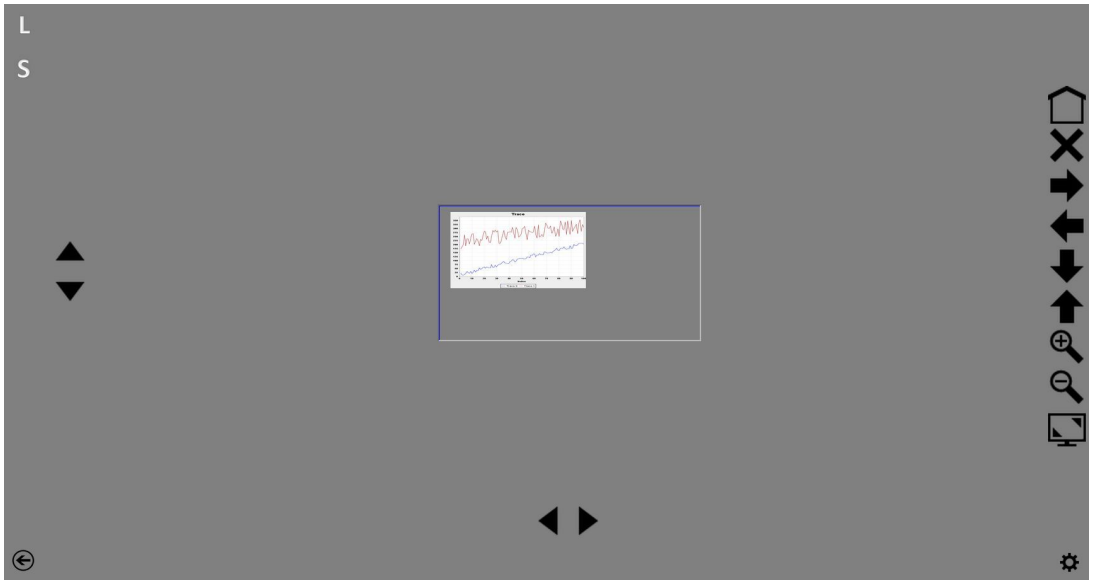


FIG. 3: *Web2cToGo* (Navigation View)

2. *Supported User Input Actions*

Web2cHMI recognizes various primitive, i.e. input device-specific gestures including

- Mouse: *Click*, *Move*
- Touch-sensitive display: *Tap*, *Move* / *Swipe*, *Pinch* (two fingers)
- Leap motion controller: *Key-Tap*, *Swipe*, *Open-Hand*, *Closed-Hand*, *Circle*
- Myo gesture control armband: *Double-Tap*, *Wave-Out* / *Wave-In*, *Fingers-Spread*, *Fist*
- Smart glass: *Move-Fast* / *Move-Slow*, *Roll*

In addition, enriched gestures formed by primitive gestures followed by moves or rotations etc. are supported. Examples are *Fingers-Spread & Clockwise Rotation* or *Sideward-Left Long Swipe*. If applicable or required by ergonomics principles, different gestures may be applied by right or left handed individuals.

The CMU Sphinx-4 speech recognition system knows a *Web2cHMI*-specific vocabulary listed in a dictionary containing the decomposition of the words in distinct phonemes. Phonemes are the basic units of sound which can be ultimately identified by the speech recognition engine. In addition pre-defined grammar rules are used

