How to Achieve Design for All:

"List", "Focus" and "Multimodality" as Minimal Requirements

Denis Chêne(✉), Éric Petit, and Sophie Zijp-Rouzier

Orange Labs, 28 Ch du Vieux Chêne, 38243 Meylan, France
{denis.chene,eric.petit,sophie.zijprouzier}@orange.com

Abstract. This paper introduces basic principles for "design for all". List, focus, multimodal feedbacks and several action means are the keys. A generic touch-based interface component (MenuDfA) was designed, offering hierarchical navigation with continuous and secure gestures along with a high level of flexibility. A novice user profile was set. A comparative test to a classical tactile interface shows that the optimized interface for novice users solves difficulties, but generates also some issues. These latter have been easily resolved through parameters adjustment.

Keywords: Design for all methods techniques and tools · Access to mobile interaction · Design for all best practice · Accessibility guidelines · Universal design methodology · Touch-based interaction · Continuous manipulation · Menudfa · Handheld device

1 Introduction

Reverse of the traditional way of adjusting afterward standard interfaces to diversity (the Assistive Technologies way), "Design for all" methodology aims at providing continuity between accessibility features and standard ones [2]. But user interfaces (UI) are mainly visually oriented and rely on absolute positioning, whereas some accessibility profiles need exclusively relative positioning (pointless interface). Closing this gap is both a social and technical challenge with the objective of designing interfaces for all kind of users in different contexts of use.

The interface elements organization, types of menus, kinds of controls, and in case of a touch-based interface, kinds of gestural commands, are all the questions this paper tries to cater to. First we explore user needs and basic user interface requirements to reach design for all purpose. By extracting common interaction patterns we underline that simple "List" (or "Menu") and "Focus" are prerequisite elements to achieve universal approach, along with minimal commands and multimodal feedbacks. On this basis, a user interface for hand-held device has been designed. His architecture is described. An optimized profile dedicated to novice users is defined. It has been assessed in a controlled user study. Experimental results highlight some useful improvements that result in an enhanced profile well suited for novice users. Discussion follows on

M. Antona and C. Stephanidis (Eds.): UAHCI 2016, Part II, LNCS 9738, pp. 117–128, 2016.
DOI: 10.1007/978-3-319-40244-4_12

how core interaction patterns and “list” and “focus” elements are likely useful for other interaction profiles.

2 User Needs Overview

2.1 Design for All Methodology: First Resolve Borderline Issues

Norman’s interaction model [9] details human-machine interaction loop into two main phases, execution and evaluation. Physical, sensory and cognitive affordances [18] are the first basic levels of the interaction loop. They all need to be adjusted to each kind of user, as such content themselves.

Human Centered Design approach requires defining user needs and context of use [4, 14] and to identify common interaction patterns that may be extracted from user needs. In his dimensions dedicated to set priorities, Vanderheiden [17] defines a level 1 with the condition to be compliant to each kind of interaction specificities. But some interaction situations are easier to solve than others. For instance, resolving a low zoom level user need is easier to achieve, as it is not so far away from a standard interface, than resolving a very high zoom level that is significantly different from standard interaction schema. This gap from standard interaction has to be taken into account because it has great consequences on design effort.

This paper argues for coping first with interaction logics that are far away from the standard one. Those interaction logics should be the first design starting point. Indeed, when they are lately taken into account in a subsequent design, it’s really hard, costly and times consuming to treat them. Those interaction specificities are well known and clearly detailed in [5]. They consist of sequential interaction, pointless interaction, secure navigation, minimal commands, refined design, and various mono-modalities. Afterwards, other interaction logics should be easily handled.

2.2 Interaction Basic Components to Design for All

Rendering for All. On the interaction level, users need that interface elements are rendered in many suitable ways. It addresses texts, colors, images, tactile, sounds and type of voices. These elements should be available through a single modality, and also through multimodality. They should be sequentially available or rendered in parallel way as some users are restraint to sequential perception (audio, tactile, restricted visual display) or can use parallel one (mainly vision). There are also some needs about rendering information rhythm. Indeed user may need slow down information or accelerated one. All this range of rendering aspects should be handled by the user interface.

