MIODMIT: A Generic Architecture for Dynamic Multimodal Interactive Systems

Martin Cronel[1], Bruno Dumas[2], Philippe Palanque[1,3](✉), and Alexandre Canny[1]

[1] ICS-IRIT, Université Paul Sabatier – Toulouse III, Toulouse, France
martin.cronel@gmail.com, {palanque,canny}@irit.fr
[2] University of Namur, Namur, Belgium
bruno.dumans@unamur.be
[3] Department of Industrial Design, Technical University Eindhoven, Eindhoven, Netherlands

Abstract. This paper proposes a generic interactive system architecture describing in a structured way, both hardware and software components of an interactive system. It makes explicit all the components that play a role in the information processing from input devices to the interactive application and back to the output devices. Along with the generic interactive system architecture the paper proposes a process for selecting and connecting those components in order to tune the generic interactive system architecture for a specific interactive application. This select, connect and tune-on-demand approach helps handle complexity of interactive applications featuring innovative interaction techniques by splitting the interactive software into dedicated functional components. It also supports design flexibility by making explicit the components impacted when the interaction design evolves. This interactive system architecture and its related process have been applied to the development of several real-life interactive systems and we illustrate their application on an interactive application offering multi-mice, multi-touch and leap motion interactions in the context of interactive cockpits of large civil aircrafts.

Keywords: Interactive systems engineering · Input/output devices integration Interaction techniques · Software architectures

1 Introduction

The diversification of technological platforms on which interactive systems are designed, developed and deployed significantly increases the complexity of designers' and developers' tasks. At the same time, such an ever-changing context has made it

© IFIP International Federation for Information Processing 2019
Published by Springer Nature Switzerland AG 2019
C. Bogdan et al. (Eds.): HCSE 2018, LNCS 11262, pp. 109–129, 2019.
https://doi.org/10.1007/978-3-030-05909-5_7

very difficult for researchers belonging to the engineering community on interactive systems, to provide generic approaches to support those tasks. Designers need to go beyond the interactive application design by providing new interaction techniques that encompass new input and output devices which can be very cumbersome to design and evaluate (as for instance fingers clustering in multi-touch interactions [29]). Developers of these systems are repetitively facing the same issues of: (i) new devices integration, software redesign (due to device drivers' evolution) and above all poor reliability of the resulting system due to the low level of maturity of the various components to integrate. Such constraints are even stronger in the area of critical systems where a failure may lead to catastrophic consequences.

This paper addresses these issues by proposing MIODMIT (Multiple Input and Output Devices and Multiple Interaction Techniques) generic interactive system architecture for integrating new input and output devices, along with their more and more (potentially multimodal) sophisticated interaction techniques. MIODMIT identifies the building components that have to be developed for integrating new devices as well as the building components for merging information from these devices in order to offer multimodal interaction to users. As such, MIODMIT helps developers with the design of systems exploiting advanced interaction technologies. While this interactive system architecture is generic (and can thus be applied to many types of interactive systems) it also comes with a set of attributes and related trade-offs giving freedom to developers using it, while constraining them when necessary. Due to its generic nature, MIODMIT needs to be tuned to and adapted for the interactive system under development, especially to the input and output devices and interaction techniques considered. Two case studies (including a real world critical system application) and two illustrative examples illustrate how the architecture is applied, as well as the benefits it brings.

The remainder of this paper is structured as follows. Next section describes relevant related work and characterizes input and output devices. Section 3 details MIODMIT making explicit how it decomposes interactive systems into connected components within a generic interactive system architecture. Section 4 presents the Tune-on-Demand process and makes explicit how to go from a tuned MIODMIT diagram to the implementation. Section 5 presents a real world case study in the area of interactive cockpits. As this application is rather complex, we also present a simple example of interactive system to demonstrate along the paper the application and functioning of the tune-on-demand process in its entirety. The last section concludes the paper, highlighting benefits and limitations of the contributions and identifying potential extensions.

2 Related Work

2.1 Software Side of Interactive Systems Architectures

Architectural models for multimodal interactive systems have been presented in research papers [38] and [39] as a way of explaining the various components of a given system. More generic ones have also been presented, but they are usually bound to one type of modality such as touch interaction [37] or speech interaction [21]. Toolkits and frameworks for supporting the development of prototypes and demonstrations have also been developed such as, for instance, PyMT toolkit [17] for interactive applications offering multi-touch interactions. Similarly SensScreen [34] is dedicated to interactive applications exploiting multimodal management of sensors distributed in the user environment and presents a very high level architecture dedicated to public displays. As far as interaction techniques are concerned, dedicated software architectures have been proposed but focusing on a specific problem raised by a specific kind of interaction technique in an interactive application. For instance [26] presents the Accelerated Touch Architecture and [15] the Layered Multi-touch Architecture but both only address specific problems related to touch input.

MUDRA [19] is one recent exception proposing a framework embedding a generic architecture for multimodal interaction. The main limitation of MUDRA architecture (in terms of genericity) lies in its hardware part which is restricted to a defined set of input devices and does not provide a generic approach making explicit how new devices can be instantiated.

Despite such architecture-based contributions for engineering systems, empirical studies have demonstrated that, generally, developers are coding from scratch [24] as the problem they are facing is only superficially addressed by the existing solutions.

2.2 Hardware Side of Interactive Systems Architectures: Input and Output Devices

In order to provide generic means to deal with the extent of future input and output devices, there is a need to characterize them and, in addition, to provide means for integrating input devices types (according to their characteristics) rather than their instances. Indeed, integration based on types provides an adequate mean to increase architecture genericity. The HCI community has been proposing several taxonomies of input devices taking into account both their hardware and software aspects.

In [6], the software side is prominent as the classification is more abstract and goes beyond the description of the physical capabilities of the input devices. They introduced the “virtual” and “logical” device concepts, which can be used to produce more versatile interaction techniques. The concept of virtual devices allows reasoning in terms of interaction methods without having to consider the input devices themselves. For instance, instead of designing the interaction techniques with low level mouse events (for instance using dx, dy relative quantity of movement as for a mouse), it is described using generic pointing events such as x, y screen coordinates. This allows replacing mouse input devices easily with other devices as long as they produce similar (compatible) pointing events. In order to support this, MIODMIT proposes the refinement of the virtual device concept into two distinctive components called Virtual Device and Logical Device as presented in the section dedicated to the architecture (see Fig. 3).