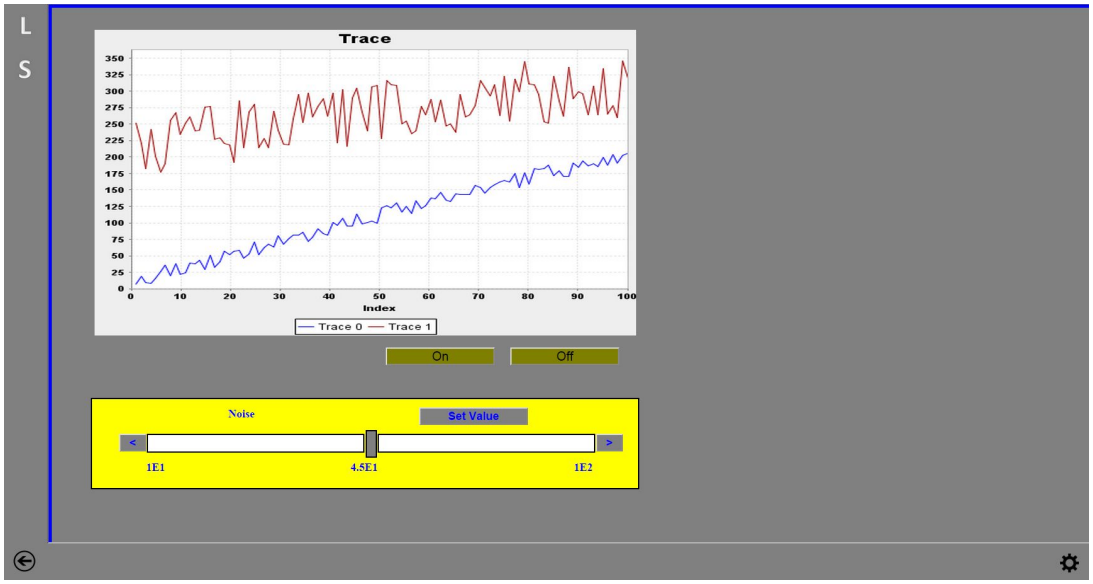


FIG. 4: *Web2cViewer* embedded in *Web2cToGo* application (Operation View)

to enhance the recognition performance and reliability of spoken commands such as "*Browse Up*" or "*Lot More*".

If a user input has been successfully recognized the next user input recognition is momentarily inhibited while the cursor label is fixed to the center of the application window to notify the user (gesture input only).

In addition to avoiding unwanted responses to unintentional user input, the recognition ability must be armed and disarmed explicitly by the user:

- Leap motion controller: *Key-Tap* / *Key-Tap*
- Myo gesture control armband: *Double-Tap* / *Double-Tap*
- Smart glass: *Clockwise-Roll - Counter-Clockwise-Roll* / *Counter-Clockwise-Roll - Clockwise-Roll*
- Microphone: "*Ok*" / "*Sleep*"

3. Software Architecture

Web2cHMI analyses user actions recorded by any modality-specific input device attached and maps the recognized gestures, spoken commands, head movements or even mouse clicks etc. through the Common Multi-Modal Human-Machine Interface to unambiguous API commands.

The API commands are used to control both the *Web2cToGo* framework application itself and the embedded *Web2cToolkit*-compliant Web applications. Fig. 5 sketches the user input data-flow within *Web2cToGo*. Besides speech recognition which is performed by the *Web2cToGo* servlet (Java) at the Web server all recognition algorithms are implemented as client-side JavaScript modules being executed by an HTML5-compliant Web browser. Commands dedicated to an embedded *Web2cViewer* or *Web2cArchiveViewer* application are redirected at server side to the corresponding Web application client and handled in exactly the same way as direct mouse- or touch-based user input (Fig. 6).

4. API Examples

Following the proposed multi-page concept providing a well-confined and standardized functionality (see section Application Design Criteria) each *Web2cViewer* application page might contain a single widget instance of each of the following interactive widget types (Fig. 4). According to their type, the interactive widgets are capable of performing a specific, predefined user action such as opening a vacuum valve or changing a set value of a power supply:

- On-type Button (user action = "On")
- Off-type Button (user action = "Off")
- Slider (user action = "Change Set Value")
- Chart (user action = "Zoom Data")

In addition a page might contain an unlimited number of passive *Web2cViewer* widgets such as labels or value fields.