Actions for All. Regarding the action part, some users need step by step actions (sequential actions, and/or selection separated from validation) or direct access (shortcuts, selection-validation fusion). But some of them have only monotask capacity or some other are able to deal with multitasks. Some of them need totally pointless interface, and other need to point at, which implies visual-motor or audio-motor coordination (deictic, head or finger). Finally there may be some needs of “uncolocated” actions (to act somewhere whereas the selected object is elsewhere) or colocated ones (to act where the selected object is). All this range of action means has to be available.

Adaptability. As seen previously, at the interface level those actions and rendering effects need to be adjustable from preset profiles to personalized one. Adjustment of the perception and action means should be available at the interaction level (a blind user will need a slow synthesis at the beginning but will quickly ask for a speedy one). They also should be available at the task/functional level (a novice user needs less available functions than an expert user).

Contents for All. Content needs address linguistic and cultural diversity (written and oral languages, signed languages). This is valuable for text but also for images. Indeed some images or icons are relevant in a certain context but may be not in a different culture [13]. Some contents need to be slightly reworded or simplified (for novice user, or in case of other cognitive constraints). Again, users need sometimes augmented contents (audio description for the blind or close caption for the deaf). All this range of contents should be available for each kind of user.
In the following part a generic navigation component is presented [10, 11]. This component aims at catering previous requirements of rendering, actions, adaptability, and contents for all. It is a set of consistent interaction techniques which demonstrates effective alternatives to visual-centric interface designs on mobile devices.

3 Implementation of a Generic Component for Touchbased Interfaces

In most graphical user interfaces (GUI), there are too many objects and it's really a mess for novice or disabled user to deal with them. Several innovative approaches about how to design "menus" were explored [3, 8, 12]. But those menus are still limited to a single usage and are not able to embrace the diversity of context of use. Moreover as they often rely on pointing, they generate accessibility limitations. Those complex objects are difficult to adapt to user or context limitations. But they all have in common to offer to make a choice among several proposals. Reducing interaction to this basic task, leads us to consider a string of elements usually displayed as a list.

3.1 Interface Objects Simplification to a Vertical List

A list is a basic interface element that enables to make choices. It can be displayed globally, or sequentially, it has a well-known spatial and linear organization, it can be adjusted to each modality, and can be easily warped to many visual adjustments. It's a generic container for colors, texts, images. It is suited to reading optimization provided that it's displayed vertically. A succession of tabs is also a list, but horizontally displayed. Those are not really optimized for rapid reading and are not adapted in case of numerous items. These are the reasons why vertical lists are the primary requisite. Starting from a *vertical list* doesn't prevent the use of more complex components as they can be recomposed afterwards for other interaction profiles. Thereby, a mosaic can be considered as a spatially spread list that still can be browsed sequentially through a Z logical path, from left to right. So starting from list and expand it to mosaic (more adapted to pointing interfaces) is still feasible. Even a slider can be considered as a list of items. Actually, most complex graphical components can be simplified in a listview, intrinsically sequentially browsable. This approach centered on the list element involves that the whole application is a cascading hierarchy of list elements.

3.2 Navigation Commands and Focus

To navigate easily through this hierarchy and inside each list is an important point for accessibility. Indeed user needs to learn the interface without the risk of missed validations. Secure navigation can be achieved through selection and validation dissociation. And pointless navigation can be achieved with focus manipulation. If a focus is materialized on the interface (visually, auditively and tactilely) and manipulated by the user, then selecting an item can be differentiated from validating it, with or without the need to point at it. Navigation through nested lists involves only four commands even if other ones may be added for other profiles (as shortcut commands). It can be achieved through Next-Previous navigation inside the current list, and through going In or Out hierarchy levels. Those commands can be achieved through many types of different gestures or other controls (as vocal), but the major point is that only four navigational commands are mandatory. Another command is useful for some users that don't have immediate feedback about the ongoing focused item. This command is called "get info" command. It enables to inform the user about the current focused item or the UI state. Rendering effect of current manipulation is expressed visually with a focus. As for vertical list rendering, it may be displayed through vocal, audio, visual, and tactile modalities.

