

EUPHORIA: An Asynchronous Event-driven Architecture for Adaptable and Flexible Interaction across Heterogeneous Devices in Smart Environments

Ovidiu-Andrei Schipor

*MintViz Lab | MANSiD Research Center, University Stefan cel Mare of Suceava
13 Universitatii, 720229 Suceava, Romania*

Radu-Daniel Vatavu*

*MintViz Lab | MANSiD Research Center, University Stefan cel Mare of Suceava
13 Universitatii, 720229 Suceava, Romania*

Jean Vanderdonckt

*Louvain School of Management, Université catholique de Louvain
B-1348 Louvain-La-Neuve, Belgium*

Abstract

Context: From personal mobile and wearable devices to public displays, our digital ecosystem has been growing with a large variety of “smart” devices that capture, communicate, and exchange raw data with applications, thus imposing an ever-changing software architecture.

Objective: We introduce EUPHORIA, a new software architecture design and implementation that enables prototyping, deployment, and evaluation of adaptable and flexible interactions across users using heterogeneous personal and public devices in smart environments.

Method: We designed EUPHORIA by following the requirements of the ISO/IEC 25010:2011 standard on Software Quality Requirements and Evaluation, applied to the specific context of smart environments.

Results: To address adaptability and flexibility, three application scenarios demonstrate how EUPHORIA could straightforwardly implement practical prototypes for multiple users, with multiple input/output devices, in different environments, thus covering various contexts of use: (1) wearable user interfaces and whole-body gesture input for a citizen interacting with pub-

lic displays, (2) multi-device interactions in physical-digital spaces, and (3) smartwatch-based interaction between a smart connected car and its driver. We also present a technical evaluation of EUPHORIA from the perspective of the main factors responsible for the magnitude of request-response times for producing, broadcasting, and consuming messages inside the architecture. Finally, we deliver the source code of EUPHORIA that is free to download and use for research and development purposes.

Conclusion: By introducing EUPHORIA and demonstrating its three application scenarios, we hope to foster advances and developments in new software architecture initiatives for our increasingly complex smart environments, but also to readily support implementations of novel interactive systems and applications for smart environments of all kinds.

Keywords: Context-aware computing, mobile computing; multi-device interaction, smart environment; smart space, software architecture; wearable computing.

1. Introduction

The mobile and wearable computing era provides users with an ever-growing variety of "smart" devices (e.g., smart rings, smart watches, intelligent displays) that, when interconnected to capture, communicate, and exchange their raw data in a physical-digital ecology of personal and public devices and applications, acquire the ability to support a larger variety of their users' digital needs [1]. Personal mobile and wearable devices as well as public displays embed increasingly sophisticated sensors and algorithms that have practically remodeled our digital ecosystem. End users are confronted with a larger and more complex palette of contexts of use [2]: the relationships between end users, their on-body devices and off-body sensors found in public installations become more complex. Software services distributed over the web has led to the emergence of a new form of physical-digital reality, in which asynchronous data collection, processing,

*Corresponding author

Email addresses: `schipor@eed.usv.ro` (Ovidiu-Andrei Schipor),
`radu.vatavu@usm.ro` (Radu-Daniel Vatavu*), `jean.vanderdonckt@uclouvain.be`
(Jean Vanderdonckt)

URL: `http://www.eed.usv.ro/~vatavu` (Radu-Daniel Vatavu*)

Preprint submitted to Information and Software Technology

November 28, 2018

and interpretation from heterogeneous input/output devices (*e.g.* using different operating systems, communication protocols, exploiting Internet-of-Things) have become the foundation of “smart environments” [3].

Although smart environments are still in their infancy, their immense potential to enable intuitive, fluent, and flexible interaction between end users and their applications will ultimately integrate user behavior [4, 5]. Consequently, there has been a long-standing interest in the community towards designing hardware and software architectures to underpin the functionality requirements of smart environments [6, 7, 8, 9]. Among the preferable design criteria for such architectures are modularity, scalability, and asynchronicity of producing, processing, and sharing events are fundamental to effective design and efficient implementation of interactions in smart environments. Nevertheless, the perpetual advances towards more relevant standards and smarter technology, accompanied by the unprecedented diversity of smart wearables [10, 11, 12, 13, 14, 15], has fostered the need for deeper exploration of new hardware and software architecture designs. Consequently, there is still room for innovation in effective and efficient software architecture for smart environments.

In this paper, we address new software architecture designs required by smart environments to enable adaptable and flexible interaction and to accommodate a wide diversity of present and future input/output devices. Specifically, we focus on a software architecture providing asynchronous, event-driven communication between heterogeneous input/output devices in ways that are simple for the designer to prototype, the programmer to implement, and the end-user to interact with. Toward this end, we developed EUPHORIA¹ a new scalable, event-driven software architecture enabling heterogeneous and asynchronous communications between personal devices and public installations and services that constitute a smart environment. Since we planned to make this architecture as flexible and interoperable as possible [16, 17, 18, 14, 19, 20, 21], we made use of web protocols and technology, such as `http` requests and interchangeable JSON data formats that are leveraged to produce, transmit, process, and interpret device- and environment-specific events through light software interfaces. For example, exposing a custom event in Euphoria is equivalent to running a standard `http` request, while consuming events is readily achieved by subscribing to event notifications, implementable in a few lines

¹The name EUPHORIA stands for Event-based Unified Platform for HeteRogeneous and Asynchronous Interactions.



Figure 1: Our smart environment scenario showing several users employing a variety of personal input/output devices and public installations. EUPHORIA enables simple integration of heterogeneous devices via a scalable, event-driven architecture.

of `JavaScript` code. The main characteristics of Euphoria are *flexibility*, *asynchronicity*, and *heterogeneousness*, which foster accommodation of a wide range of current and future input and output devices. We release the source code of Euphoria for free for research purposes at the permanent web address <http://www.eed.usv.ro/mintviz/resources/Euphoria>, together with code examples and description of application scenarios. Our contributions in this paper are as follows:

1. We introduce EUPHORIA, a new software architecture for designing flexible, heterogeneous, asynchronous interactions with digital content in smart environments. EUPHORIA implements device- and environment-specific event-driven production, sharing, and processing using common web protocols and technology readily accessible from any web-connected device, such as smartwatches, smartphones.
2. To demonstrate EUPHORIA’s effectiveness for practical usages in smart environments, we describe three application scenarios involving a variety of devices representative for today’s smart technology: smart rings, smartwatches, smartphones, motion tracking sensors, a multi-touch tabletop, and a large ambient wall-display; see Figure 1 for an illustration of our testing environment in which EUPHORIA was designed, implemented, and evaluated.

3. To foster and encourage further developments in the community toward software architecture for smart environments, we deliver EUPHORIA for free to researchers and practitioners as an open-source architecture, driving interactions for smart environments with event-based communications and processing.

1.1. Design and Technical Requirements for EUPHORIA

The discussion so far identified design approaches adopted by prior work in software architecture design for smart environments. EUPHORIA builds on this work, but also differentiates in a number of ways. To better understand how EUPHORIA positions in the context of the relevant literature and developments that we overview in this section, we present a comparative analysis among prior work according to a set of specific criteria, such as the software quality properties. To be able to conduct this discussion effectively, we first outline the design and technical requirements of EUPHORIA.

We designed the EUPHORIA software architecture for handling devices and software applications within a small smart environment consisting in no more than tens of consumers and producers in order to cope with scenarios such as smart rooms or in-vehicle interaction. Several handling techniques and quality properties (Figure 2) were adopted, two most salient being:

- H1. *The Event-Response System (ERS) technique* for multi-threaded dialogues [22] enables physical, hardware, and software changes that occur in a smart environment to be readily detected and integrated in the business logic of the EUPHORIA software architecture. In this technique, events can be generated by any entity ranging from the end user to any software/hardware component and handled through event listeners in a way that preserves decomposition principles [23]: high level of abstraction, high internal cohesion, and low coupling.
- H2. *Asynchronicity of data processing* [6] enables appropriate handling of unpredictable time-response behavior of entities within the smart environment.

We also adopted the following four quality properties for EUPHORIA by relating to the Software Quality Requirements and Evaluation (SQuaRE) ISO/IEC 25000 series of standards [24, 25], as follows:

- Q1. *Adaptability*, which subsumes *Scalability*, measures a system ability to adjust effectively and efficiently to changes occurring in its context of use, *e.g.*, new hardware and software modules. Adaptability represents a key characteristic of EUPHORIA, which, by definition, needs

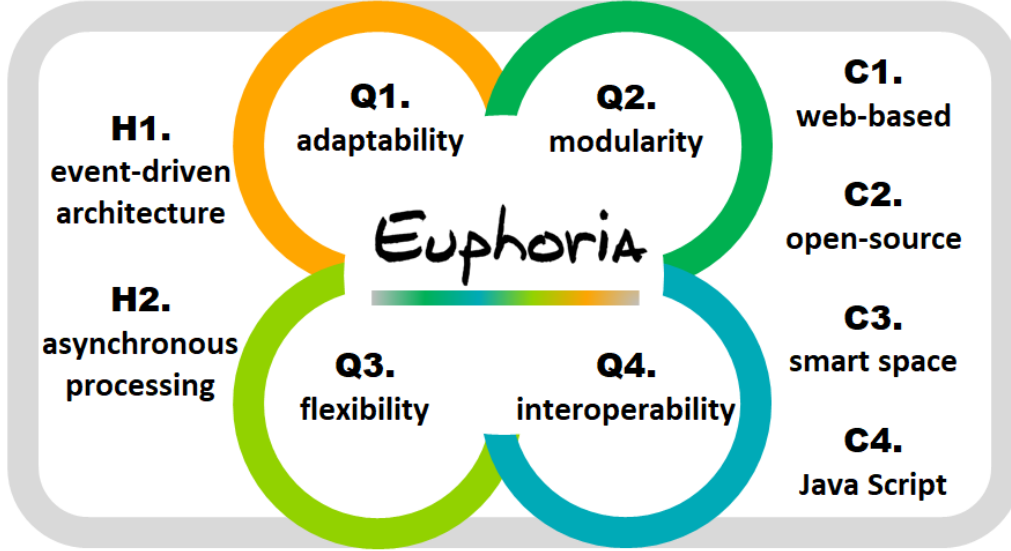


Figure 2: Visual overview of the two handling techniques (H1 and H2), the four quality properties (Q1, Q2, Q3, and Q4), and the four contextual properties (C1, C2, C3, and C4) adopted for the design of the EUPHORIA software architecture.

to accommodate various types of input/output devices for smart environments. Two levels of scalability are relevant in the case of smart environments: scalability with respect to the size of the messages exchanged and scalability with respect to the number of consumers consuming these messages.

- Q2. *Modularity* evaluates the degree to which changes occurring in one system component determine changes to occur in other components, which is important for low coupling [23]. Modularity is important for EUPHORIA to facilitate integration of diverse input/output devices.
- Q3. *Flexibility* represents the system ability to be used reliably in new contexts of use, particularly those which were not specified in the initial requirements, without any external intervention or with a limited one. EUPHORIA should be flexible to implement a variety of application scenarios for smart environments with as little as possible changes in its architecture.
- Q4. *Interoperability* refers to the degree in which two or more systems are able to work together toward a common goal, such as exchanging data. Interoperability enables a heterogeneous smart environment,

composed of different input/output devices, to operate effectively.

A set of four contextual properties was also adopted in order to shape the EUPHORIA architecture to the specific needs of designing flexible and adaptable interaction in smart environments:

- C1. **Web-based** refers to adherence to common Internet standards and protocols for data processing and exchanging, such as the JSON format for parsing messages.
- C2. **Adoption of open source standards and technology** refers to the property of relying on open source technologies that preserve public accessibility and free usage of the resulting environment.
- C3. **Smart environment oriented** refers to capability of supporting interactions between different users, their various personal devices and surrounding public displays, all in a smart environment.
- C4. **JavaScript oriented** refers to the usage of JavaScript as the default description and programming language shared by all EUPHORIA modules. Beyond this capability, any language that implements WebSockets technology can readily be used to connect to and use the features of the EUPHORIA architecture.

The next section presents a comparison between several prior software architectures for smart environments based on these design criteria, quality properties, and technical requirements.