From a hardware perspective, we classify devices as in [30] as according to the discrete versus continuous nature of events provided. At an abstract level, MIODMIT architecture remains independent and thus generic whatever the category (continuous or discrete) the device belongs to. Nevertheless, the device type will be taken into account at the refinement time (i.e. during the development of the architecture components). This way of dealing with these two types of devices has been identified when integrating different devices such as keyboards, mice, speech recognition, touch input, and more recently gesture input, eye-tracking and speech synthesis. The only constraint (to ensure the correct functioning of the final system) is to make sure that data processing is consistent both in terms of input and output for each component throughout the entire pipe-line of information processing (from input to output) as identified in [28].

In the literature, much less work has been done on addressing the output side of interactive systems and while a plethora of input devices taxonomies is available, output device taxonomies are seldom. Noticeable exceptions are [18] which provides characterization of both input and output devices and [32] which is dedicated to multimodal output engineering. MIODMIT encompasses this work using the same decomposition for output devices as the one for input devices presented above.

3 MIODMIT: A Generic Architecture for Interactive Systems

The Multiple Input and Output Devices and Multiple Interaction Techniques (MIODMIT) generic interactive system architecture explicitly depicts the various components (both hardware and software) of an information pipe-line in modern interactive systems. In that sense, MIODMIT is compatible with the principles and objectives of the Model Driven Architecture approach at OMG (http://www.omg.org/mda). Such systems contain multiple input devices each providing an information flow that are usually fused with other ones to offer multiple (and often multimodal) interaction techniques. This architecture presents input and output flows as well as how they can be integrated altogether. The following sections present an overview of MIODMIT and detail functionalities and responsibilities of each component via an illustrative example.

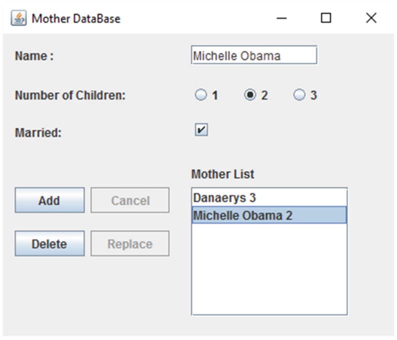
3.1 Illustrative Example with a Simple JAVA Application

In order to illustrate how MIODMIT is structured, we use a simple example application developed with Java Swing, presented in Fig. 1. This application allows users to add, modify and remove elements in a database. An element is composed of three attributes: a text field (the name), an enumerated field (the number of children) and a Boolean value (married or not). The list of elements in the database are displayed in the listbox (called mother list). Once added to the database, elements can be selected in that listbox to be deleted or modified.

In terms of input and output modalities, this application is standard, offering a mouse and a keyboard for input, and a computer screen for display.

3.2 MIODMIT Overview

MIODMIT is meant as a thinking and design tool for developers working on the development of advanced interactive systems. It seeks to clearly describe which components are to be designed, built or reused when envisioning such systems, and the interplay between these components.


Mother DataBase

Name : Michelle Obama

Number of Children: 1 [✓ 2] 3

Married: [✓]

Mother List

Danaerys 3

Michelle Obama 2

Fig. 1. Simple JAVA application

The overview of MIODMIT is presented in Fig. 2 while the full description of the generic architecture is provided in Fig. 3. Figure 2 is described from top (left to right) and then towards bottom (right to left). The overview of the architecture is composed of several components each of them represented by a rounded rectangle. When developing systems using MIODMIT developers have to describe the precise behavior of each component. Due to space constraints, we cannot present it here but the interest reader can access full details in [10].

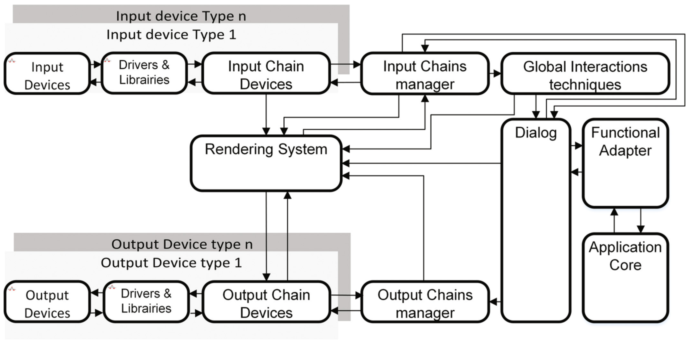


Fig. 2. Overview of MIODMIT

The grayed out boxes labeled "Input Device Type" handle events flow for each type of input device used (according to the classification presented in the related work section). For instance, having two mice and a voice-recognition system would require two separate "Input Device Type" boxes as they do not belong to the same type. The same holds for the output processing. Following the normal flow of events (in which the interactive system is idle waiting for input from users) a given "input device" sends events to the "driver & Library". The "input chain device type 1" transforms the raw data into higher-level information (e.g. transformation of the amount of motion of a mouse (dx, dy) into absolute coordinates for the mouse pointer).