Triggered by each of the following user actions recorded by the corresponding input device the common API command "Change Set Value Small Positive Step" increases a set value of an attached controls device in small steps using a slider widget in a *Web2cViewer* interactive synoptic live display:

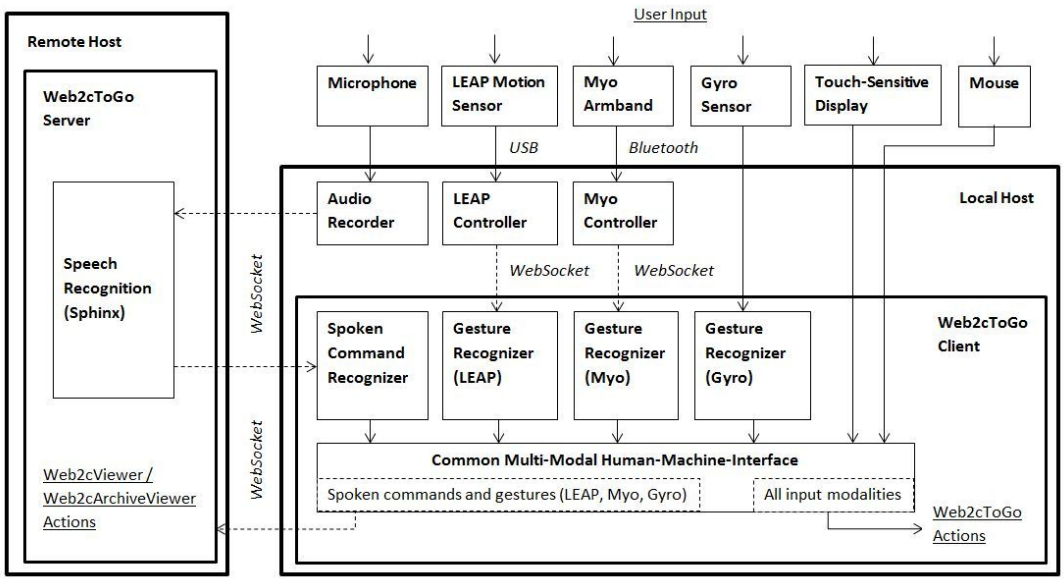


FIG. 5: *Web2cToGo* / *Web2cHMI* user input data flow

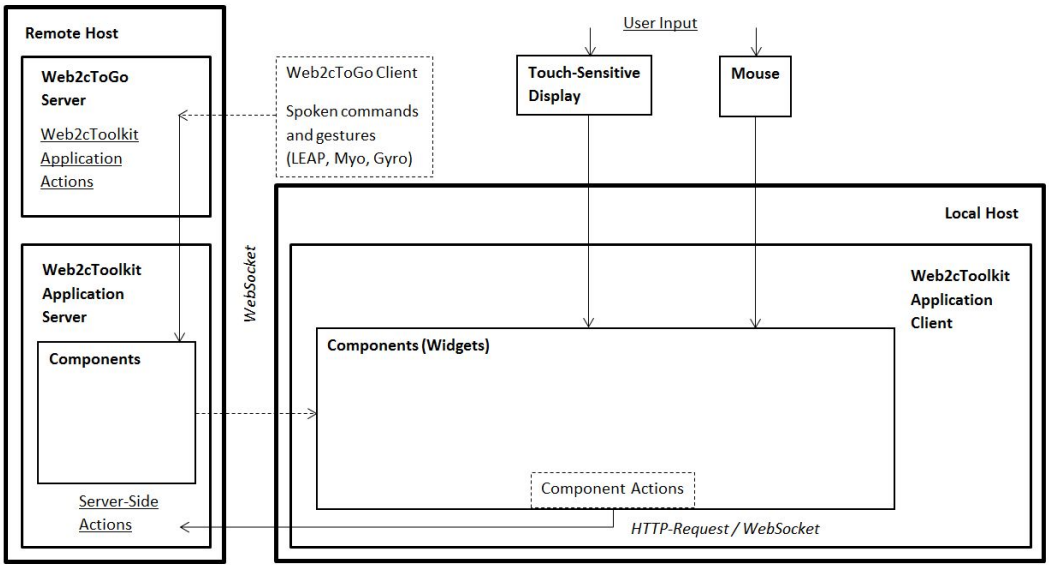


FIG. 6: *Web2cToolkit* application / *Web2cHMI* user input data flow

- Mouse: *Click* ">"-button and *Click* "Set Value"-button of the slider widget
- Touch-sensitive display: *Tap* ">"-button and *Tap* "Set Value"-button of the slider widget (right or left hand)
- Leap motion controller: *Downward Long Swipe* (right or left hand)
- Myo gesture control armband: *Fist & Clockwise Rotation* (right or left arm)
- Smart glass: *Upward Left-Tilted Move-Slow*
- Microphone: "More"