2. Related Work

In this section, we discuss prior work in software architecture related to interactions in smart environments. We rely on this prior work to inform the design principles, approach, and technical requirements of the EUPHORIA software architecture. To this end, we start by conducting a requirement elicitation and we identify the main factors that make EUPHORIA original and unique with respect to prior work. We also rely on our discussion of prior work to introduce the conceptual framework of our design approach by highlighting relevant definitions and engineering approaches of software architecture for smart environments. We do so by connecting to prior work from software architecture design, smart environments, and designing interactions for ecosystems of smart devices.

2.1. Event-driven Software Architecture Design

The EUPHORIA architecture implements the principles and concepts of Event-Driven Design (EDD) to support ERS [22], while also sharing other characteristics with object-oriented, client-server, and layered software architectures. To understand this design approach, we start with a brief overview of the related work in this area.

The term “software architecture” generally refers to an abstract, yet implementable, structural and functional description of a software system [26]. Due to the central role played by the software architecture in the lifetime of an application, a vast amount of literature addresses this topic: eight architectural styles accompanied by various combinations and adaptations [27] resulting into a consolidation into an comprehensive meta-model [28]. In the area of engineering interactive systems, a similar attempt has been made which resulted into the Arch/slinky metamodel, which structures the software architecture of any interactive system into five components [29]: a dialog controller ensures the sequencing of asynchronous events from and to the functional core of the application and from/to its interaction toolkit, respectively via two independence modules: a functional core adapter and a presentation adapter. Acting from the functional core, the architecture directly determines the general behavior, limits, and advantages of the final software product. To this end, a software architecture usually consists of several components that are connected through data flows [30].

A first attempt to introduce event-driven software architectures (EDA) belongs to [31], who were the first to coin the concept of the transaction center as a software component that is *“able to get (obtain or respond to) transactions in raw form, analyze each transaction to determine its type, dispatch on type of transaction, [and] complete the processing of each transaction”* (p. 225). EDA architectures consist of four layers: producers, emitters, an engine, and consumers; see [32]. A formal distinction is made between the concept of an “event,” a change in state that occurs at a precise moment in time at the producer level, and that of a “message,” an event notification that travels from the emitter to the consumer [33]. By construction, an EDA is well suited for implementing applications supporting low coupling [23] between components that communicate with each other asynchronously [34, 35].

The EDA approach has been demonstrated suitable for a wide range of applications, and especially for highly interactive systems [36], from networked sensors [37, 38] and smart grids [39, 40] to handling business process

management systems [41, 42] and running operations in smart cities [43, 44]. Three-dimensional visual reconstruction using trains of events with very high temporal resolution [45], simulations of spiking neural networks [46], and integration of smart multi-agent [47] are among other recent applications for which EDA played an important role. All these applications rely on the same set of design requirements, including responsiveness, asynchronicity of events, and heterogeneousness of data sources.

2.2. Software Architectures for Smart Environments

Previous work provided various definitions of what a smart environment is from the perspective of creating visions for ubiquitous computing [48, 1] and ambient intelligence [49], but also for providing practical guidelines for designing the user experience in such environments [50]. In this work, we rely on the definition of [51], according to which a smart environment is *“a small world where all kinds of smart devices are continuously working to make inhabitants’ lives more comfortable”* (p. 3). Particularly, we address the aspect of implementing smart environments by focusing on the specific part referring to *“all kinds of smart devices”* and their interoperability and ability to communicate seamlessly. Furthermore, we connect to the variety and richness of devices from the vision described by [48] of a *“physical world richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives and connected through a continuous network”* (p. 694). To this end, EUPHORIA addresses the practical aspects of implementing applications for smart environments by integrating the functional and operational diversity of input/output devices into flexible, adaptable interactions, while respecting asynchronicity and supporting device heterogeneousness.

Special attention has been devoted to EDAs for the conceptual foundation of smart environments [43, 52, 19]. EDAs are suitable for such applications since smart environments consist of numerous heterogeneous sensors and devices interacting with each other in ways that are not always deterministic. From wearable sensors and gadgets, such as smart rings [11], finger augmentation devices [53], and armbands [54] that enable always-available input to smartwatches [16, 15], health-monitoring smart belts [55], and smart clothing [56], there is a wide range of mobile, wearable, and embedded technology to integrate smart environments. Moreover, these devices need to communicate with complex installations, such as systems that track people’s location and movement inside the environment [57], producers of multimedia content [58], or very large systems implementing

crowd-computer interaction [59] and mass-computer interaction [60]. Low-level parameters provided by devices and applications within a smart environment can be further used to access high-level, contextual information on behalf of context-aware software [61, 62]. For example, reliable information about the health or the emotional state of a user usually emerges from more than one sensor. Multiple variables need to be measured across various channels, such as physiological data (*e.g.*, blood pressure, skin conductance, respiratory rhythm), audio (*e.g.*, the tone or the pitch of the user’s voice), and video (*e.g.*, the face expression, body posture, or body gestures); see [63], or [?] for example. Collecting, processing, integrating, and interpreting such complex data in real-time in a unified manner becomes critical to realize the transition from raw data to information and knowledge. As EDA can handle heterogeneous and asynchronous data collection and processing, they represent an appropriate choice for implementing the software components of a smart environment.

There has been a significant trend in the community to provide high-level information, such as emotion [63, 64], activity [65, 66], or gestures [9, 67] through abstract software components in smart environments. For instance, the Gesture Profile for Web Services architecture (GPWS) of [8] is an event-driven architecture that supports development of gesture interfaces for smart environments by integrating third-party gesture recognition services. Such an approach, specific to software-oriented architecture design, hides from the developer all the complex pattern recognition techniques required to detect and recognize gesture input and allows developers to focus on prototyping interactions. More recently, [9] developed Gesture Services for Cyber-Physical Environments (GS-CPE) by extending the GPWS architecture of [8]. GS-CPE is a novel, event-driven, service-oriented framework that implements personalized gesture recognition and user identification. [7] introduced a software architecture to implement interactions with spatially-indexed media in smart environments that instantiate mappings between digital content and specific regions of the physical space. The event-driven architecture design approach has also been applied for real-time activity recognition within a smart home [65] and for collaborative environments [21]. EDA flexibility enables readily integration of data processing and interpretation algorithms into the functional core of applications, such as specialized detectors to assess the similarity of activity events or complex machine learning approaches, such as Support-Vector Machines (SVM), for recognizing users’ activity patterns in real-time [65].

2.3. Current Software Architectures for Designing Interactions in Smart Environments

Interacting in smart environments poses many challenges. While it is simple to immerse users and their devices in the same environment, achieving straightforward and demanding interactions remains challenging. While people can interact with each other through speech and body language, devices need more formal contexts and structures for communication. For this reason, prior work has explored natural language interpretation [68], gestures and whole-body movements [67], emotion recognition [64, 69], or the interpretation of electrical brain signals [70].

The “nomadic gestures” [71] introduced a shift of perspective in terms of the reusability of gesture commands across ambient interactive systems, which can download users’ gesture preferences on the fly and adapt their internal gesture set accordingly, thus representing a first step towards transferring gestures across devices for a same user. “Smart-Pockets” [72] is another attempt to make interactions between users and ambient interactive systems simple and intuitive. With Smart-Pockets and their extension in the form of “Smart-Containers,” users can access their personal digital content by pointing to various pocket locations on their body or accessories [72]. The short access time (similar in magnitude to the production time of touch gestures) and the high accuracy rates for recognizing Smart-Pockets actions make this technique viable for retrieving digital contents.

The continuous miniaturization of computing devices has created the need to investigate new interaction modalities. For example, [73] employed a touch-sensitive wristband to enable text entry on a smartwatch. The authors showed that a multi-tap keyboard layout superimposed on the wristband was appropriate for entering short text. Hands-free interaction with small wearables was investigated as well [74]. Gaze interaction for smartwatches by tracking eye movements [17] has also proved to be a robust and highly accurate technique. In addition, smartwatches have been used to assist interactions with smartphones. For instance, DUET coordinates motion and touch input of both devices in order to extend their visual and tactile output [16]. However, despite innovative solutions and promising results, such previous attempts have focused on immediate functionality, leaving room for many improvements in terms of the interoperability, standardization, and scalability of the interaction.

Some smart environments need to operate a large number of sensors and devices and, thus, networks of wireless sensors have been proposed [75]. For

instance, the Mobile Agent Platform for Sun SPOT of [76] implemented mobile agents to enable flexible programming of interaction rules within an ecosystem of sensors. The Vi-SMARt system [21] showed that the combination of smart environments and multi-agent systems was an appropriate solution for integrating various combinations of sensors. The open-source platform of [77] and the BodyCloud software system of [14] focused on sensor networks located on the user’s body. BMF-WSANs, a Building Management Framework for Wireless Sensor and Actuator Networks [78], is another platform that allows rapid development and flexible management of monitoring applications in smart environments. A step forward was made by offering an entire “ready to use” platform as starting point for complex smart infrastructure development. The Future Internet Lab (FIWARE Lab) represents a management solution for cloud federation whose main goal is to keep configuration settings at minimum possible [79]. Several federation management tools such as Pegasus (Platform as a Service Component), DCA (Deployment and Configuration Adapter), and Monitoring Middleware work together in order to minimize the development and deployment effort. However, this solution targets an area beyond EUPHORIA’s goal since it contains up to 14 running nodes on four continents [80].

A promising approach to design novel interactions in smart environments relies on the ability of smart devices to sense their surroundings, such as to enable “proxemic interaction,” a concept first employed to address inter-human communication [81]. For example, a hand-held device can exploit spatial relationships, *e.g.*, orientation and distance, between itself and its users, other devices, and even uninstrumented physical objects from its vicinity in order to establish a communication context [82]. This context further enables a dynamic form of proxemic-aware knowledge and control in relation to the surrounding entities [83]. A smart environment built on top of an EDA architecture can leverage the advantages of proxemic awareness since it is able to detect and integrate state changes in a unified manner [20]. The spatial characteristics of entities that interact in a smart environment can be measured using various techniques. For example, localization of users can be achieved with video cameras, such as the Microsoft Kinect sensor. The XDKinect framework [20] is an example of an architecture that facilitates development of cross-device applications by delivering a time-based application programming interface (API) for whole-body gesture interaction. XDKinect also features a configurable, multi-modal software module to capture gesture and speech commands. [84]

	Adapt. (Q1)	Modul. (Q2)	Flex. (Q3)	Inter. (Q4)	Web (C1)	OS (C2)	Smart. (C3)	JS (C4)
UbiComp Middleware [75]	●	◐	◑	●	◐	◑	●	◐
MAPS [76]	◐	●	◑	●	◐	◑	◐	●
SPINE [77]	◐	◑	◑	●	◐	●	◐	●
Proximity Toolkit [82]	◐	●	◑	●	◑	●	●	◑
BMF-WSANs [78]	●	◑	◑	●	◐	◑	◑	◐
BodyCloud [14]	◐	◐	◑	◐	◑	◑	◑	◑
XD-Kinect [20]	◐	●	◐	◑	●	◑	◐	◐
FiWare [79]	●	●	◐	◑	●	●	◑	◐
Proxemic [81]	●	●	◐	●	◑	◐	◑	◐
Vi-Smart [21]	●	●	◑	◑	●	◐	●	◑
Mobile@Old [85]	◐	◐	◐	◑	◐	◐	◑	◐
GS-GPE [9]	◐	●	◐	◑	●	◑	●	◑
Average	◑	◑	◐	◑	◑	◑	◑	◐

Table 1: Comparison table of related work in chronological order of appearance.

employed the “Proximity Toolkit” of [82] and a Vicon motion tracking system to design remote controls for ubiquitous computing ecologies. Their work investigated several interactive scenarios and showed the reliability of such a framework for interacting with appliances and devices from a smart environment. The Microsoft Kinect systems was also used with wearable sensors in the Mobile@Old platform [85] to assist elderly people to maintain a healthy lifestyle in the comfort of their homes.

A comparison among related platforms and software architectures for smart environments discussed up to this point is presented in Table 1. For this comparison, we used the four qualitative properties (Q1 to Q4) and the four contextual properties (C1 to C4) as differentiating factors to highlight the unique features of EUPHORIA. Each entry is depicted Table 1 based on Harvey’s balls as follows: a ball with a single quarter denotes that the criteria is qualitatively fulfilled by at least one feature, a half ball denotes that the criterion is qualitatively fulfilled by more than one feature, a three-quarter ball denotes that the criterion is qualitatively fulfilled by many features which can be combined, a full ball denotes a total fulfillment. Based on our estimation, we also compute an average score for each criterion in order to overall characterize the investigated previous work. This comparative analysis was performed based only on data available in the accessible literature. No quantitative measurement was actually performed in order

to obtain a similar analysis because this would have required the actual usage and benchmarking of all these environments, which are themselves heterogeneous and inconsistent with each other. In the next section, we describe how these features were implemented in the software architecture of EUPHORIA.