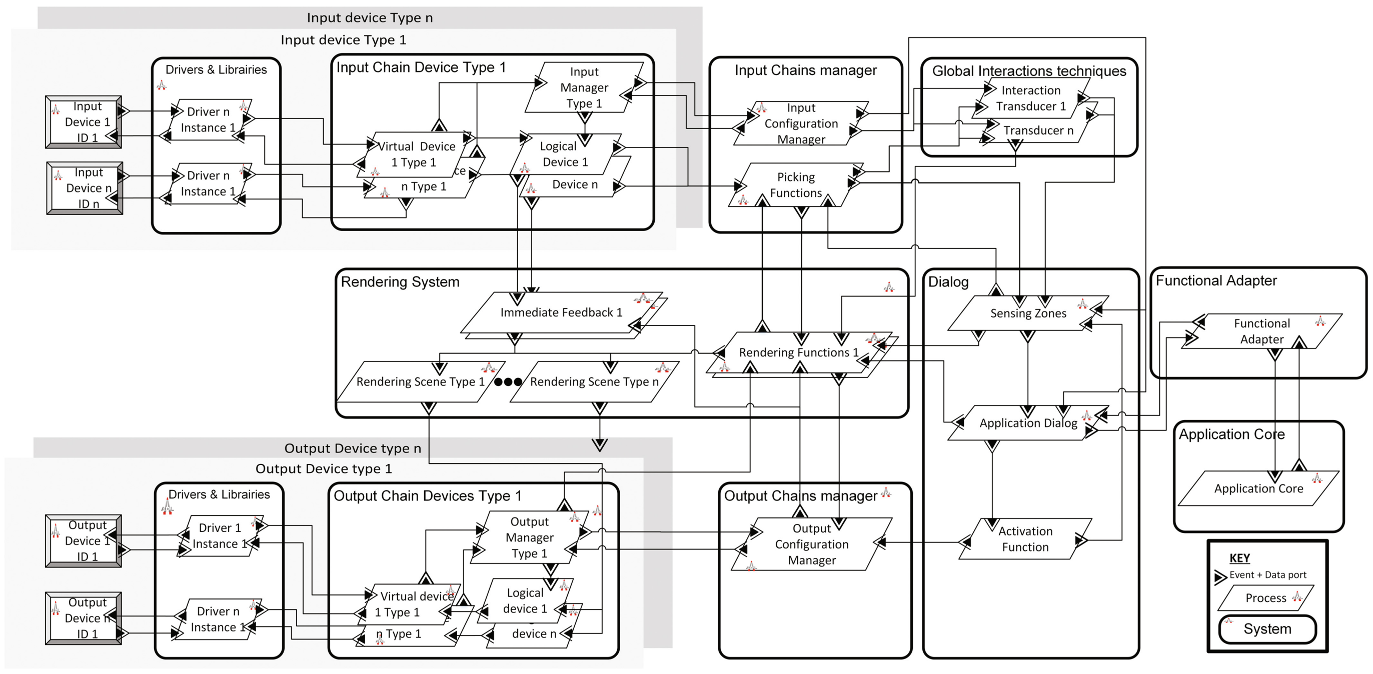


Fig. 3. Multiple Input and Output Devices and Multiple Interaction Techniques expressed in AADL [33]

Such information is then processed by the “input chains manager” (e.g. picking function connecting the input event to user interface objects) that possibly fuses information from the various input devices types. The input chain manager and its output counterpart are also responsible for managing dynamic reconfiguration of interaction when a failure occurs in the flow of events, thus being compliant with [27]. Fused information is then dispatched either directly to the “Dialog” or to “Global Interaction Technique” which behaves as a transducer as defined in [1] and then dispatch information to the “Dialog”. Both the “Dialog” and “Application Core” system have a similar responsibility as in standard interactive architecture models such as Seeheim [31] or ARCH [4]. The “Functional Adapter” have a similar responsibility as the “Functional Core Adaptor” in ARCH [4]. The output part processing is a mirror of the input side. The “rendering system” component in the middle of the diagram includes immediate feedback function and more sophisticated state-based rendering functions.

3.3 Inside the Details of MIODMIT

Figure 3 presents the refinement of Fig. 2 making explicit both the content of each component and the information flow between components. Even in its detailed description, MIODMIT remains abstract on purpose which means that each component may be decomposed into several classes. More details about that aspect are given in the case studies section.

The MIODMIT architecture, presented in Fig. 3, uses the AADL notation [33], a standard for describing software architecture which been applied in several domains including automotive and aeronautics. Other notations could have been used but the standard nature of AADL eases its understanding. We do not provide here a description of the elements of AADL but a key is provided at the bottom of Fig. 3.

Input Devices
The first layer of this interactive system architecture is composed of the physical input devices directly handled by the users. In Fig. 3, they are defined with a number corresponding to their type and an ID corresponding to their number within a given type. For instance, two input devices (Input Device 1 ID 1 and Input Device n ID n) represent the fact that both input devices are similar (two mice for instance) but with small functionalities that need to differentiate their drivers. If two identical mice were to be connected, ID will then allow to differentiate them. Figure 3 only details one generic type of device. The addition of a new modality associated with a new input device results in a new set of input devices, new driver(s) and a new input chain, represented within a gray rectangle. In the case of our Java app example, the mouse and keyboard are the input devices.

Drivers
Usually only one device driver is used in an operating system per type of device at a time. This software component is in charge of retrieving (or receiving) raw information from the hardware input device and makes it available for the upper layer.

The driver may also allow some control over the hardware components of the physical input device such as the sampling frequency (e.g. of the touch acquisition in

[25]) or to provide user identification [36]. Drivers are either provided together with the hardware (typically for specific non-standard input devices such as gesture tracking cameras) or by the operating system when the input device is rather standard or has been around for a significant amount of time (typically several years). For instance, mouse and keyboard drivers are handled at the OS level. This component is usually OS dependent and includes the libraries and the API needed for using the device. The API can be a composite object in order to translate information from a low-level language or OS level, to the higher one in the information flow.

Virtual Devices and Logical Devices
Virtual devices are necessary extensions to the logical device concept of Buxton [6] as we take into account explicitly both hardware and software aspects.

For instance, as quickly introduced in the related work section, for a mouse, the virtual device will be a software component mirroring the state of each physical button (pressed or released) and the amount of motion (dx, dy), while the logical device handles the cursor pointer positions (x, y). It is important to note that this is independent of the rendering aspect that is handled in a dedicated set of components ("rendering system" and "output device type" in the architecture). Indeed, these (x, y) values are abstract and how they are presented to the user or fused before rendering is a responsibility not belonging to the input chain.

This distinction allows using a single virtual device, with different logical devices in order to propose different interaction techniques. For instance, with a gesture recognition device such as Leap Motion, with one virtual device (a computerized hand), one logical device could be dedicated to two-dimension interactions, as a pointing device, while another could be used in a 3D environment.

Virtual and logical device components are transducers (as defined in [6]) as they provide processed information to a higher level. Virtual devices can be dynamically instantiated as with plug-and-play devices. Logical devices might also be dynamically instantiated at operations time, as for multi-touch input devices where a "logical input device" component is created each time a finger touches the device [16].

Input Manager
The "Input Manager" component manages the availability and instantiation of devices in order to address configuration and dynamic reconfiguration. This layer is composed of several managers, one per configuration of input devices, each of them being responsible for handling the dynamic aspects of input devices of the same type. At initialization time, these managers are responsible for the instantiation of the input devices and inform "Logical Input Devices" components.

Input Configuration Manager (Input Chains Manager)
One of the specificities of MIODMIT is its intrinsic ability to support dynamic reconfiguration of the interaction techniques. It is a functionality of paramount importance for different systems such as: critical systems, systems with a long exploitation life, systems with long start up procedures (such as civil aircrafts), or systems with a high replacement cost. Indeed, if one or several modalities fail at runtime (also called operation time), it may be critical to offer other modalities for allowing operators to perform their tasks even with degraded or less efficient

interactions. To this end, MIODMIT includes an “input configuration manager” which is responsible for handling reconfigurations. It is a unique software component, whatever the amount of input devices is. At runtime, the “input configuration manager” component links the (possibly) dynamically instantiated “logical input devices” components to the relevant “interaction transducers” which are in charge of processing users’ input.