Similarly, sets of interactive widgets have been defined for *Web2cArchiveViewer* application pages (Fig. 7). For instance the common API command "Single Chart Get Data" requests data from control system archive storage to be displayed in a single chart page of a multi-page *Web2cArchiveViewer* application in response to each of the following user actions:

- Mouse: *Click* on "Get Data" button

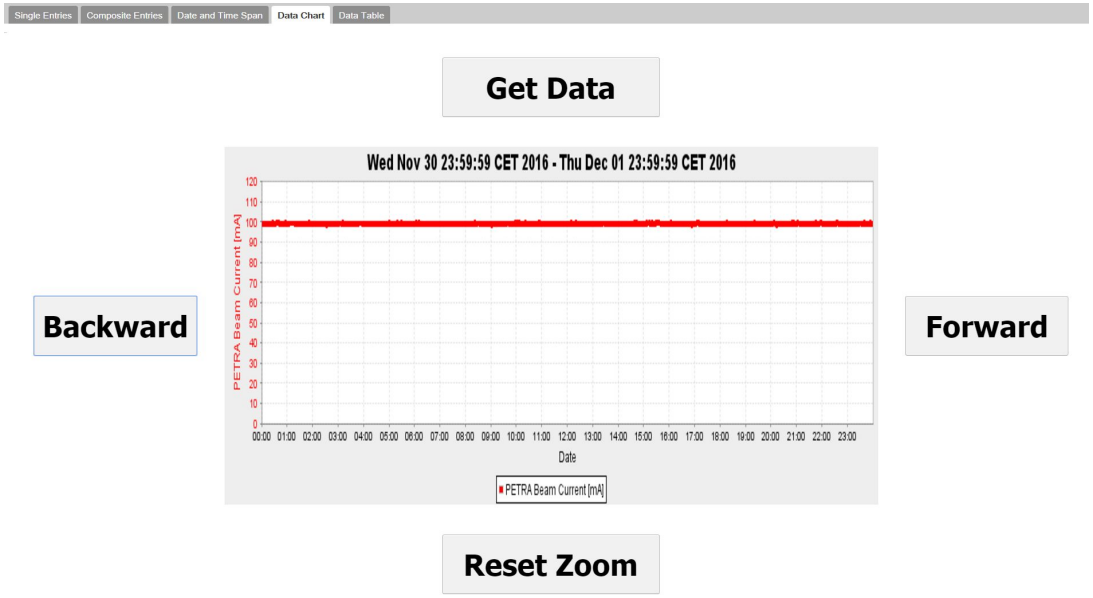


FIG. 7: Data chart page of multi-page *Web2cArchiveViewer* application

- Touch-sensitive display: *Tap* on "Get Data" button (right or left hand)
- Leap motion controller: *Downward Long Swipe* (right or left hand)
- Myo gesture control armband: *Wave-In & Downward Move* (right or left arm)
- Smart glass: *Downward Move-Fast*
- Microphone: "*Download*"

A compilation of all API commands implemented by *Web2cHMI* can be found elsewhere [21].

V. FINAL REMARKS

The work described in this paper is still an R&D project. It explores only a limited fraction of capabilities provided by native user interface input devices available for consumer and industrial applications. It is inspired and abetted by a growing number of use cases and technological developments in this field. Thus it is expected that the acceptance level of native user interaction technologies will steadily increase in the future.

The applicability of native user interaction technologies for accelerator controls and maintenance has to be studied in detail.

- To compete successfully with standard mouse-based interactions performant gesture and speech recognition procedures have to be provided which are reliable under any circumstances even in a multi-user or in an otherwise disturbed environment.
- Well-designed user application concepts best suited for advanced interaction devices providing novel interface features and environments have to be developed or explored. The optimal combination of different modalities and the most intuitive, most common and best matching sets of gestures and spoken commands have to be defined and the usefulness and potential predominance of this approach have to be explored and proved in real field tests.
- In addition, it is an open issue how a future control room might look. Will it be a camera supervised collective multi-user attentive environment or rather an environment for operators wearing individual single-user augmented reality glasses? It is obvious that along with operational and technical issues sociological issues must be considered as well.