3. Euphoria Software Architecture

We present the blueprint of the EUPHORIA software architecture by focusing on details that are relevant for smart environments, such as our own environment depicted in Figure 1. The core design approach of EUPHORIA revolves around the production, transmission, detection, and consumption of device- and environment-specific events, which we define as significant changes in the state of a hardware or software module that is part of the smart environment [33].

Figure 3 illustrates the architecture design adopted for EUPHORIA with its main components highlighted: PRODUCERS, EMITTERS, PROCESSING-ENGINE, RECEIVERS, and CONSUMERS, according to the model of event-driven architectures [32]. We designed EUPHORIA with separation of concerns between its functional blocks (quality property Q2) and the thickness of its adaptation layers (Q1). Also, following the principles of event-driven (H_1) software architecture design [31, 33, 32], the information about events triggered by PRODUCERS are collected, packed into messages, and exposed by EMITTERS to the next layer. Afterwards, messages are processed and dispatched by the PROCESSING-ENGINE to third-party RECEIVERS AND CONSUMERS, who will interpret the messages according to their inner business logic and take appropriate action in the smart environment, *e.g.*, an event detected by a smartwatch worn by the user on their wrist will result in an item being selected on the large display in front of the user; see Figure 3. A message transmitted through EUPHORIA consists of a *header* with meta-data (*e.g.*, the device name, device address, and the name of the event), and a message *body* that contains the relevant data for the event. Figure 4 illustrates four examples of messages of various complexities produced by a wearable armband, a smartphone, and two environmental sensors.

The PRODUCER is the component for which a relevant change in state is notified to the architecture in the form of an *event*. Producers are instantiated by input devices (*e.g.*, motion sensors from the environment, smart mobile devices and wearables, etc.) or by software services (*e.g.*, a particular gesture command has been detected for a particular user by a gesture

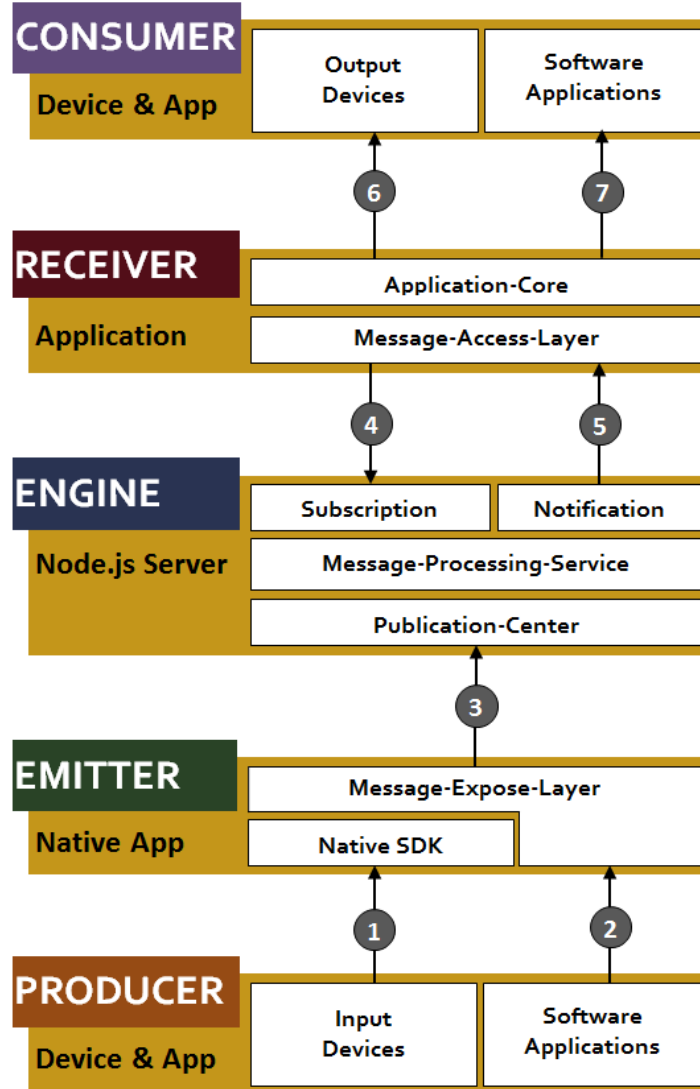


Figure 3: The layered architecture of EUPHORIA, an event-based software architecture for designing interactions in smart environments. EUPHORIA consists of event producers, emitters, a processing engine, and event receivers and consumers. Numbers ❶ to ❷ represent dataflows that are discussed in the text.

recognition service from the web). For PRODUCERS that are instantiated by input devices, events refer to any physical change in their states that are relevant to the environment, such as the press of a button or a change in the orientation of the wrist for a smartwatch. Software events can also

<pre> message{2} ▾ header{3} ▶ deviceName :MYO Armband #1 deviceIP :192.168.0.9 eventName :Gesture timestamp :1493376106270 body {3} ▶ gesture :TwistRight acceleration{3} ▶ x : 1.221 y : 0.613 z : -0.121 orientation {3} ▶ azimuth : 0.196 pitch : -0.184 roll : 1.254 </pre>	<pre> message{2} ▾ header{3} ▶ deviceName :Smart Phone #3 deviceIP :192.168.0.5 eventName :Touch-Select timestamp :1493376183594 body {4} ▶ touchTime :267 touchPressure :0.62 touchLocation {2} ▶ x: 273 y: 361 deviceLocation{3} ▶ x: 2425 y: 578 z: 1587 </pre>
<pre> message{2} ▾ header{3} ▶ deviceName :Vicon #1 deviceIP :192.168.0.7 eventName :Right-Hand timestamp :1493376183681 body {3} ▶ modelInfo{3} ▾ name :Right-Hand pointingMarker :A orientationSegment:ABCD markers {4} ▾ A{1} ▶ location {3}▶... B{1} ▶ ... C{1} ▶ ... D{1} ▶ ... segments {1} ▾ ABCD{2} ▶ markers: [A,B,C,D] ▶ orientation{3}▶... </pre>	<pre> message{2} ▾ header{3} ▶ deviceName :Kinect #1 deviceIP :192.168.0.6 eventName :Bob-Skeleton timestamp :1493376182723 body {3} ▶ joints{25} ▾ SpineBase {1}▶location {3} ▶... SpineMid {1}▶location {3} ▶... Neck {1}▶location {3} ▶... ... ThumbRight{1}▶location {3} ▶... bones {14} ▾ ArmUpLeft {2}▶joints :[...] orientation{3}▶... ... FootRight {2}▶joints :[...] orientation{3}▶... </pre>

Figure 4: Examples of JSON messages of various complexity: a message containing acceleration and orientation data collected from a Myo armband (top left); touch input data reported by a smartphone (top right); motion data tracked by a Vicon system (bottom left); and a message containing a Microsoft Kinect skeleton (bottom right).

be produced in relation to changes in the condition of some software application. Software events prove useful for debug phases, running simulations for the environment, or for integrating complex software logic provided by third-party libraries and services, such as those available on the web. Characteristic to event-driven architectures, one feature of EUPHORIA is that distinct layers are unaware of the inner working details of the other layers of the architecture (Q2). For example, it is not important for PRODUCERS to know which components, if any, will process and consume the events they produce. This feature facilitates seamless integration of new devices and software applications into EUPHORIA, including those yet to appear or be developed in the future, such as devices for mobile augmented reality [86], without any modification in EUPHORIA’s main core.

Once an event has been produced, the event triggers the instantiation of a message (see dataflows ❶ and ❷ in Figure 3), which travels to the next levels of the architecture. In the case the event was produced by an input device, the EMITTER is built around that device’s software develop-

ment kit (SDK) that delivers the required software components to collect the specific parameters of the event (*e.g.*, the properties of a touch on a touchscreen, the orientation of a motion-sensing device, etc.; see the examples illustrated in Figure 4). The **EMITTER** collects event data and packs them in the form of messages which are delivered to the higher layers of the architecture. A message consists of a header with metadata (*i.e.*, the event type and the identification of the producer) and a message body with details about the change of state (*e.g.*, the button that was pressed, new orientation data of the device, the type of gesture performed by the user, new position coordinates from a motion sensor, etc.). While the header of the message has a fixed structure, its body depends on the type of device and the event that was produced. In most cases, the **Message-Expose-Layer** (see Figure 3) needs to be implemented in native code, although some SDKs or software applications may already offer higher-level interfacing options for a given input device. The **EMITTER** and **RECEIVER** are the interface between the **EUPHORIA** core and the variety of devices and software application we want to connect each other. In order to expose or receive a message there is a need for a software conversion between a **PRODUCER** or **CONSUMER** centric model to a generic, **EUPHORIA** centric model. The utilization of well established and optimized technologies such as **http** or **JSON** leads to an adaptation layer as thin as possible while conserving flexibility and interoperability. In fact, the **EMITTER** and **RECEIVER** are the only levels of architecture that have to be implemented by developers. Since the **EMITTER** component is required for every input device to be integrated in applications that will run in the smart environment, we designed it as general and reusable as possible (according to quality properties Q_3 and Q_4). In a raw form, the **EMITTER** only consists in the code necessary for encapsulate specific properties in a standardized **JSON** representation and for sending this message through a common **http** request. Feeding our architecture with an event is as simple as requesting the delivery of contents from a specific Uniform Resource Locator (URL): the query string represents the event message.

The **PROCESSING-ENGINE** is the core component of **EUPHORIA**. Its main responsibility is to receive messages from **EMITTERS** (see dataflow ③ in Figure 3) and to dispatch them to the appropriate **CONSUMERS** that have registered to receive those specific types of events. In our implementation, the **PROCESSING-ENGINE** is run by a **node.js** web server². The event

²<https://nodejs.org/en/>

messages collected by the Publication-Center are unpacked by the Message-Processing-Service; see Figure 3. Before receiving any notification, a new CONSUMER must register with the PROCESSING-ENGINE by following the protocol of a Subscription module. As part of the subscription protocol, the CONSUMER will specify the type of events it requires and the corresponding PRODUCERS. Once registration is completed, the PROCESSING-ENGINE will automatically notify the CONSUMER about those events that meet the requirements agreed during subscription. By performing both registration and notification, the PROCESSING-ENGINE and the CONSUMER rely on **WebSockets** mechanisms, a set of communication protocols that enable full-duplex, TCP-based sessions between a client and a server [87, 88]. The Message-Processing-Service (Figure 3) routes the messages to the appropriate CONSUMERS and also converts the data in the JSON (JavaScript Object Notation) interchangeable format (Q4). The PROCESSING-ENGINE is completely unaware about the internal structure of producers and consumers. This abstraction supports both scalability (Q1) and the ability to integrate heterogeneous input devices (Q4).

The RECEIVER is any third-party software application that requests events from the smart environment to run its functional core. The RECEIVER establishes a **WebSocket** connection with the PROCESSING-ENGINE using the Message-Access-Layer (see dataflows ④ and ⑤ in Figure 3). In addition to the filter list of the types of events and associated PRODUCERS, the RECEIVER can also specify other optional parameters, such as a minimum time interval between receiving two consecutive events of the same type to prevent intractable accumulations of events to handle at once. This practice also serves to prevent the overloading of the network with unnecessary or redundant traffic. One important goal is to maintain a very thin layer (low coupling) between a RECEIVER and the PROCESSING-ENGINE so that any existing software application can integrate with EUPHORIA with a minimum coding effort (Q1 and Q2). To this end, we decided to implement the Message-Access-Layer using the industry-mature technology of **WebSockets**, enabling thus real-time and asynchronous data transfer (H1) between the PROCESSING-ENGINE and the RECEIVER. Once received, the message triggers a specific action on the CONSUMER which is the fifth layer of the architecture (see dataflows ⑥ and ⑦ in Figure 3). This tier consists in output devices and software applications that users interact with directly. It is the output interface between Euphoria and the outer environment. The implementation details of CONSUMERS are locked within the RECEIVER

layer to make the ENGINE as decoupled as possible.