Beyond that linking aspect, this manager is in charge of verifying the physical input device configurations to ensure that the current configuration still allows users to trigger all the needed events, and thus to produce all the information that the interactive application is expecting. In the case of input devices failure, the manager would reconfigure the interaction in such a way that the remaining physical input devices could compensate the failing ones (provided that this aspect has been addressed at design time). An example of a behavioral description of such a reconfiguration manager can be found in [27] (and is highlighted in the case study section). It is important to note that in critical systems, the failure must have been expected and so the reconfiguration possibilities are predefined (so that operators can be trained) and not dynamically made.

Input Device Type and Output Device Type
The gray boxes labelled Input Device Type and Output Device Type do not represent component of the architecture. They represent the fact that the components “drivers and libraries” and “input/output chain device type 1” have to handle all the input/output devices of the same type. Mouse and keyboard are examples of different input device type. A screen and a loudspeaker are examples of different output device type.

Picking Functions (Input Chains Manager)
This component channels the input event to the intended sensing zones. These functions are generally handled by the OS for standard devices, but for non-standard interaction, recipients of events must be designed and implement a picking function.

Interaction Transducers (Global Interaction Techniques)
The “interaction transducers” are responsible for generating the high level user events used by the application to trigger the various commands it provides. Usually, one “interaction transducer” is associated with one global interaction technique. These “interaction transducers” perform the recognition of a specific interaction technique which is not linked to a sensing zone such as a button (e.g. an interaction technique such as a double click is a composition of 2 simple clicks performed within a predefined temporal window and the click is a succession of a “down” event followed by an “up” event on a button). In the case of multi-touch interaction techniques, the transducer receives fingers’ (logical devices) movement and triggers the appropriate high level events based on the gesture recognition or the clustering of fingers. A basic transducer description for two mice can be found in [7], a more detailed one in [1], one for a keyboard in [2] and one for a tactile screen in [16]. A detailed behavioral description of such component can be found in [15]. The global interaction technique component is thus made of multiple interaction transducers, each of them bringing defining one or several interaction technique. The global interaction techniques are not necessarily linked to a special zone. For example, on a Samsung smartphone with

Samsung Experience, a palm swift on the screen will take a snapshot of the screen, whatever the state of the OS is or whatever application is launched.

Sensing Zone (Dialogue and Application Core)
To match the WIMP paradigm, "sensing zone" components include concepts such as interactive widgets (e.g. radio boxes, buttons…). With post-WIMP interaction, those objects are not enough, thus "sensing zones" are to be defined, containing representation parameters (coming from design), their precise behavior as well as how this behavior is triggered (i.e. their local interaction technique). For instance, a "sensing zone" reacts to a specific spoken sentence when highlighted, whereas the rest of the application will not react to the same sentence. The "ok Google" sentence always triggers an event on Android, whichever app is currently active. It is thus a nice example of a global interaction technique whereas a sentence such as "tell me if it's going to rain today" triggers a result only when the Google Now app is active.

Application Dialog
The "Application Dialog" component is a composite component and represents the functional behavior of the application as defined in ARCH [4]. The "Activation Function" component activates or disables the "Sensing Zone" depending on the current state of the application.

Functional Adapter and Application Core
The "Functional Adapter" component adapts the flow of information from the "Dialog" to the "Application Core" as defined in ARCH [4]. The "Application Core" is the component that is responsible of providing the data and services of the application.

Rendering System
The rendering system is composed of several components of two main types: the "rendering functions" and the "rendering scenes". The "rendering function" describes how to present the information of a specific state which might be distributed information in the other components of MIODMIT. The immediate feedback is an example of such a rendering function depending mainly on the information in components "logical devices" and "global interaction techniques". A "Rendering Scene" component composes all rendering function of a given type (e.g. a graphical scene, a sound scene…). These components prepare the final composition of the information before the output processing. They are thus connected to one or several output devices which can effectively present the information to the users.

Output Chain Manager
The "Output Chain Manager" offers the same functionalities as the "Input Chains Manager" presented above. Nonetheless, the main difference is that while the input is event-based, the output is state-based, thus, there is no equivalent to the output, of the "Global Interactions Techniques" component.

Fusion and Fission Engines: A Distributed Function
MIODMIT does not include specific components for fusion and fission as other architecture do [38]. Indeed, in that case, only one device was used (a photo browser) and fusion was located by the device. In MIODMIT, fusion can occur at different levels (e.g. low level with two input devices for a CTRL+Click event or high level where

merging speech sentences with mouse event for a “Put that there” multimodal command). Fusion and fission mechanism are thus to be specified within components as for example, fusion engines within the “Global Interaction Technique” component to fuse two (or more) high-level events into a multimodal interaction technique. Engineering issues of multimodal input interactions for a single user have been studied and classified in [6] and a taxonomy based on this classification has been proposed in [23]. Indeed, the various models identified in that survey spread over several components of MIODMIT.

4 Tune-on-Demand Process

According to the type and the number of input and output devices and according to the complexity of the multimodal interaction techniques the generic MIODMIT architecture has to be adapted (tuned) to the specificities of the application under consideration. This section presents a systematic process for tuning MIDOMIT that will be exemplified on case studies in Sect. 5.

4.1 Prototyping

Figure 4 presents a process to tune the generic architecture into a specific one. The top left-hand side (labeled prototyping) represents an abstraction of the user-centered design process of interactive systems. This part is presented in a very abstract way only highlighting the productions that are used as input in the other parts of the process.

4.2 Tuning the Generic Interactive System Architecture

The right-hand side of the diagram (labeled tuning) corresponds to the tuning-on-demand part of the proposed approach while the bottom part focuses on the implementation aspects. These three main phases have been highlighted using gray boxes with dashed lines in Fig. 4. The tuning-on-demand step refines the diagram by concretizing each component of MIODMIT making explicit (in the diagram):

- where software parts (e.g. API; libraries…) provided by the input device’s manufacturer are distributed in the architecture,
- if existing code has already been produced where it has to be distributed in the architecture,
- which component have to be coded from scratch.