Finally, the next steps to introduce these novel interaction technologies for accelerator operations and maintenance have to be discussed. Is an intermediate step preferable, for instance providing user applications adapted for touch pads or interactive tables yet keeping the traditional application look-and-feel? The work described in this paper is consistent with this approach. Or should a larger step be taken, where mouse-click interactions are omitted entirely, thereby skipping the familiar mouse-centric application design pattern e.g. implemented by Microsoft HoloLens-compliant applications?

ACKNOWLEDGMENTS

The author thanks P. Duval (DESY) for carefully reading the manuscript.

[1] Leap Motion Inc., "Leap Motion Controller", https://www.LEAPmotion.com (2017), Last accessed: 2017-01-02.

[2] Microsoft Corporation, "Microsoft Kinect Motion Controller", https://developer.microsoft.com/en-us/windows/kinect (2017), Last accessed: 2017-01-02.

[3] Thalmic Labs Inc., "Myo Gesture Control Armband", https://www.myo.com (2017), Last accessed: 2017-01-02.

[4] Ultrahaptics, "UHDK5 Touch Development Kit", https://www.ultrahaptics.com/products/touch-development-kit (2017), Last accessed: 2017-01-02.

[5] Carl Zeiss AG, "Zeiss Smart Lenses Get Right What Google Glass Got So Wrong", https://www.wired.com/2016/01/zeiss-smart-glasses (2017), Last accessed: 2017-01-02.

[6] Google Inc., "Google Glass Head-Mounted Display", https://developers.google.com/glass (2017), Last accessed: 2017-01-02.

[7] Epson Inc., "Moverio BT-200 Augmented Reality Smartglass", https://epson.com/moverio-augmented-reality (2017), Last accessed: 2017-01-02.

[8] Vuzix Corporation, "M100 Smartglass", https://www.vuzix.com/Products/m100-smart-glasses (2017), Last accessed: 2017-01-02.

[9] Microsoft Corporation, "Microsoft HoloLens Mixed Reality Smartglass", https://www.microsoft.com/microsoft-hololens (2017), Last accessed: 2017-01-02.

[10] Apple Inc., "Apple Siri Personal Assistant", https://www.apple.com/ios/siri (2017), Last accessed: 2017-01-02.

[11] Google Inc., "Google Now Personal Assistant", https://www.google.com/search/about/features/01 (2017), Last accessed: 2017-01-02.

[12] Microsoft Inc., "Microsoft Cortana Personal Assistant", https://www.microsoft.com/en/mobile/experiences/cortana (2017), Last accessed: 2017-01-02.

[13] Carnegie Mellon University, "CMU Sphinx-4 Open Source Speech Recognition Toolkit", http://cmusphinx.sourceforge.net (2017), Last accessed: 2017-01-02.

[14] R. Bacher, "Web2cToolkit Framework for Web-based Controls Clients", http://web2ctoolkit.desy.de (2017), Last accessed: 2017-01-02.

[15] Google Inc., "Google Web Toolkit Open Source Project", http://www.gwtproject.org (2017), Last accessed: 2017-01-02.

[16] P. Duval et al., "Threefold Integrated Network Environment", http://tine.desy.de (2017), Last accessed: 2017-01-02.

[17] K. Rehlich et al., "Distributed Object Oriented Control System", http://doocs.desy.de (2017), Last accessed: 2017-01-02.

[18] The EPICS Collaboration, "Experimental Physics and Industrial Control System", http://www.aps.anl.gov/epics (2017), Last accessed: 2017-01-02.

[19] The Tango Collaboration, "Tango Controls", http://www.tango-controls.org (2017), Last accessed: 2017-01-02.

[20] T. Kosuge et al., "Simple Transmission and Retrieval System", http://stars.kek.jp (2017), Last accessed: 2017-01-02.

[21] R. Bacher, "Web2cHMI Documentation", http://web2ctoolkit.desy.de/Web2cHMI/Web2cHMI.pdf (2017), Last accessed: 2017-01-02.