4. Technical Evaluation of Euphoria

4.1. Experiment

We conducted an experiment to evaluate the technical performance of EUPHORIA by quantifying the effect of several key factors (message size, device type, and the complexity of the smart environment) on the real-time performance of EUPHORIA for creating, transmitting, and processing events. Besides reporting results on technical performance, we also use our measurements to estimate the upper bound of the workload that EUPHORIA can effectively process under the constraints of real-time operation, which we defined as instantaneously-perceived response.

4.1.1. Experiment Design

We are primarily interested in the temporal performance of event processing in EUPHORIA, which we evaluated using the REQUEST-RESPONSE-TIME dependent variable defined as the average time, expressed in milliseconds, required for an event to be processed by the EUPHORIA architecture, from its creation by a PRODUCER until its consumption by a CONSUMER. Our technical evaluation experiment was a within-subject design with the following three independent variables:

1. MESSAGE-SIZE, interval variable with five experimental conditions representing message sizes ranging from 64 bytes to 16 Kbytes in a geometric progression with common ratio 4, *i.e.*, 64, 256, 1024, 4096, and 16,384 bytes, respectively. MESSAGE-SIZE represents the amount of information contained by an event created, transmitted, and processed by EUPHORIA; see Figure 4 from the previous section for examples of messages. Our null hypotheses for the MESSAGE-SIZE are that the size of message does not influence the request-response time and, consequently, that there is no relationship between the two variables. In addition, the alternative hypotheses are that larger messages will require more time to process and that the variation will follow a linear dependency:

H₁₀: The message size does not influence the request response time.

H₁₁: Larger event messages will be processed in EUPHORIA with longer request-response times.

H₂₀: There is no relation between size of an event message and the time required to process the event.

H₂₁: A linear dependency exists between the size of an event message and the time required to process the event.

2. DEVICE-TYPE, ordinal variable with three conditions representing devices with *low-end*, *midrange*, and *high-end* processing capability; see the “Apparatus” subsection next for details. We control the DEVICE-TYPE variable in order to understand the effect of computing capabilities on REQUEST-RESPONSE-TIME, with the following hypotheses:

H₃₀: There is no relation between the resources available on a PRODUCER or CONSUMER and the request-response times.

H₃₁: The more processing resources available on a PRODUCER or CONSUMER, the shorter the request-response times.

3. ENVIRONMENT-COMPLEXITY, interval variable, representing the number of CONSUMERS running requests and receiving responses in EUPHORIA at the same time. To understand thoroughly the implications of the number of consumers on the real-time performance of EUPHORIA, we varied the complexity of the environment from 1 (one consumer) to 101, by following an arithmetic progression with the common difference 10, *i.e.*, 1, 11, 21, ..., 91, and 101 consumers. Our hypotheses for ENVIRONMENT-COMPLEXITY are:

H₄₀: The number of consumers simultaneous connected to EUPHORIA does not influence the request-response times.

H₄₁: Environments with a larger number of consumers will cause longer request-response times in EUPHORIA.

H₅₀: There is no relation between the number of consumers and the request-response times.

H₅₁: The number of consumers from the smart environment and the average response time follow a linear relationship.

Both MESSAGE-SIZE and ENVIRONMENT-COMPLEXITY variables are directly linked with the quality property of scalability. The DEVICE-TYPE variable measures the quality property of flexibility since it address the capability to handle various hardware and software devices.

4.1.2. Methodology

We varied the SIZE and content of each message event and randomized the order of DEVICE-TYPES and the COMPLEXITY of the environment. As a default benchmarking scenario, every message generated by any PRODUCER was broadcasted by the EUPHORIA architecture to all CONSUMERS, which enabled us to measure time responses in conditions of maximum network load. We found that the results were slightly different on different repetitions and that is why we choose to average the obtained values over multiple runs. We empirically ascertained that the results remains the same in the case of batches larger than 1000 iteration. Thus, we ran 1000 repetitions for each combination of MESSAGE-SIZE $\times$ DEVICE-TYPE $\times$ ENVIRONMENT-COMPLEXITY, resulting in $5 \times 3 \times 11 \times 1,000 = 165,000$ experimental trials.

4.1.3. Apparatus

The ENGINE of EUPHORIA ran on a 64-bit Intel Core i7-4510U CPU with 2.60 GHz, 8 GB RAM, and Windows 7. PRODUCERS, EMITTERS, RECEIVERS, CONSUMERS, and simulation modules were implemented on the following mobile devices, representing the three conditions of the DEVICE-TYPE independent variable:

1. Samsung Galaxy S III, a smartphone with 32-bit dual-core Cortex A9 CPU @1GHz and 1GB RAM, running Android 4.1 (our *low-end* CPU condition). 1GHz CPUs are now common on many wearables, such as smartwatches (*e.g.*, Samsung’s Gear Fit 2), while the A9 ARM CPU has been superseded by more advanced architectures; see next.
2. Samsung Galaxy Tab 4, a tablet with 64-bit quad-core Cortex A53 CPU @1.4GHz, 1.5GB RAM, running Android 4.4 (our *midrange* condition). Compared to the A9 architecture, A53 has more CPU cores, higher CPU frequency, support for 64-bit applications, hardware-assisted virtualisation, dynamic frequency scaling, and an enhanced Floating-Point Unit. More RAM memory and an improved operating system imply better performance.
3. Acer Aspire V15 Nitro, a notebook with 64-bit quad-core i7-4710 CPU @2.50GHz, 12 GB RAM, running Windows 10. Compared to A9 and A53, the i7 CPU architecture relies on an advanced tri-gate transistor technology that allows low power consumption and high processing speeds. It also offers hyper-threading technology and dedicated

instruction sets, such as FMA (Fused Multiply-Add) and AVX (Advanced Vector Extensions).

We have to mention that we chose these devices as being representative for three generic categories: low-end, midrange and high-end. As of the year 2018, we believe they reflect the computing power segments the end users have access to and that the obtained results are representative for entire computing device market. The Euphoria machine and the mobile devices were connected through a 300 Mbps Wireless N Gigabit router.

4.1.4. Threats to Validity

There are several factors that may affect the validity of the experiment findings. We discuss in this section the main threats to this validity.

Regarding the *construct validity*, the three variables have been tested to their maximum potential values observed in reality: MESSAGE-SIZE has a maximum of 16,384 bytes, which covers the vast majority of messages exchanged, although messages containing streamlined big data, such as videos, could exceed this capability; DEVICE-TYPE covers three kinds of devices from the most constrained to the most powerful; ENVIRONMENT-COMPLEXITY is bounded by 101 consumers: a significantly larger amount of concurrent consumers could probably threaten this part, but the Request-Response-Time follows a linear regression function of this variable (Fig. 6). So, largely increasing the amount of consumer will not exceed a linear increasing of the response time.

Regarding the *internal validity*, we tested EUPHORIA in 165 distinct configurations, one for each combination of MESSAGE-SIZE $\times$ DEVICE-TYPE $\times$ ENVIRONMENT-COMPLEXITY. In order to avoid the *History* and *Maturation* threats, we blocked the automatic updates on all three devices so that all the trials are based on the same software configuration. Moreover, we randomized these combinations with 1,000 repetitions. We could precisely measure the succession of events in time so as to avoid the *Ambiguous temporal precedence* threat. Although we wanted to be as objective as possible, our experiment may be subject to the *Selection bias* threat in the case of DEVICE-TYPE variable: no “generic device” exists since all devices differ to some extent with other, even similar ones. The same software and hardware configuration for EUPHORIA was kept throughout the experiment for avoiding the *Instrumentality* threat. Moreover, a relative big number of repetitions was elected in order to compensate some unpredictable influences.

In order to strengthen the *External validity* of our experiment we presented all the required information for a good *Replication*, including the software our architecture was build on. However, we did not tested our architecture against the *Generalizability* except those 165 configurations. Future research must be conducted in order to extend the results to a wider variety of configurations. Although we did not repeat the data collection procedure to test the *conclusion validity*, this can be done in a very short amount of time since all configurations can be replayed.

4.2. Results

Figure 5 illustrates the effect of MESSAGE-SIZE on the REQUEST-RESPONSE-TIME performance of EUPHORIA for each condition of the DEVICE-TYPE independent variable. On average, REQUEST-RESPONSE-TIMES varied between 11 ms (for messages of 64 bytes received by consumers running on the high-end device) and 141 ms (for messages of 16 KBytes broadcasted to consumers running on the low-end device). These results allow us to reject hypothesis H_{10} and to accept H_{1A} : larger event messages are processed with longer request-response times. The effect of COMPLEXITY on REQUEST-RESPONSE-TIME is illustrated in Figure 6 for each DEVICE-TYPE. On average, response times varied from 9 ms (for the condition with one consumer running on the high-end device) and 113 ms (when messages were broadcasted to 101 consumers running on the low-end device), with an average response time of 47.4 ms ($SD = 45.5$ ms). These outcomes determine us to reject hypothesis H_{40} and to accept hypothesis H_{41} . The trends from Figures 6 and 5 also determine us to reject hypothesis H_{30} and to accept H_{31} : more powerful and resourceful devices process messages faster.

Our analysis also suggest a linear relationship between REQUEST-RESPONSE-TIME (denoted with T in the equations below) and MESSAGE-SIZE (denoted with MS and ENVIRONMENT-COMPLEXITY (denoted with EC), as follows:

$$\begin{aligned}
\text{low-end:} \quad T &= 0.005 \cdot MS + 0.749 \cdot EC + 14.374 \\
&\quad (R^2 = .877, F = 185.997, p < .001) \\
\text{midrange:} \quad T &= 0.004 \cdot MS + 0.494 \cdot EC + 3.376 \\
&\quad (R^2 = .855, F = 152.844, p < .001) \\
\text{high-end:} \quad T &= 0.001 \cdot MS + 0.152 \cdot EC + 3.862 \\
&\quad (R^2 = .949, F = 482.574, p < .001)
\end{aligned}$$

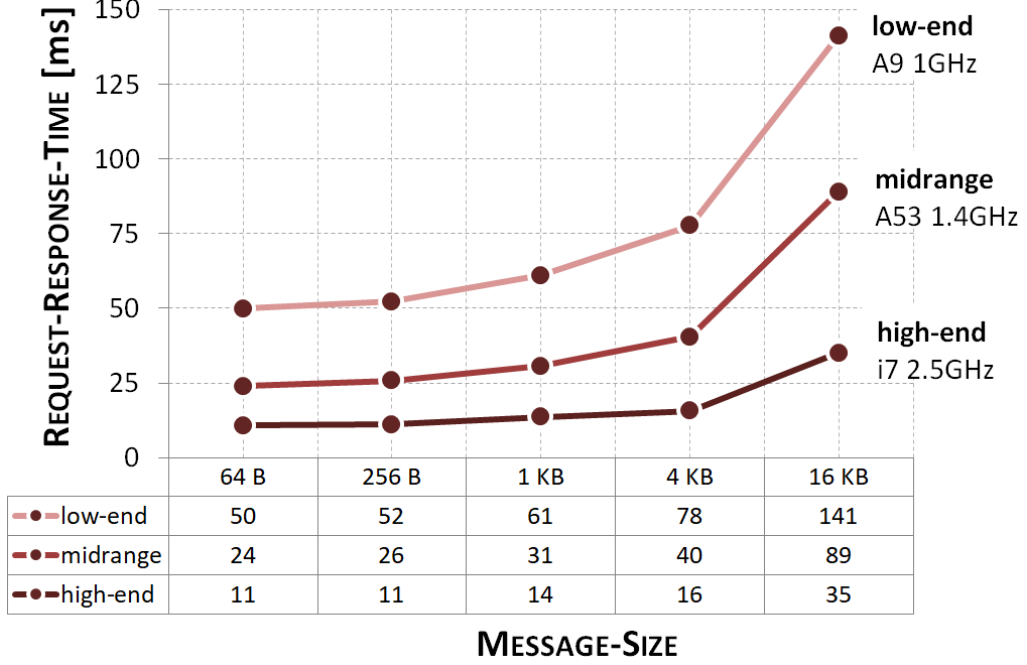


Figure 5: Average REQUEST-RESPONSE-TIMES, expressed in milliseconds, function of MESSAGE-SIZE and DEVICE-TYPE. *Note:* the horizontal scale is logarithmic with base 4.

These findings allow us to reject hypotheses $H2_0$ and $H5_0$ and to accept the alternative hypotheses $H2_A$ and $H5_A$.