3.3 MenuDfA Component

MenuDfA is a generic navigation component for touch-based interfaces under Android OS which implements principles mentioned above along with several interaction profiles optimized to each kind of user. So, MenuDfA focused on overcoming difficulties faced by part of the population when operating touch-based interfaces, especially smartphones [6, 7, 15]. In particular, standard pointing gestures, such as "tap" pose several usability problems: they require high positioning accuracy and temporal control and imply strong visual-motor coordination. Beyond being prone to manipulation error, they are not suited to all users (e.g. a person with low vision). This is exacerbated by the fact that standard touch interaction does not separate selection and validation: you just brush the screen and the item underneath finger is triggered. In fact, there is no means to safe explore the interface before using its features. In addition, modern touchscreen interfaces embed various interface elements (e.g. icons, tabs, grids, vertical and horizontal lists, widgets, …) which provide rich but heterogeneous layouts and behaviors. The consequence is twofold, on one hand, this generates cognitive load for the user, and on the other hand, this prevents incorporating appropriate navigation mechanisms for achieving usability for all.

MenuDfA component combines following principles. Firstly, it organizes the various items of the intended application in the form of a set of hierarchical lists where the items of each list are displayed vertically. Secondly, MenuDfA provides a sequential access to the interface elements, as well as a direct access. In both cases selection and validation may be set to two separated operations. Interaction mode can be set to pointless or pointed at, and thus allowing uncolocated or colocated actions. Therefore it covers a large diversity of user needs and situations.

Sequential navigation relies on the use of a selection focus that the user manipulates by means of slide gestures. Thus, the user can navigate the entire application just by moving the focus in the four directions: Vertical displacements for moving through the on screen items, and horizontal displacements for moving through the hierarchy of lists. The mapping of the gestures is chosen to be aligned with the displacement of the focus (up, down, right and left). Note, that contrary to classical touch screen interfaces; here the gestural interaction doesn't rely anymore on absolute positioning as all the gestures make use of relative positioning (they can begin anywhere on the screen). Moreover, gesture can be discrete or continuous. A discrete mapping means that, between the moment the finger touch the screen and the moment it leaves the surface, a gesture triggers a unique command (e.g. move the focus to the next item). This is the most current in sequential interfaces. Continuous mapping consists in a continuous gesture control of the focus. In that case, the detection criterion relies on the distance traveled by the finger during gesture, combined with the movement direction. This implies to define a special transfer function (similar to a CD-gain [1]) between the motor space and the visual space. In that case, appropriate feedbacks have to be designed in order to ensure a smooth coupling between perception and action, as implemented in MenuDfA.

Direct access to interface elements (classical way), is still available with MenuDfA in order to allow pointing when it is more appropriate for the user. So, the user can choose to "Press & Hold" onto the intended item. In that case, validation consists in two phases: firstly selection (i.e. the focus comes underneath finger on the intended item), then validation of the selected item after a given time. It is different from classical interfaces where the both are merged. Note, that here, depending on the setting value of the time duration between selection and validation, these two phases are more or less separated. To summarize, MenuDfA provides separate selection and validation operations both through direct access and sequential access. This feature is then combined with appropriate feedback (e.g. vocal and tactile feedback) to ensure secure exploration of the interface, meaning the possibility for the user to get prior information on a given function before to trigger it. In the case of sequential access it allows furthermore to use "relative positioning", eliminating the need to point at a precise place, thus relaxing constraints on absolute positioning and on visual control. Those properties are fundamental to address usability for all.