It is important to note that due to an absence of standards, provided software packages often require to split or merge functionalities in order to fit in the structure of the generic architecture. For instance, the Leap Motion is provided with three software packages: the driver (for a dedicated OS), the library (making it possible to exploit the driver on a dedicated OS and integrating C and C++ API), and a wrapper for high-level programming language (e.g. Java). The driver corresponds to the component “Driver instance” in MIODMIT while the functions in the Leap Motion library cover the “virtual device” component and several interaction transducers (e.g. detection of a

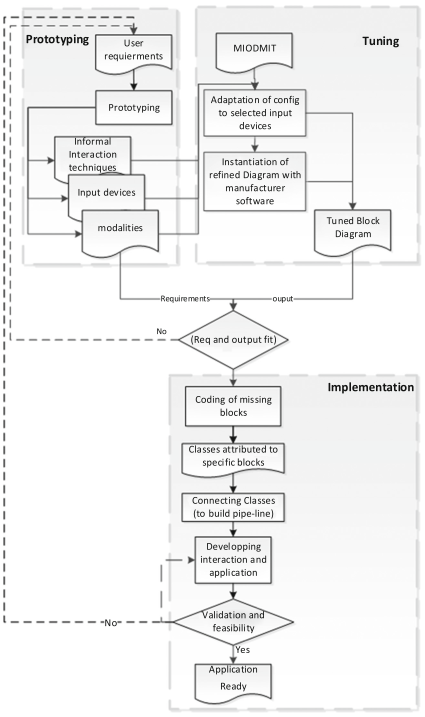

Prototyping
User requierments
Prototyping
Informal Interaction techniques
Input devices
modalities
MIODMIT
Tuning
Adaptation of config to selected input devices
Instantiation of refined Diagram with manufacturer software
Tuned Block Diagram
Requirements
ouput
No
(Req and output fit)
Implementation
Coding of missing blocks
Classes attributed to specific blocks
Connecting Classes (to build pipe-line)
Developping interaction and application
Validation and feasibility
No
Yes
Application Ready

Fig. 4. Process of tuning-on-demand

circle called CircleGesture) located in the "Global Interaction Techniques" component. The wrapper provides functions for all the other components of the input flow but only covers a very limited set of functionalities. The provided set of interaction techniques is basic and has to be extended according to the expected use of the Leap Motion in the application. This is a clever design choice made by Motion manufacturers to allow direct exploitation of the Leap within an application by providing standard interaction techniques but making it also possible to easily extend this set according to the designers' needs.

4.3 Illustration of the Process with the Java Application Example

The result of the application of the tune-on-demand process described above on the Java Application illustrative example is presented in Fig. 5. The Figure can be split in three sections. The left-hand side represent the physical input and output devices used with the Java Application. The center represents the MIODMIT components that are taken care of by the operating system (surrounded by a grey box named "Black Box: OS Windows"). While no access to the Windows source code is given, it remains possible to describe the OS behavior using MIODMIT. Indeed, drivers for a keyboard, mouse and screen as well as part of the rendering system are an integral part of modern OS. The OS merges input and output aspects at hardware level and thus only one component handling both input and output devices drivers is represented. The input and output chains components are merged handling both abstract input abstraction, immediate feedback (position of the mouse cursor on the screen) and graphical rendering (related to the presentation part of the widgets, e.g. display of items in the list box). The window manager of the OS handles picking function (identification of the widgets which are recipients of user input) thus merging input and output chain managers. The right-hand side describe the four components from the MIODMIT architecture that have to be implemented.

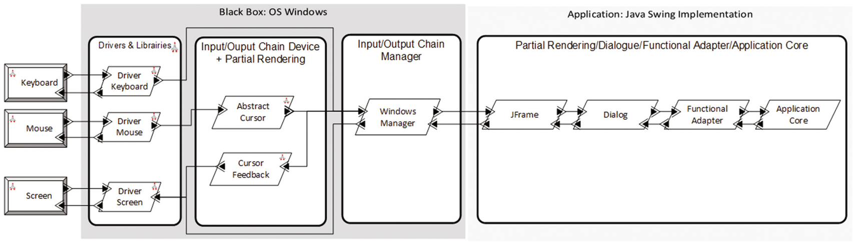


Fig. 5. MIODMIT tuned for Java Application: most of the components of the architecture are integrated within the operating system (especially management input and output devices).

5 A Real World Case Study: A Weather Radar

This case study demonstrates a more complex tuning of the generic interactive system architecture and addresses the issues of integration of input devices as well as the possibility to reconfigure the interaction techniques dynamically. This case study comes from the field of aeronautical critical systems and thus must follow development processes such as the DO178C [12] and certification specifications as defined in CS-25 [11]. In the context of this paper these standards make it impossible to use in critical applications software components for which the code is not available. This prevents using Operating Systems offering integrated handling of devices, drivers… as was the case in the previous example. Indeed, every component of the architecture must be specified and developed from scratch and may be subject to inspection by the certification authorities.

The case study corresponds to a subset of an envisioned weather radar system of civil aircraft providing atmospheric data to the flying crew. This weather radar is controlled by a set of input devices (allowing input from the flying crew) and the processed information is graphically rendered on a computer screen in the cockpit (usually called Navigation Display). This application uses colors and shapes to present information such as dimensions, distance and density of clouds (as visible on the bottom of Fig. 6).

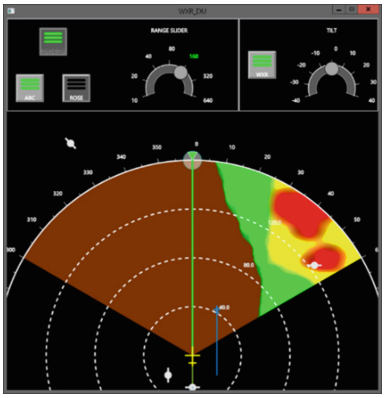


Fig. 6. User interface of the weather radar application

1. The central lower part of the interface is composed of a map and a custom selector:
   - The map displays the aircraft's position and current heading (small numbers from 0 to 360 on the outside circle) as nearby traffic (small blue plane near 40.0 number on the inner circle) and weather information (colored zone on the right-hand side of the image).
   - The custom selector (the small circle at the edge of the biggest circle) controls the heading of the aircraft. When modified, it modifies the information presented in the map display.
2. The upper left part of the interface is composed of three toggle buttons and a custom discrete slider. They provide the following controls:
   - The "HEADING" toggle button (top) controls the heading validation and triggers heading changes,
   - The "ARC" and "ROSE" toggle buttons control the two mutually exclusive display modes of the navigation display,
   - The custom slider (labeled "RANGE SLIDER") defines the zoom level of the navigation display in nautical miles (10 nm, 20 nm…). Current selection is 160 nm.
3. The right upper part of the interface of Fig. 6 called Weather radar control is composed of two widgets:
   - A toggle button "WXR" control the weather visibility on the navigation display,
   - A custom discrete slider controls the weather radar orientation.