Our empirical results show that EUPHORIA can process messages with a temporal performance that can be perceived as instantaneous response by users: under 150 ms average request-response time for the slowest device (A9 CPU) that processed the largest messages (64 Kbytes) in the highest density condition (101 consumers) evaluated in our simulations. However, EUPHORIA was not tested against critical, real time requirements because it is not intended to behave as a safety-critical system. The main objective was to prove it is possible to make heterogeneous devices to communicate each other in a simple and efficient way with a minimum effort of coding and configuration. The comparative analysis between EUPHORIA and previous software architectures (Table 2) suggests that this solution is more compliant with the aforementioned requirements. The quality property Q1 (adaptability-scalability) is directly linked with both number and types of devices and with different sizes for the message body. The mod-

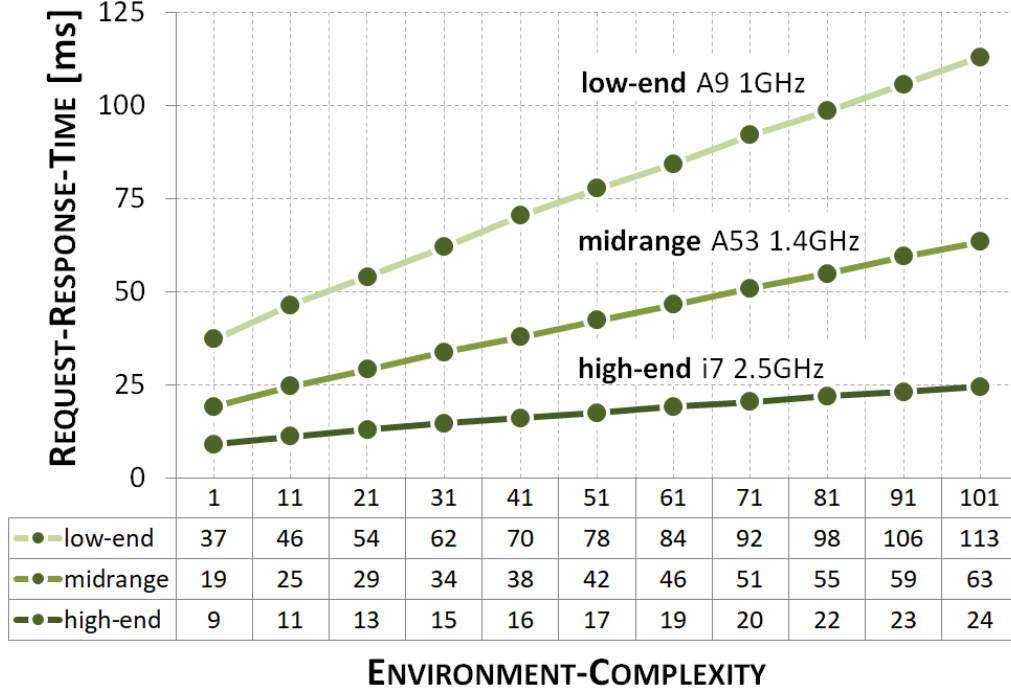


Figure 6: Average REQUEST-RESPONSE-TIMES, expressed in milliseconds, function of the ENVIRONMENT-COMPLEXITY and measured on various DEVICE-TYPES.

ularity (Q2) naturally emerges from the layered architecture and from the loose coupling communication (JSON messages sent through `http` requests and `WebSockets`). Moreover, once the `EMITTER` or / and the `RECEIVER` are implemented for one particular device they can be almost instantly used (Q3) in the case of similar devices (e.g. having the same operating system and messages structure). Due to common standardized way of communication, totally different devices can interact each other (Q4) as long as they “understand” the basic web languages.

5. Application Scenarios

We discuss in this section several application scenarios for EUPHORIA, highlighting its capability for prototyping and implementing interactions in smart environments of various kinds: (1) wearable user interfaces and whole-body gesture input for interacting with public ambient displays, (2) multi-device interactions in physical-digital spaces, and (3) delivering notifications in smart, connected cars. These three scenarios have been selected for

	Adapt. (Q1)	Modul. (Q2)	Flex. (Q3)	Inter. (Q4)	Web (C1)	OS (C2)	Smart. (C3)	JS (C4)
Average								
Euphoria								

Table 2: Final comparison between EUPHORIA and average score from the comparative analysis.

their significant coverage of various input/output devices (Fig. 7, which can be classified according to Buxton’s taxonomy of devices [89]. This taxonomy is structured along the horizontal axis according to the number of dimensions supported by the device (ranging from 1 to 3) and along the vertical axis according to the physical properties sensed (position, motion, and pressure). Each row is further divided into two sub-classes: *mechanical* when the device possesses a mechanical intermediary system between the user and the system (typically on-body) or *sensing* when the devices senses user’s properties (typically off-body). Each column is further divided into sub-classes depending on the level of sophistication of the device.

		Number of dimensions							
		1		2		3			
Property sensed	Position	Rotary Pot	Sliding Pot	Tablet	Tablet & stylus	Light Pen	Floating Joystick	3D Joystick	Mechanical
				Touch Tablet		Wall display 	Tabletop 	Smartwatch 	
	Motion	Continuous Rotary Pot	Treadmill	Mouse			Trackball	 Ring Zero	Mechanical
							Smartphone 	Microsoft Kinect  Vicon Motion	
	Pressure	Torque Sensor					Isometric Joystick		Sensing

Figure 7: Coverage of devices by the three application scenarios.

5.1. Smart-Pockets: Fast Access to Personal Data for Ambient Interactions

Smart-Pockets represent a set of whole-body gestures that enables users to access their personal digital content efficiently for visualization on public

ambient displays [72]. A Smart-Pockets gesture is a movement of the hand entailing access to some pocket, such as the trousers left pocket or the right inner breast pocket, followed by a pointing movement of the hand toward the ambient display. A complementary hand movement from the display to the pocket stores the respective content back to the user’s pocket for later retrieval. [72] showed that Smart-Pockets interactions are fast, can be recognized robustly even under user-independent training, and are flexible and easily extensible to other containers, such as bags and hand-held objects, denoted as “Smart-Containers” in [72]. The pocket metaphor implemented by Smart-Pockets creates a link between physical objects and personal digital content, useful for fast access to data in smart environments. In the following, we show how the implementation of Smart-Pockets can be readily reproduced with the EUPHORIA architecture.

Following the description from [72, p. 6], users’ whole-body movements are captured with the Microsoft Kinect sensor, which reports, among others, the position and movement of the user’s body in 3-D as a sequence of skeleton data; see [90] for details. In our implementation, we opted for the “explicit segmentation” approach advocated in [72, p. 14] as the preferred detection technique for Smart-Pockets gestures, where users decide when the system should interpret their whole-body movements by signaling explicit “start” and “stop recording” commands. [72] discussed that any device that can deliver start/stop events to the system, such as through-fabric capacitive touch input, mid-air devices, or specific hand poses, would be a good candidate to segment Smart-Pockets gestures in continuous whole-body movement (p. 14). For the illustration of this specific use case, Smart-Pockets gestures are segmented with taps performed on the Ring ZERO device [91], an interactive smart ring that reports finger motion and touch events; see Figure 8, top.

Figure 8, bottom illustrates the instantiation of EUPHORIA for this specific implementation as well as examples of event messages. There are two PRODUCERS to implement for the Smart-Pockets scenario, corresponding to the two input devices (the Kinect sensor and the smart ring), and one CONSUMER, represented by the ambient display. The smart ring PRODUCER reports to EUPHORIA taps performed on the ring’s button to be interpreted as start and stop events for the purpose of gesture segmentation. The Ring ZERO device connects wirelessly to a smart device, where a custom Android application receives ring events via Bluetooth 4.0, filters those events, and sends taps to EUPHORIA. The Kinect sensor PRODUCER collects the

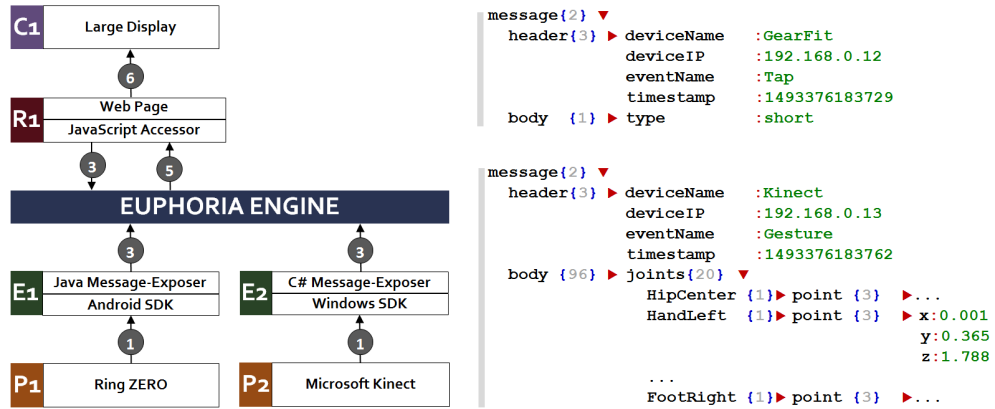


Figure 8: EUPHORIA applied to implement the Smart-Pockets technique of [72]: hand gestures captured by a Microsoft Kinect sensor and segmented using taps on a smart ring (top image) are processed as events through the EUPHORIA architecture (bottom left) implemented using JSON messages (bottom right). In this example, the user performs a Smart-Pockets gesture to visualize a YouTube video link, stored in one of their pockets, on a large ambient display.

user’s whole-body gestures and reports them to EUPHORIA as well. The CONSUMER subscribes to both taps and gesture events and implements the Smart-Pockets recognition logic: the whole-body gesture recorded between two consecutive start and stop taps on the smart ring is matched against a set of Smart-Pockets gesture templates. The recognition algorithm is described by [72] in detail, so we won’t insist on it here. If the match between the candidate gesture and any of the templates is successful, a corresponding action is executed, such as display the content associated to the specific pocket onto the large ambient display. Messages travel in EUPHORIA from the two input devices to the ENGINE thorough a native software layer implemented in Java for Android in the case of the Ring ZERO device and in C# for Windows for the Microsoft Kinect sensor. An HTML web page implements the CONSUMER represented by the public display that plays YouTube videos of links stored in user’s smart pockets.

5.2. Access to Multimedia Digital Content Anchored to Physical Locations in a Smart Environment

New physical-digital interfaces that enable users to access digital content mapped to the physical space, such as spatial augmented reality [92, 93], imaginary interfaces [53, 10, 94], or physical loci [95], have introduced a new perspective on the interplay between our digital and physical realities. A digital layer superimposed upon the topography of the physical environment unlocks new possibilities to access and operate digital content. For instance, digital objects can be dynamically associated with specific regions of the physical space and accessed with a mere pointing of the hand. Multiple digital layers can also coexist at once in the same physical space leading to different experiences for different users.

To demonstrate how our software architecture can be used to readily implement such interactive scenarios, but also how multiple devices can render various views of the same content, we designed an instantiation of EUPHORIA with one PRODUCER and three CONSUMERS. A 6-camera Vicon motion capture system [96] was employed to track users and their pointing actions inside an area of $4\text{ m} \times 4\text{ m}$. Three output devices were considered to display the digital content pointed to by users: (1) a Samsung UE55D Wall Display with an active area of 13 m^2 and 8K Ultra-HD resolution, (2) a 46-inch Idem Platform Full-HD interactive table, and (3) a 7-inch Android tablet with 800×1280 screen resolution; see Figure 9, top. Three digital layers were created on top of the physical setting by associating multimedia content (in this specific implementation, links to YouTube videos) with

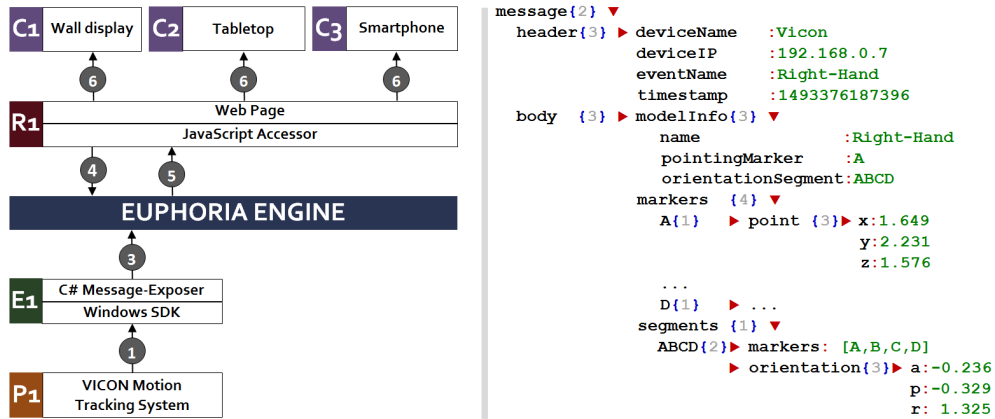


Figure 9: EUPHORIA applied to implement a multi-device scenario to access content in a smart environment: a wall display, an interactive tabletop, and a smartphone (top image) implement consumers in the EUPHORIA architecture (bottom left) that process motion tracking events encoded as JSON messages (bottom right).

physical regions in mid-air in the form of cuboids; see Figure 9, top for a visual illustration. When the user enters one of the physical regions, the associated content is rendered on all devices, according to the capabilities of

each device: from the personal, mobile device to the large ambient display, screen size, display quality, and computing resources increase. Therefore, the presentation of content on each device was adapted to best match the display and computing capabilities of each device. Using EUPHORIA made this task simple, where each device was implemented as a CONSUMER that required content to be delivered by EUPHORIA at a specific pixel resolution or for a specific complexity of the digital environment, *i.e.*, fewer graphical objects were displayed on devices with lesser computing capabilities.