The third characteristic of MenuDfA is related to the flexibility of interaction, both in terms of feedback and gestures. Regarding gestures, Table 1. gives the whole set of gestures managed by MenuDfA. In addition, a set of parameters is provided along with these various gestures allowing to finely customize each type of interaction according to the user needs. The next part deals with MenuDfA's architecture and design.

Table 1. Gestures available in MenuDfA component

Vertical navigation (“selection” command) or Information feedback (“get info”)		
Gesture description	**Associated Command**	**Targeted profiles**
↑↓ Vertical slides, with inertia and continuous mapping without colocation constraint	- the focus moves up or down.	- novice - elderly - standard/expert
↑↓ Vertical slides (discrete mapping if fast and continuous if slow), without inertia nor colocation	- the focus moves up or down.	- blind - motor1
● located Press&Hold (direct access)	- Selection (the focus jump underneath finger)	- customized
◉ Tap without colocation constraint (anywhere on the screen)	- get information (vocal, visual tooltip) on the current selection	- low vision, blind - illiterate, cognitive - motor1
◉ Tap without colocation constraint	- the focus moves to the next item	- motor2 - blind
Gestures for horizontal navigation (only validation or back command)		
Gesture description	**Associated command**	**Targeted user profiles**
→ Right slide without colocation constraint	- validation of the selected item.	- standard/expert - blind
← Left slide without colocation constraint	- back to the upper level	- all profiles
● Press&Hold without colocation constraint	- validation of the selected item	- motor1&2 - blind
◉ ◉ Double tap without colocation constraint	- validation of the selected item - zoom in/out	- customized
Combined gestures for vertical and horizontal navigation (selection and validation)		
Gesture description	**Associated command**	**Targeted profiles**
↳ Spatial compound gesture without colocation constraint	- selection, then validation, then selection, then validation	- standard/expert - blind
●→ located Press&Hold then slide Right	- selection (the focus jump underneath finger) then validation	- standard/expert
● located Press with a very short delay (almost Tap)	- selection and immediate validation	- near-usual tactile interface
● located Press&Hold with a medium delay	- selection then validation	- standard/expert - novice
● located Press&Hold with a long delay	- selection then validation	- illiterate - cognitive
↺ Symbolic gesture without colocation constraint	- launch of a specific command	- expert

3.4 Architecture

Design for all in the field of HCI results in complex software systems, primarily because it relies on a global design approach which must cope with various human and technical factors. Among these factors, we find: multimodal rendering (visual, audio, tactile), navigation logic, gesture recognition, and customization capabilities. When considered together it poses a great technical concern. For example, making a simple sequential, circular and re-sizable item list with standard API (e.g. Android 5.0) is already a problem in itself because such mechanisms are not originally supplied. Likewise, manipulating a selection focus with slide gestures is not common (and thus available) and requires implementing a dynamic transfer function along with a gesture recognition engine. And also, designing a flexible event-driven program that allows hot-changing the interaction mode is not so common. That's why MenuDfA's software architecture involves several modules and frameworks. In particular, it consists of the following pieces: a gestural interaction framework named DGIL as a C ++/Java library, an event-driven programming framework named AEvent as a Java library, a graphical sequential list view named DfAList as a Java Class, a graphical "pager" named DfAPager as a Java Class to handle the lists hierarchy and a central module based on a Model-View-Controller architecture (MVC).