5.1 Informal Description (Prototyping Step of the Process)

The informal description provided below is representative of the potential use of several redundant modalities for such application. It is important to note that design aspects of this application is beyond the scope of this paper both in terms of usability and operational validity. In the proposed case study, operators are able to use multiple input devices, modalities and interaction techniques:

- **Input devices:** interaction can take place using a KCCU (Keyboard Cursor Control Unit) which blends a graphical designator (a track-ball) and a keyboard. Two KCCUs are available in the cockpit (one for each pilot) thus enabling parallel interaction with two mice. Such interactions and input devices are available in most recent large civil aircrafts e.g. Airbus A380, A350 and Boeing 787. In the case study, it is also possible to interact in a tactile way using the multi-touch screen presented in Fig. 6.
- **Interaction techniques and modalities:** on top of these input devices, interaction can take place in various ways. Using the multi-touch screen, operators can perform "Flick 2 fingers", "Tap", "Tap long" and "Drag" which are global interaction techniques (i.e. they can be performed everywhere on the screen). Mice are used for triggering events on the WIMP interactors while multimodal events (e.g. "Flick 2 fingers") are assigned (as defined in CARE properties [9]) to the touch screen. In case of a touch screen failure, the mice can be used to trigger high level events previously devoted to the touch screen. In that case the application must be able to switch from one configuration to another. While multi-user interactions with two mice for triggering equivalent multi-touch interactions might be cumbersome, guaranteeing the possibility of events triggering in presence of faulty touch devices was a requirement.

5.2 Overview of the Tuned Generic Interactive System Architecture

According to the process in Fig. 4, the first step, from the prototyping phase, is to refine MIODMIT by specifying all the components. During this refinement, it is important to make explicit where the software provided by the input devices manufacturers is located in the diagram. It is important to note that currently, software packages provided by manufacturers often require splitting or merging in order to fit in the structure of the architecture.

As for the description of MIODMIT in previous sections, Fig. 7 presents an overview of the architecture tuned for the weather radar case study.

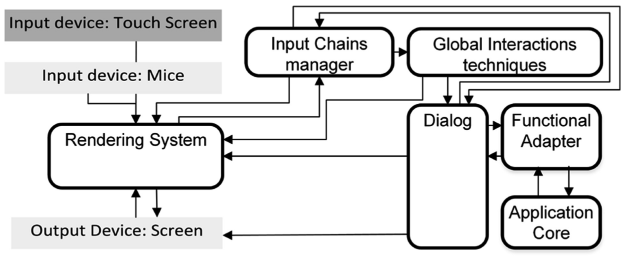


Fig. 7. MIODMIT tuned for the Weather Radar Application

The gray boxes on the left-hand side of the figure correspond to the input and output devices available in the case study. These components will be detailed further in the following sections. The input devices provide input to the "Input chains manager" component. The "input chains manager" component is able to switch between two predetermined input configurations:

- The normal one where touch and mice are available,
- The degraded configuration (resulting from a loss of tactile functionality). This behavior is similar to the one proposed in [27] where reconfiguration was performed at the interaction technique level for keyboards and mice.

As there is only one output device in this case study, there cannot be several output configurations. The "output configuration manager" component is therefore not necessary.

The "Global Interaction Techniques" component contains several transducers that produce high-level events such as "flick-2-fingers", "Combined-Click", "Tap", "Drag", etc. Part of the behavior of those transducers consists of fusing input from multiple input devices thus implementing functionalities of fusion engines. As the weather radar application is a real case study in aeronautics, the behavior of the "Core Application", "Functional Adapter", "Dialog" and a part of the "rendering system" has already been coded. The process of tuning the architecture to include touch and mice interactions does not deeply impact the existing application. Those modifications mainly consist of ensuring that the components connect (plug) and function altogether (play).

The remainder of the "rendering system" concerns mostly the immediate feedback that has to be linked to the two input chains and more particularly, to the "abstract cursor" from the mice input chain and to the "finger" from the touch input chain. These aspects, which are at the center of the contribution, are detailed in the following section.

5.3 Application to the Case Study: Tuning MIODMIT

Input Device Type 1: Touch Screen
Figure 8 presents the tuning of the gray box "Input Device Type 1" from Fig. 3 for a touch-screen device. Adding a touch screen device requires a touch driver (see [13]). All the components within the "Input Chain Touch Screen" are within the Java Virtual Machine (JVM) via the use of a dedicated Java library (JavaFX). These events are retrieved by the "virtual screen" component while the "Touch Screen Manager" instantiates the various logical input devices (fingers in this case) one each time a finger is detected. The "Touch Screen manager" link Fingers events and data to the registered interaction transducers (within the "Global Interaction Techniques" component) as, for instance, the "Flick 2 Fingers" one detecting the eponym interaction.

Input Device Type 2: Two Mice
Even though the mouse is a standard input device, as we use multi-mice interactions, we cannot use drivers provided natively by the operating system. The data from mice is accessed by having a thread polling the JInput library information (e.g. JInput.dll for

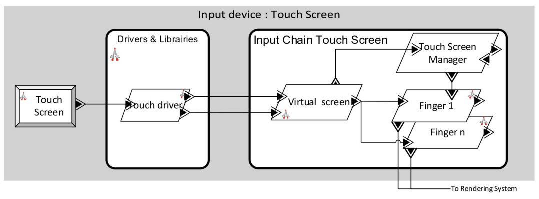


Fig. 8. Tuning MIODMIT for a touch-screen device

windows). The two mice are then handled by the "mice manager" (Fig. 9) that instantiates two virtual mice and two abstraction of cursors. The "mice manager" links these cursors to "interaction transducers" within "Global Interaction Techniques" that recognize high level events such as Click, DoubleClick, etc. and possibly multimodal ones such as combined clicks as defined in [1].