Figure 9, bottom represents the instantiation of EUPHORIA for this specific implementation as well as examples of event messages. The location of the user in the physical environment is detected, tracked, and reported by the Vicon system using the Vicon Datastream SDK (dataflow ❶ in Figure 9) and a custom Message-Expose-Layer creates the messages with the information about the Vicon event. The same layer shares the messages with the `node.js` server (dataflow ❸ in Figure 9) via HTTP POST requests. To receive those messages, the large display, the interactive tabletop, and the tablet are instantiated as CONSUMERS and each run a web page implementing `WebSockets` (dataflows ❹ and ❺ in Figure 9). When the user is inside a predefined region of the physical space, the digital content associated to that region is rendered in each web page. Figure 9, top shows a visual representation of the physical regions as cuboids superimposed on a schematic representation of the physical space. Rendering of the graphical objects was implemented using the `three.js` 3-D library for JavaScript.³ Different versions of the web page are loaded on each device, depending on the corresponding pixel resolution, screen size, and the computing resources available, *i.e.*, the large display offers a full view of the entire digital environment, while smaller subsets of filtered objects are shown on the tabletop and the mobile device, respectively; see Figure 9, top.

5.3. A Wearable User Interface for Managing Social Notifications in an In-Vehicle Smart Environment

We continue with another use case of EUPHORIA, this time for a specific smart environment – the interior of a connected car [97, 98] – to show the usefulness and adaptability of EUPHORIA to a variety of physical settings and contexts of use. For the purpose of this use case, we are interested in the in-vehicle user interface [99] and an application scenario for delivering the

³three.js - JavaScript 3D library, <https://threejs.org/>

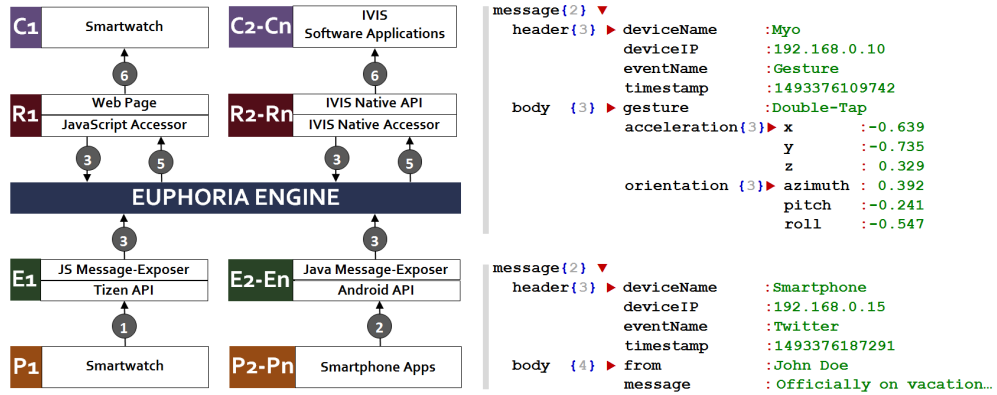


Figure 10: EUPHORIA applied to implement a wearable user interface for social notifications in an in-vehicle smart environment: a smartwatch (top) acts as both producer and consumer in the EUPHORIA architecture (bottom left) to process events from smartphone applications, but also to enter commands (bottom right).

driver with recent notifications from their social networks, such as incoming text messages on their mobile phone or recent tweets. Recently, there has been a growing interest in designing gesture user interfaces to interact with the infotainment system of modern, connected cars [100, 101, 102], so we

follow this design trend in our use case as well. The driver’s smartphone is connected to EUPHORIA and alerts the architecture of incoming calls, messages, or updates from social networks, such as the latest tweets [103]. In the case of any of those events occurring, feedback is provided to the driver in the form of audio messages read to the driver by the In-Vehicle Infotainment System (IVIS) of the car or as subtle vibrations on the wrist; see Figure 10, top for a visual illustration of the in-vehicle environment. The smartwatch also acts as an input device that can detect taps, touches, and motion gestures performed with the wrist or the hand in mid-air.

Figure 10, bottom represents the instantiation of EUPHORIA for this specific implementation as well as examples of event messages. For this use case, there are several PRODUCERS and CONSUMERS. PRODUCERS are represented by the smartwatch that reads user input and by the applications running on the smartphone that notify EUPHORIA about incoming phone calls, messages, or tweets. The IVIS system (simulated with a software application running on a laptop for this implementation) instantiates a CONSUMER that receives messages from PRODUCERS and delivers them to other subscribing CONSUMERS. A particularity of this scenario is that the smartwatch acts both as a PRODUCER and as a CONSUMER since it detects user input, but also notifies the driver about incoming alerts via vibrotactile feedback. The double role of the smartwatch is readily handled in EUPHORIA due to the separation of layers and the modularity of our architecture.

6. CONCLUSION

We introduced EUPHORIA, an asynchronous event-driven software architecture for supporting adaptable and flexible interaction for end users using heterogeneous input/output devices in smart environments throughout the life cycle: from prototyping to evaluation. Our technical evaluation showed that EUPHORIA delivers real-time response even for application scenarios involving large messages exchanged between a large number of producers and consumers from and to a reasonable large variety of devices. Future research must be conducted in order to expand these outcomes to significantly more complex and demanding scenarios such as smart cities, potentially involving thousands of concurrent users, such as in mass-computer interaction. Programming and connecting to EUPHORIA using protocols, standards, and languages developed for the web, such as **JavaScript**, represent the default choices for instantiating our software architecture to practical appli-

cations. Moreover, EUPHORIA enables simple integration of applications, devices, APIs, and source code independently of their software/hardware platforms, operating systems, and native programming languages as long as `WebSockets` support is ensured.

We deliver the open source code of EUPHORIA for free for research and development purposes at the web address <http://www.eed.usv.ro/mintviz/resources/Euphoria> along with examples of application scenarios. In this way, we hope to foster advances and developments in new software architecture initiatives for our increasingly complex smart environments, but also to readily support implementations of novel interactive systems and applications for smart environments of all kinds. Since multiple, potentially heterogeneous, input/output devices can be integrated in EUPHORIA in various smart environments, researchers and practitioners are prompted to design, invent, and test new interaction modalities.

ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI-UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0917, contract no. 21PCCDI/2018, within PNCDI III, sub-project P₂, “*Efficient communications based on smart devices for in-car augmented reality interactive applications.*” Research was conducted in the Machine Intelligence and Information Visualization Lab (MintViz) of the MANSiD Research Center. The infrastructure was provided by the University of Suceava and was partially supported from the project “*Integrated center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control*”, No. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness, co-funded from the European Regional Development Fund.

References

- [1] S. Poslad, [Ubiquitous computing: smart devices, environments and interactions](#), John Wiley & Sons, 2011. doi:10.1002/9780470779446.
URL <http://dx.doi.org/10.1002/9780470779446>
- [2] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt, [A unifying reference framework for multi-target user interfaces](#), *Interacting with Computers* 15 (3) (2003) 289–308. doi:10.1016/S0953-5438(03)00010-9.
URL [https://doi.org/10.1016/S0953-5438\(03\)00010-9](https://doi.org/10.1016/S0953-5438(03)00010-9)

- [3] N. F. Raun, Smart environment using internet of things(iots) - a review, in: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016, pp. 1–6. doi:[10.1109/IEMCON.2016.7746313](https://doi.org/10.1109/IEMCON.2016.7746313).
- [4] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, [Internet of things \(iot\): A vision, architectural elements, and future directions](#), Future generation computer systems 29 (7) (2013) 1645–1660. doi:[10.1016/j.future.2013.01.010](https://doi.org/10.1016/j.future.2013.01.010). URL <http://doi.org/10.1016/j.future.2013.01.010>
- [5] A. Greenfield, *Everyware: The Dawning Age of Ubiquitous Computing*, Peachpit Press, Berkeley, CA, USA, 2006.
- [6] P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*, River Publishers, 2013.
- [7] O.-A. Schipor, W. Wu, W.-T. Tsai, R.-D. Vatavu, [Software architecture design for spatially-indexed media in smart environments](#), Advances in Electrical and Computer Engineering 17 (2) (2017) 17–22. doi:[10.4316/AECE.2017.02003](https://doi.org/10.4316/AECE.2017.02003). URL <http://dx.doi.org/10.4316/AECE.2017.02003>
- [8] R.-D. Vatavu, C.-M. Chera, W.-T. Tsai, [Gesture profile for web services: An event-driven architecture to support gestural interfaces for smart environments](#), in: International Joint Conference on Ambient Intelligence, Springer, 2012, pp. 161–176. URL https://doi.org/10.1007/978-3-642-34898-3_11
- [9] Y. Lou, W. Wu, R.-D. Vatavu, W.-T. Tsai, [Personalized gesture interactions for cyber-physical smart-home environments](#), Science China Information Sciences 60 (7) (2017) 072104. URL <https://doi.org/10.1007/s11432-015-1014-7>
- [10] S. Gustafson, D. Bierwirth, P. Baudisch, [Imaginary interfaces: Spatial interaction with empty hands and without visual feedback](#), in: Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, UIST '10, ACM, New York, NY, USA, 2010, pp. 3–12. doi:[10.1145/1866029.1866033](https://doi.org/10.1145/1866029.1866033). URL <http://doi.acm.org/10.1145/1866029.1866033>
- [11] B.-F. Gheran, J. Vanderdonckt, R.-D. Vatavu, [Gestures for smart rings: Empirical results, insights, and design implications](#), in: Proceedings of the 2018 Designing Interactive Systems Conference, DIS '18, ACM, New York, NY, USA, 2018, pp. 623–635. doi:[10.1145/3196709.3196741](https://doi.org/10.1145/3196709.3196741). URL <http://doi.acm.org/10.1145/3196709.3196741>
- [12] ThalmicLabs, [Myo gesture control armband — wearable technology by thalmic labs](#) (2018). URL <https://www.myo.com/>
- [13] G. Bailly, J. Müller, M. Rohs, D. Wigdor, S. Kratz, [Shoesense: A new perspective on gestural interaction and wearable applications](#), in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, ACM, New York, NY, USA, 2012, pp. 1239–1248. doi:[10.1145/2207676.2208576](https://doi.org/10.1145/2207676.2208576). URL <http://doi.acm.org/10.1145/2207676.2208576>
- [14] G. Fortino, D. Parisi, V. Pirrone, G. Di Fatta, [Bodycloud: A saas approach for community body sensor networks](#), Future Gener. Comput. Syst. 35 (2014) 62–79. doi:[10.1016/j.future.2013.12.015](https://doi.org/10.1016/j.future.2013.12.015). URL <http://dx.doi.org/10.1016/j.future.2013.12.015>