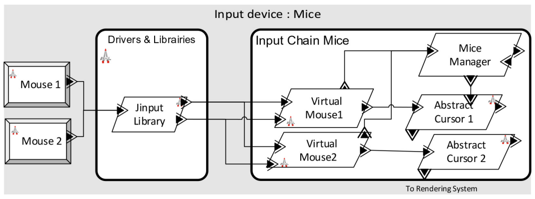


Fig. 9. Tuning MIODMIT for two mice

Output Device: Screen

The management of the screen is straightforward as there is no multiplicity of devices. As there are no redundancies of output modalities in this case study, there is no need for an "output chain manager" component (see Fig. 10).

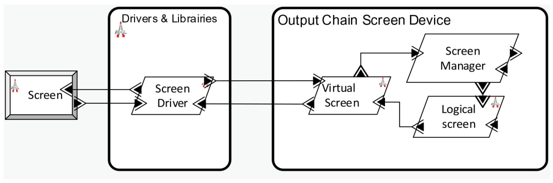


Fig. 10. Tuning MIODMIT for the screen device

5.4 Application to the Case Study: Implementation

Following the process in Fig. 4 after MIODMIT tuning, each remaining component has to be implemented. Implementation concerns the definition of the behavior of each component of the tuned architecture. Such implementation can be done using different programming languages being formal or not. In previous work ICO-based descriptions were used for some of the components (e.g. “Global Interaction Technique” component [21], “Dialogue” [16] component…). Work such as [13] has provided C implementation of all the components of the “Input Chain Kinect”. In section “Tune on Demand Process” we have detailed the various implementation steps that are thus not duplicated here as they are generic to every type of application.

5.5 Application to the Case Study: Adding a Device

One of the most important aspects of MIODMIT is its modularity providing flexibility and modifiability to the applications designed. This section highlights the modifications to be made when an additional input device is added, here a Leap Motion (see Fig. 11). Once the device is chosen, two processes can be done in parallel. The first one is to prototype the interaction using the new device (design, evaluation… etc.). The other is to tune MIODMIT for the chosen device and integrate the tuning within the existing application. In the following, we describe how to add a Leap Motion hand gestures tracker as well as corresponding gesture-based commands to our case study.

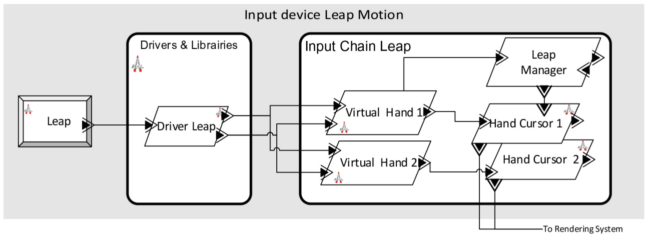


Fig. 11. Tuning MIOMIT for a Leap Motion Device

The Leap Motion is provided with three software packages as explained in Sect. 4.2.

While the wrapper provides functions for several components of the input chain (namely “Virtual Hands”, “Leap Manager”, “Global Interaction Technique”) it only covers a very limited set of expected functionalities for instance only basic interactions (e.g. KeyTap or CircularGesture) are recognized (in the “Global Interaction Technique”), partial transducer for the cursor is provided…

As the case study uses a non-provided interaction technique named Hand-Flick (corresponding to a mid-air “Flick 2 Fingers”) the “Global Interaction Technique” component has to be programmed exploiting the functions of the API. Adding this new device as an equivalent modality does not impact the rest of the implementation as long as it provides the same high level events.

6 Conclusion and Perspectives

This paper presented a generic interactive system architecture and its associated tuning process for the engineering of interactive systems. It addresses the issue of the complexity of engineering current interactive systems integrating non-standard input and output devices and offering multimodal interactions. Both hardware and software aspects are described within a single framework.

The generic interactive system architecture makes explicit the relationships between input devices and interaction techniques. It also makes explicit how such elements are related to implementation considerations involving various software entities such as device drivers, transducers, toolkits and APIs. As such, the integration of novel interaction techniques such as gesture interaction is simplified and better structured, and can be tuned depending on the needs of specific applications. The illustrated development process presented helps demonstrating how the MIODMIT generic interactive system architecture can be applied. An example in the field of critical systems, in our case a weather radar panel for civilian aircrafts, shows a real world application of our approach.

The proposed approach brings multiple benefits including the division of complex interactive systems into generic components loosely coupled and highly coherent thus enforcing the locality of modifications. It also brings research work achieved in the area of critical interactive systems (such as self-checking interactors 35 and reconfigurations) to the broader world of mainstream interactive multimodal systems.

Due to its white box principle (each component of the architecture contributes to the processing of input and the production of output) the approach is particularly suitable for interactive critical systems where each component has to be auditable. However, as demonstrated by the classical interactive application example it is also applicable to the engineering of more mainstream interactive systems. The only difference is that some components are directly managed by the programming environment (Java and Java VM) or the operating system on which they are executed.

References

1. Accot, J., Chatty, S., Palanque, P.: A formal description of low level interaction and its application to multimodal interactive systems. In: Bodart, F., Vanderdonckt, J. (eds.) Design, Specification and Verification of Interactive Systems. Eurographics, pp. 92–104. Springer, Heidelberg (1996). https://doi.org/10.1007/978-3-7091-7491-3_5
2. Accot, J., Chatty, S., Maury, S., Palanque, P.: Formal transducers: models of devices and building bricks for the design of highly interactive systems. In: Harrison, M.D., Torres, J.C. (eds.) Design, Specification and Verification of Interactive Systems. Eurographics, pp. 143–159. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-7091-6878-3_10
3. Bass, L.: Software Architecture in Practice. Pearson Education India, Gurgaon (2007)
4. Bass, L., et al.: The arch model: Seeheim revisited. In: User Interface Developpers' Workshop (1991)