- [15] P.-V. Cioată, R.-D. Vatavu, [In tandem: Exploring interactive opportunities for dual input and output on two smartwatches](#), in: Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion, IUI '18 Companion, ACM, New York, NY, USA, 2018, pp. 60:1–60:2. doi:10.1145/3180308.3180369. URL <http://doi.acm.org/10.1145/3180308.3180369>
- [16] X. A. Chen, T. Grossman, D. J. Wigdor, G. Fitzmaurice, [Duet: Exploring joint interactions on a smart phone and a smart watch](#), in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '14, ACM, New York, NY, USA, 2014, pp. 159–168. doi:10.1145/2556288.2556955. URL <http://doi.acm.org/10.1145/2556288.2556955>
- [17] A. Esteves, E. Velloso, A. Bulling, H. Gellersen, [Orbits: Gaze interaction for smart watches using smooth pursuit eye movements](#), in: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15, ACM, New York, NY, USA, 2015, pp. 457–466. doi:10.1145/2807442.2807499. URL <http://doi.acm.org/10.1145/2807442.2807499>
- [18] M. Villari, A. Celesti, M. Fazio, A. Puliafito, [Alljoyn lambda: An architecture for the management of smart environments in iot](#), in: Smart Computing Workshops (SMARTCOMP Workshops), 2014 International Conference on, IEEE, 2014, pp. 9–14. doi:10.1109/SMARTCOMP-W.2014.7046676. URL <https://doi.org/10.1109/SMARTCOMP-W.2014.7046676>
- [19] G. Fortino, A. Guerrieri, W. Russo, C. Savaglio, [Middlewares for smart objects and smart environments: overview and comparison](#), in: Internet of Things Based on Smart Objects, Springer, 2014, pp. 1–27. doi:10.1007/978-3-319-00491-4_1. URL https://doi.org/10.1007/978-3-319-00491-4_1
- [20] M. Nebeling, E. Teunissen, M. Husmann, M. C. Norrie, [Xdkinect: Development framework for cross-device interaction using kinect](#), in: Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '14, ACM, New York, NY, USA, 2014, pp. 65–74. doi:10.1145/2607023.2607024. URL <http://doi.acm.org/10.1145/2607023.2607024>
- [21] C. Roda, A. Rodríguez, E. Navarro, V. López-Jaquero, P. González, [Towards an architecture for a scalable and collaborative ami environment](#), in: Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, Springer, 2016, pp. 311–323. URL https://doi.org/10.1007/978-3-319-40159-1_26
- [22] R. D. Hill, [Event-response systems: A technique for specifying multi-threaded dialogues](#), SIGCHI Bull. 18 (4) (1986) 241–248. doi:10.1145/1165387.275637. URL <http://doi.acm.org/10.1145/1165387.275637>
- [23] D. L. Parnas, [On the criteria to be used in decomposing systems into modules \(reprint\)](#), in: M. Broy, E. Denert (Eds.), Software Pioneers., Springer Berlin Heidelberg, 2002, pp. 411–427. doi:10.1007/978-3-642-59412-0_26. URL https://doi.org/10.1007/978-3-642-59412-0_26
- [24] ISO, [Iec 25000 software and system engineering—software product quality requirements and evaluation \(square\)—guide to square](#), International Organization for Standardization. URL <https://www.iso.org/standard/35683.html>
- [25] F. Schneider, B. Berenbach, [A literature survey on international standards for](#)

- systems requirements engineering, *Procedia Computer Science* 16 (2013) 796–805.
URL <https://doi.org/10.1016/j.procs.2013.01.083>
- [26] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, R. Little, *Documenting software architectures: views and beyond*, Pearson Education, 2002.
 - [27] D. Garland, M. Shaw, *An introduction to software architecture, advances in software engineering and knowledge engineering* (1993).
URL https://doi.org/10.1142/9789812798039_0001
 - [28] D. E. Perry, A. L. Wolf, *Foundations for the study of software architecture*, *ACM SIGSOFT Software Engineering Notes* 17 (4) (1992) 40–52. doi:10.1145/141874.141884.
URL <http://doi.acm.org/10.1145/141874.141884>
 - [29] *A metamodel for the runtime architecture of an interactive system: The uims tool developers workshop*, *SIGCHI Bull.* 24 (1) (1992) 32–37. doi:10.1145/142394.142401.
URL <http://doi.acm.org/10.1145/142394.142401>
 - [30] P. D. Bruza, T. Van der Weide, *The semantics of data flow diagrams*, University of Nijmegen, Department of Informatics, Faculty of Mathematics and Informatics, 1989.
 - [31] E. Yourdon, L. L. Constantine, *Structured design: Fundamentals of a discipline of computer program and systems design*, Prentice-Hall, Inc., 1979.
 - [32] C. Moxey, M. Edwards, O. Etzion, M. Ibrahim, S. Iyer, H. Lalanne, M. Monze, M. Peters, Y. Rabinovich, G. Sharon, et al., *A conceptual model for event processing systems*, IBM Redguide publication doi:10.1.1.454.8442.
 - [33] K. M. Chandy, *Event-driven applications: Costs, benefits and design approaches*, Gartner Application Integration and Web Services Summit 2006.
 - [34] B. M. Michelson, *Event-driven architecture overview*, Patricia Seybold Group 2.
 - [35] H. Taylor, A. Yochem, L. Phillips, F. Martinez, *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise*, 1st Edition, Addison-Wesley Professional, 2009.
 - [36] D. M. Hilbert, D. F. Redmiles, *Extracting usability information from user interface events*, *ACM Computing Surveys (CSUR)* 32 (4) (2000) 384–421. doi:10.1145/371578.371593.
URL <https://doi.org/10.1145/371578.371593>
 - [37] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, *System architecture directions for networked sensors*, *SIGARCH Comput. Archit. News* 28 (5) (2000) 93–104. doi:10.1145/378995.379006.
URL <http://doi.acm.org/10.1145/378995.379006>
 - [38] M. Ilyas, I. Mahgoub, *Smart Dust: Sensor network applications, architecture and design*, CRC press, 2016.
 - [39] W. Wang, Y. Xu, M. Khanna, *A survey on the communication architectures in smart grid*, *Computer Networks* 55 (15) (2011) 3604–3629.
URL <https://doi.org/10.1016/j.comnet.2011.07.010>
 - [40] C.-h. Yang, G. Zhabelova, C.-W. Yang, V. Vyatkin, *Cosimulation environment for event-driven distributed controls of smart grid*, *IEEE Transactions on Industrial Informatics* 9 (3) (2013) 1423–1435. doi:10.1109/TII.2013.2256791.
URL <https://doi.org/10.1109/TII.2013.2256791>

- [41] P. Shaya, P. Sander, D. Remco, G. Paul, [A systematic literature review on the architecture of business process management systems](#), *Inf. Syst.* 66 (C) (2017) 43–58. doi:10.1016/j.is.2017.01.007.
URL <https://doi.org/10.1016/j.is.2017.01.007>
- [42] K. Julian, W. Benjamin, W. Dirk, P. Loos, [Event-driven business process management: where are we now? a comprehensive synthesis and analysis of literature](#), *Business Process Management Journal* 20 (4) (2014) 615–633.
URL <https://doi.org/10.1108/BPMJ-07-2013-0092>
- [43] E. Patti, A. Acquaviva, M. Jahn, F. Pramudianto, R. Tomasi, D. Rabourdin, J. Virgone, E. Macii, [Event-driven user-centric middleware for energy-efficient buildings and public spaces](#), *IEEE Systems Journal* 10 (3) (2016) 1137–1146. doi:10.1109/JSYST.2014.2302750.
URL <https://www.doi.org/10.1109/JSYST.2014.2302750>
- [44] J. Morales, M. Garcia, [Geosmart cities: Event-driven geoprocessing as enabler of smart cities](#), in: *Smart Cities Conference (ISC2)*, 2015 IEEE First International, IEEE, 2015, pp. 1–6. doi:10.1109/ISC2.2015.7366207.
URL <https://www.doi.org/10.1109/ISC2.2015.7366207>
- [45] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. B. Benosman, B. Linares-Barranco, [On the use of orientation filters for 3d reconstruction in event-driven stereo vision](#), *Front Neurosci.*
URL <https://doi.org/10.3389/fnins.2014.00048>
- [46] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al., [A million spiking-neuron integrated circuit with a scalable communication network and interface](#), *Science* 345 (6197) (2014) 668–673. doi:10.1126/science.1254642.
URL <https://doi.org/10.1126/science.1254642>
- [47] L. Li, D. W. Ho, S. Xu, [A distributed event-triggered scheme for discrete-time multi-agent consensus with communication delays](#), *IET Control Theory & Applications* 8 (10) (2014) 830–837. doi:10.1049/iet-cta.2013.0761.
URL <https://doi.org/10.1049/iet-cta.2013.0761>
- [48] M. Weiser, R. Gold, J. S. Brown, [The origins of ubiquitous computing research at parc in the late 1980s](#), *IBM Syst. J.* 38 (4) (1999) 693–696. doi:10.1147/sj.384.0693.
URL <http://dx.doi.org/10.1147/sj.384.0693>
- [49] E. Aarts, R. Harwig, M. Schuurmans, [The invisible future: the seamless integration of technology into everyday life](#), McGraw-Hill, Inc., New York, NY, USA, 2002, Ch. Ambient Intelligence, pp. 235–250.
URL <http://dl.acm.org/citation.cfm?id=504949.504964>
- [50] M. Kuniavsky, [Smart Things: Ubiquitous computing user experience design](#), Morgan Kaufmann, Burlington, MA, USA, 2010. doi:10.1016/C2009-0-20057-2.
URL <http://dx.doi.org/10.1016/C2009-0-20057-2>
- [51] D. Cook, S. K. Das, [Smart Environments: Technology, Protocols and Applications](#), Wiley-Interscience, 2004. doi:10.1002/047168659X.
URL <http://dx.doi.org/10.1002/047168659X>
- [52] D. Korzun, I. Galov, A. Kashevnik, S. Balandin, [Virtual shared workspace for smart spaces and m3-based case study](#), in: *Open Innovations Association FRUCT*,

- Proceedings of 15th Conference of, IEEE, 2014, pp. 60–68. doi:10.1109/FRICT.2014.6872437.
URL <https://doi.org/10.1109/FRICT.2014.6872437>
- [53] R.-D. Vatavu, A. Mossel, C. Schönauer, *Digital vibrons: Understanding users' perceptions of interacting with invisible, zero-weight matter*, in: Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI '16, ACM, New York, NY, USA, 2016, pp. 217–226. doi:10.1145/2935334.2935364.
URL <http://doi.acm.org/10.1145/2935334.2935364>
- [54] M. A. Rahman, M. S. Hossain, *A gesture-based smart home-oriented health monitoring service for people with physical impairments*, in: Proceedings of the 14th International Conference on Inclusive Smart Cities and Digital Health - Volume 9677, ICOST 2016, Springer-Verlag, Berlin, Heidelberg, 2016, pp. 464–476. doi:10.1007/978-3-319-39601-9_42.
URL https://doi.org/10.1007/978-3-319-39601-9_42
- [55] M. J. Deen, *Information and communications technologies for elderly ubiquitous healthcare in a smart home*, Personal Ubiquitous Comput. 19 (3-4) (2015) 573–599. doi:10.1007/s00779-015-0856-x.
URL <http://dx.doi.org/10.1007/s00779-015-0856-x>
- [56] M.-K. Le, H.-T. Chang, Y.-M. Chang, Y.-H. Hu, H.-T. Chen, *An efficient multilevel healthy cloud system using spark for smart clothes*, in: Computer Symposium (ICS), 2016 International, IEEE, 2016, pp. 182–186. doi:10.1109/ICS.2016.0044.
URL <https://doi.org/10.1109/ICS.2016.0044>
- [57] A. Pfister, A. M. West, S. Bronner, J. A. Noah, *Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis*, Journal of medical engineering & technology 38 (5) (2014) 274–280. doi:10.3109/03091902.2014.909540.
URL <https://doi.org/10.3109/03091902.2014.909540>
- [58] M. Lui, J. D. Slotta, *Immersive simulations for smart classrooms: exploring evolutionary concepts in secondary science*, Technology, Pedagogy and Education 23 (1) (2014) 57–80. doi:10.1080/1475939X.2013.838452.
URL <https://doi.org/10.1080/1475939X.2013.838452>
- [59] B. Brown, K. O'Hara, T. Kindberg, A. Williams, *Crowd computer interaction*, in: CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09, ACM, New York, NY, USA, 2009, pp. 4755–4758. doi:10.1145/1520340.1520733.
URL <http://doi.acm.org/10.1145/1520340.1520733>
- [60] J.-Y. L. Lawson, J. Vanderdonckt, R.-D. Vatavu, *Mass-computer interaction for thousands of users and beyond*, in: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI EA '18, ACM, New York, NY, USA, 2018, pp. LBW032:1–LBW032:6. doi:10.1145/3170427.3188465.
URL <http://doi.acm.org/10.1145/3170427.3188465>
- [61] V. G. Motti, J. Vanderdonckt, *A computational framework for context-aware adaptation of user interfaces*, in: Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on, IEEE, 2013, pp. 1–12. doi:10.1109/RCIS.2013.6577709.

- URL <https://doi.org/10.1109/RCIS.2013.6577709>
- [62] D. Korzun, [On the smart spaces approach to semantic-driven design of service-oriented information systems](#), in: International Baltic Conference on Databases and Information Systems, Springer, 2016, pp. 181–195. doi:10.1007/978-3-319-40180-5_13.
URL https://doi.org/10.1007/978-3-319-40180-5_13
- [63] O. A. Schipor, et al., [Improving computer assisted speech therapy through speech based emotion recognition](#), in: Conference proceedings of eLearning and Software for Education (eLSE), Editura Univ. Carol I, 2014, pp. 101–104.
URL <http://arxiv.org/abs/1405.7796>
- [64] K. Bahreini, R. Nadolski, W. Westera, [Towards multimodal emotion recognition in e-learning environments](#), Interactive Learning Environments 24 (3) (2016) 590–605.
URL <https://doi.org/10.1080/10494820.2014.908927>
- [65] C. Ye, Y. Xia, Y. Sun, S. Wang, H. Yan, R. Mehmood, [Erar: An event-driven approach for real-time activity recognition](#), in: Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015 International Conference on, IEEE, 2015, pp. 288–293. doi:10.1109/IIKI.2015.69.
URL <https://doi.org/10.1109/IIKI.2015.69>
- [66] D. J. Cook, N. C. Krishnan, P. Rashidi, [Activity discovery and activity recognition: A new partnership](#), IEEE transactions on cybernetics 43 (3) (2013) 820–828. doi:10.1109/TSMCB.2012.2216873.
URL <https://doi.org/10.1109/TSMCB.2012.2216873>
- [67] S. S. Rautaray, A. Agrawal, [Vision based hand gesture recognition for human computer interaction: A survey](#), Artif. Intell. Rev. 43 (1) (2015) 1–54. doi:10.1007/s10462-012-9356-9.
URL <http://dx.doi.org/10.1007/s10462-012-9356-9>
- [68] M. R. Abid, E. M. Petriu, E. Amjadian, [Dynamic sign language recognition for smart home interactive application using stochastic linear formal grammar](#), IEEE Transactions on Instrumentation and Measurement 64 (3) (2015) 596–605.
URL <https://doi.org/10.1109/TIM.2014.2351331>
- [69] O. Schipor, S. Pentiuc, M. Schipor, [The utilization of feedback and emotion recognition in computer based speech therapy system](#), Elektronika ir Elektrotechnika 109 (3) (2011) 101–104.
URL <http://dx.doi.org/10.5755/j01.eee.109.3.181>
- [70] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, T. P. Jung, [Augmented brain computer interaction based on fog computing and linked data](#), in: Intelligent Environments (IE), 2014 International Conference on, IEEE, 2014, pp. 374–377. doi:10.1109/IE.2014.54.
URL <https://doi.org/10.1109/IE.2014.54>
- [71] R.-D. Vatavu, [Nomadic gestures: A technique for reusing gesture commands for frequent ambient interactions](#), J. Ambient Intell. Smart Environ. 4 (2) (2012) 79–93.
URL <http://dl.acm.org/citation.cfm?id=2350758.2350765>
- [72] R.-D. Vatavu, [Smart-pockets: Body-deictic gestures for fast access to personal data during ambient interactions](#), International Journal of HumanComputer Studies 103

- (2017) 1–21. doi:10.1016/j.ijhcs.2017.01.005.
URL <http://dx.doi.org/10.1016/j.ijhcs.2017.01.005>
- [73] M. Funk, A. Sahami, N. Henze, A. Schmidt, [Using a touch-sensitive wristband for text entry on smart watches](#), in: CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14, ACM, New York, NY, USA, 2014, pp. 2305–2310. doi:10.1145/2559206.2581143.
URL <http://doi.acm.org/10.1145/2559206.2581143>
- [74] W.-H. Chen, [Blowatch: Blowable and hands-free interaction for smartwatches](#), in: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15, ACM, New York, NY, USA, 2015, pp. 103–108. doi:10.1145/2702613.2726961.
URL <http://doi.acm.org/10.1145/2702613.2726961>
- [75] C. Goumopoulos, A. Kameas, P. Hellas, [Smart objects as components of ubi-comp applications](#), International Journal of Multimedia and Ubiquitous Engineering 4 (3).
URL http://www.sersc.org/journals/IJMUE/vol4_no3_2009/1.pdf
- [76] F. Aiello, G. Fortino, R. Gravina, A. Guerrieri, [A java-based agent platform for programming wireless sensor networks](#), Comput. J. 54 (3) (2011) 439–454. doi:10.1093/comjnl/bxq019.
URL <http://dx.doi.org/10.1093/comjnl/bxq019>
- [77] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, [Spine: A domain-specific framework for rapid prototyping of wbsn applications](#), Softw. Pract. Exper. 41 (3) (2011) 237–265. doi:10.1002/spe.998.
URL <https://doi.org/10.1002/spe.998>
- [78] G. Fortino, A. Guerrieri, G. M. P. O'Hare, A. Ruzzelli, [A flexible building management framework based on wireless sensor and actuator networks](#), J. Netw. Comput. Appl. 35 (6) (2012) 1934–1952. doi:10.1016/j.jnca.2012.07.016.
URL <http://dx.doi.org/10.1016/j.jnca.2012.07.016>
- [79] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, Y. Al-Hazmi, [Fiware lab: managing resources and services in a cloud federation supporting future internet applications](#), in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, IEEE, 2014, pp. 792–799. doi:10.1109/UCC.2014.129.
URL <https://doi.org/10.1109/UCC.2014.129>
- [80] F. Lab, [Fiware lab](#) (2018).
URL <http://status.lab.fiware.org/>
- [81] E. Hall, [The Hidden Dimension](#) (Anchor Books a Doubleday Anchor Book), Anchor, 1966.
URL <http://www.worldcat.org/isbn/0385084765>
- [82] N. Marquardt, R. Diaz-Marino, S. Boring, S. Greenberg, [The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies](#), in: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, ACM, New York, NY, USA, 2011, pp. 315–326. doi:10.1145/2047196.2047238.
URL <http://doi.acm.org/10.1145/2047196.2047238>
- [83] A. P. Volpentesta, [A framework for human interaction with mobiquitous services](#)

- in a smart environment, *Comput. Hum. Behav.* 50 (C) (2015) 177–185. doi:
10.1016/j.chb.2015.04.003.
URL <http://dx.doi.org/10.1016/j.chb.2015.04.003>
- [84] D. Ledo, S. Greenberg, N. Marquardt, S. Boring, *Proxemic-aware controls: Designing remote controls for ubiquitous computing ecologies*, in: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI '15*, ACM, New York, NY, USA, 2015, pp. 187–198. doi:10.1145/2785830.2785871.
URL <http://doi.acm.org/10.1145/2785830.2785871>
- [85] I. Mocanu, O. A. Schipor, *A serious game for improving elderly mobility based on user emotional state*, in: *The International Scientific Conference eLearning and Software for Education, Vol. 2, "Carol I" National Defence University*, 2017, p. 487.
URL <https://doi.org/10.12753/2066-026x-17-154>
- [86] D. Chatzopoulos, C. Bermejo, Z. Huang, P. Hui, *Mobile augmented reality survey: From where we are to where we go*, *IEEE Access* 5 (2017) 6917–6950. doi:10.1109/ACCESS.2017.2698164.
URL <https://doi.org/10.1109/ACCESS.2017.2698164>
- [87] V. Wang, F. Salim, P. Moskovits, *The definitive guide to HTML5 WebSocket*, Vol. 1, Springer, 2013. doi:10.1007/978-1-4302-4741-8.
URL <https://doi.org/10.1007/978-1-4302-4741-8>
- [88] D. Skvorc, M. Horvat, S. Srbljic, *Performance evaluation of websocket protocol for implementation of full-duplex web streams*, in: *Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014 37th International Convention on, IEEE, 2014, pp. 1003–1008. doi:10.1109/MIPRO.2014.6859715.
URL <https://doi.org/10.1109/MIPRO.2014.6859715>
- [89] W. Buxton, *Lexical and pragmatic considerations of input structures*, *SIGGRAPH Comput. Graph.* 17 (1) (1983) 31–37. doi:10.1145/988584.988586.
URL <http://doi.acm.org/10.1145/988584.988586>
- [90] Microsoft, *Kinect for windows 1.5, 1.6, 1.7, 1.8. skeleton class* (2017).
URL <https://msdn.microsoft.com/en-us/library/microsoft.kinect.skeleton.aspx>
- [91] Logbar, *Ring zero* (2017).
URL <http://ringzero.logbar.jp/>
- [92] O. Bimber, R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*, A. K. Peters, Ltd., Natick, MA, USA, 2005.
- [93] Y. Ueda, Y. Asai, R. Enomoto, K. Wang, D. Iwai, K. Sato, *Body cyberization by spatial augmented reality for reaching unreachable world*, in: *Proceedings of the 8th Augmented Human International Conference, AH '17*, ACM, New York, NY, USA, 2017, pp. 19:1–19:9. doi:10.1145/3041164.3041188.
URL <http://doi.acm.org/10.1145/3041164.3041188>
- [94] P. Baudisch, H. Pohl, S. Reinicke, E. Wittmers, P. Lühne, M. Knaust, S. Köhler, P. Schmidt, C. Holz, *Imaginary reality gaming: Ball games without a ball*, in: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, UIST '13*, ACM, New York, NY, USA, 2013, pp. 405–410. doi:10.1145/2501988.2502012.

- URL <http://doi.acm.org/10.1145/2501988.2502012>
- [95] S. T. Perrault, E. Lecolinet, Y. P. Bourse, S. Zhao, Y. Guiard, *Physical loci: Leveraging spatial, object and semantic memory for command selection*, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15, ACM, New York, NY, USA, 2015, pp. 299–308. doi:10.1145/2702123.2702126.
URL <http://doi.acm.org/10.1145/2702123.2702126>
- [96] Vicon, *Motion capture systems - vicon* (2018).
URL <https://www.vicon.com/>
- [97] R. Coppola, M. Morisio, *Connected car: Technologies, issues, future trends*, ACM Comput. Surv. 49 (3) (2016) 46:1–46:36. doi:10.1145/2971482.
URL <http://doi.acm.org/10.1145/2971482>
- [98] M. K. Svangren, M. B. Skov, J. Kjeldskov, *The connected car: An empirical study of electric cars as mobile digital devices*, in: Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI '17, ACM, New York, NY, USA, 2017, pp. 6:1–6:12. doi:10.1145/3098279.3098535.
URL <http://doi.acm.org/10.1145/3098279.3098535>
- [99] P. Di Lena, S. Mirri, C. Prandi, P. Salomoni, G. Delnevo, *In-vehicle human machine interface: An approach to enhance eco-driving behaviors*, in: Proceedings of the 2017 ACM Workshop on Interacting with Smart Objects, SmartObject '17, ACM, New York, NY, USA, 2017, pp. 7–12. doi:10.1145/3038450.3038455.
URL <http://doi.acm.org/10.1145/3038450.3038455>
- [100] K. R. May, T. M. Gable, B. N. Walker, *Designing an in-vehicle air gesture set using elicitation methods*, in: Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '17, ACM, New York, NY, USA, 2017, pp. 74–83. doi:10.1145/3122986.3123015.
URL <http://doi.acm.org/10.1145/3122986.3123015>
- [101] K. May, T. M. Gable, X. Wu, R. R. Sardesai, B. N. Walker, *Choosing the right air gesture: Impacts of menu length and air gesture type on driver workload*, in: Adjunct Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '16 Adjunct, ACM, New York, NY, USA, 2016, pp. 69–74. doi:10.1145/3004323.3004330.
URL <http://doi.acm.org/10.1145/3004323.3004330>
- [102] T. M. Gable, K. R. May, B. N. Walker, *Applying popular usability heuristics to gesture interaction in the vehicle*, in: Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '14, ACM, New York, NY, USA, 2014, pp. 1–7. doi:10.1145/2667239.2667298.
URL <http://doi.acm.org/10.1145/2667239.2667298>
- [103] TwitterInc., *Twitter. it's what's happening*. (2018).
URL <https://twitter.com/?lang=en>