5. Bastide, R., Navarre, D., Palanque, P., Schyn, A., Dragicevic, P.: A model-based approach for real-time embedded multimodal systems in military aircrafts. In: Proceedings of the 6th International Conference on Multimodal Interfaces (ICMI 2004), pp. 243–250. ACM, New York (2004)
6. Buxton, B.: Developing a Taxonomy of Input, chapter 4. http://www.billbuxton.com/input04.Taxonomies.pdf. Accessed 15 January
7. Buxton, B.: A three state model of graphical input. In: Diaper, D., et al. (eds.) Human-Computer Interaction - INTERACT 1990, pp. 449–456. Elsevier Science Publishers (1990)
8. Campos, J.C., Harrison, M.D.: Formally verifying interactive systems: a review. In: Harrison M.D., Torres J.C. (eds.) Design, Specification and Verification of Interactive Systems. Eurographics, pp. 109–124. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-7091-6878-3_8
9. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.M.: Four easy pieces for assessing the usability of multimodal interaction: the care properties. In: Nordby, K., Helmersen, P., Gilmore, D.J., Arnesen, S.A. (eds.) Human—Computer Interaction. IFIP Advances in Information and Communication Technology, pp. 115–120. Springer, Heidelberg (1995). https://doi.org/10.1007/978-1-5041-2896-4_19
10. Cronel, M.: Une approche pour l'ingénierie des systèmes interactifs critiques multimodaux et multi-utilisateurs: Application à la prochaine génération de cockpit d'aéronefs, thèse de doctorat, Université Paul Sabatier, octobre 2017
11. CS-25 - Amendment 17 - Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes. EASA (2015)
12. DO-178C/ED-12C, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA and EUROCAE (2012)
13. Deshayes, R., Palanque, P., Mens, T.: A generic framework for executable gestural interaction models. In: VL/HCC 2013, pp. 35–38 (2013)
14. Feiler, P.H., Gluch, D.P., Hudak, J.J.: The architecture analysis & design language (AADL): An introduction (No. CMU/SEI-2006-TN-011). Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst (2006)
15. Hamon, A., Palanque, P., André, R., Barboni, E., Cronel, M., Navarre, D.: Multi-touch interactions for control and display in interactive cockpits. In: HCI'Aero 2014. ACM DL (2014)
16. Hamon, A., Palanque, P., Silva, J.L., Deleris, Y., Barboni, E.: Formal description of multi-touch interactions. In: 5th ACM SIGCHI EICS, pp. 207–216. ACM (2013)
17. Hansen, T.E., Hourcade, J.P., Virbel, M., Patali, S., Serra, T.: PyMT: a post-WIMP multi-touch user interface toolkit. In: ACM ICITS, pp. 17–24. ACM (2009)
18. Hinckley, K., Jacob, R.J.K., Ware, C., Wobbrock, J., Wigdor, D.: Input/output devices and interaction techniques. In: Computing Handbook, 3rd edn., Chap. 21, pp. 1–54 (2014)
19. Hoste, L., Dumas, B., Signer, B.: Mudra: a unified multimodal interaction framework. In: ICMI 2011, pp. 97–104. ACM (2011)
20. Kammer, D., Keck, M., Freitag, G., Wacker, M.: Taxonomy and overview of multi-touch frameworks: architecture, scope and features. In: Workshop on EPMI (2010)
21. Kraleva, R., Kralev, V.: On model architecture for a children's speech recognition interactive dialog system. In: Proceedings of International Scientific Conference on Mathematics and Natural Sciences (2009). https://arxiv.org/pdf/1605.07733
22. Ladry, J.-F., Navarre, D., Palanque, P.: Formal description techniques to support the design, construction and evaluation of fusion engines for sure (safe, usable, reliable and evolvable) multimodal interfaces. In: ACM ICMI, pp. 185–192 (2009)
23. Lalanne, D., Nigay, L., Palanque, P., Robinson, P., Vanderdonckt, J., Ladry, J.F.: Fusion engines for multimodal input: a survey. In: ICMI, pp. 153–160. ACM (2009)

24. Latoschik, M.E., Reiners, D., Blach, R., Figueroa, P., Dachselt, R.: SEARIS: software engineering and architectures for realtime interactive systems. In: 24th ACM SIGPLAN OOPSLA, pp. 721–722 (2009)
25. Lee, J.S., et al.: A 0.4 V driving multi-touch capacitive sensor with the driving signal frequency set to (n + 0.5) times the inverse of the LCD VCOM noise period. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 682–685 (2014)
26. Ng, A., Lepinski, J., Wigdor, D., Sanders, S., Dietz, P.: Designing for low-latency direct-touch input. In: 25th ACM UIST Conference, pp. 453–464. ACM (2012)
27. Navarre, D., Palanque, P., Basnyat, S.: A formal approach for user interaction reconfiguration of safety critical interactive systems. In: Harrison, Michael D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 373–386. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87698-4_31
28. Nigay, L., Coutaz, J.: Multifeature systems: from HCI properties to software design. In: Proceedings of First International Workshop on Intelligence and Multimodality in Multimedia Interfaces. AAAI Press Publ. (1995)
29. Morris, M., Huang, A., Paepcke, A., Winograd, T.: Cooperative gestures: multi-user gestural interactions for co-located groupware. In: ACM CHI Conference 2006, pp. 1201–1210 (2006)
30. Palanque, P., Bastide, R., Navarre, D., Sy, O.: Computer discretized interaction: from continuous to discrete and back again. In: Workshop on Continuity in Human Computer Interaction, CHI 2000, The Hague (2000)
31. Pfaff, G.E. (ed.): User Interface Management Systems. Springer, Heidelberg (1985). https://doi.org/10.1007/978-3-642-70041-5
32. Rousseau, C., Bellik, Y., Vernier, F.: Multimodal output specification/simulation platform. In: ACM ICMI 2005, pp. 84–91 (2005)
33. SAE-AS5506B: SAE Architecture Analysis and Design Language (AADL), International Society of Automotive Engineers, Warrendale, PA, USA, September 2012
34. Schneegass, S., Alt, F.: SenScreen: a toolkit for supporting sensor-enabled multi-display networks. In: Gehring, S. (ed.) Proceedings of the International Symposium on Pervasive Displays (PerDis 2014). ACM, New York (2014). 6 pages
35. Tankeu-Choitatk, A., Navarrek, D., Palanquek, P., Delerisk, Y., Fabrek, J.-C., Fayollask, C.: Self-checking components for dependable interactive cockpits using formal description techniques. In: PRDC 2011, pp. 164–173 (2011)
36. Vu, T., et al.: Distinguishing users with capacitive touch communication. In: Mobicom 2012, pp. 197–208. ACM (2012)
37. Echtler, F., Klinker, G.: A multitouch software architecture. In: Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges (NordiCHI 2008), pp. 463–466. ACM, New York (2008)
38. Vennelakanti, R., Dey, P., Shekhawat, A., Pisupati, P.: The picture says it all!: Multimodal interactions and interaction metadata. In: Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI 2011), pp. 89–96. ACM, New York (2011)
39. Kousidis, S., Kennington, C., Baumann, T., Buschmeier, H., Stefan, K., Schlangen, D.: A multimodal in-car dialogue system that tracks the driver's attention. In: Proceedings of the 16th International Conference on Multimodal Interaction (ICMI 2014), pp. 26–33. ACM, New York (2014)