In this schema, DGIL provides a syntactic description of the gesture during its articulation, in the form of a sequence of gestural primitives. They chunk the gestural dialog into meaningful units, allowing a one-to-one correspondence between desired operations and gesture. For example, in a particular mode of DGIL, a "Press & Hold" gesture gives rise to the following sequence:

```
PRESS → SHORT_PRESS → LONG_PRESS → RELEASE → END_GESTURE
```

A vertical slide leads to a richer one:

```
PRESS → SHORT_PRESS → START_MOVE →EARLY_MOVE_DOWN
(1    %,2    %,4    %,9    %,17    %,30    %,53    %,
90 %) →MOVE_DOWN→ EARLY_MOVE_DOWN
(0 %,4 %,13 %,25 %,48 %,73 %)→CANCEL_EARLY_MOVE→END_GESTURE
```

In this example, the `MOVE_DOWN` primitive indicates (and causes) an effective change of item selection. But before it occurs, a series of `EARLY_MOVE_DOWN` primitives is emitted. This series of events is in charge of continuously moving the focus during the transient phase. This primitive has a parameter which indicates the percentage of distance to travel before triggering the change. In fact, this parameter is used to implement the transfer function that drives the motion rendering of the focus. More specifically it ensures that it works at all scales, that the CD-gain is correct (equal to 1 on average), and that the motion direction is always taken into account. Actually, within the software architecture, DGIL's primitives are converted into events of type AEvent.

Regarding AEvent, firstly it provides a means to implement flexible coupling between events and commands, allowing changing the gesture mapping at run-time. Secondly, it handles memory persistence of the parameters, facilitating customization. With AEvent framework, one can create "Handlers" that contain each a collection of event-command pairs, and then create a "Profile" that gathers several handlers which determine a particular user profile. Concerning the central module, it integrates DGIL-AEvent with an MVC design pattern, as a key point to reduce and master complexity in order to handle multi-profile.

4 Test of the Profile for "Novice User"

Even if the different profiles were built upon user needs analysis, a user centered design methodology requires checking continually that needs are still met. A controlled study was conducted to evaluate novice and expert tactile interaction modes. Those profiles where defined from studies underlying elderly people difficulties with tactile interfaces [7] and also the opportunity to use thumb interaction [6, 16] knowing that single-handed device is a very current context of use.

An interaction profile for novice users (N interface) was designed. Touch-based interaction in the N interface was designed to cater to novice constraints. As they need time to ask for information and to feel confident, selection time was separated from validation time. Thus, focus manipulation (selection) is followed by validation. This means that standard tap cannot be used and must be replaced by continuous manipulation: either using "Press & Hold" with a long delay of 800 ms (●) or using vertical and horizontal slides (↑↓ ←→). As they need support for pointing with inaccuracy then large items were designed and pointing with tolerance zone was added. As they need access to any screen area through the thumb, then selection and validation were available without colocation constraint. Thus it means that the user can manipulate the focus while pointing at another location (see Fig. 1) as well as on the same (Fig. 2).

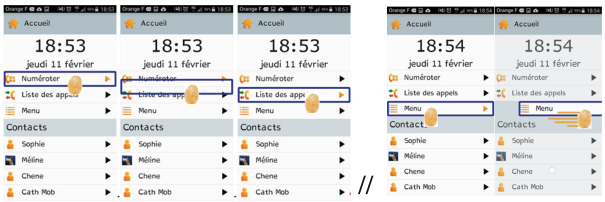

Fig. 1. Colocated continuous Up-Down focus manipulation (selection) and colocated Right gesture (validation).

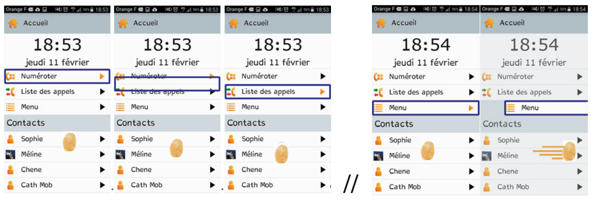

Fig. 2. Uncolocated continuous Up-Down focus manipulation (selection) and uncolocated Right gesture (validation).

To sum up, in the N interface, selection can be performed either by a continuous vertical slide or by a Press & Hold (under 800 ms). In the latter case, selection occurs during the first stage of the press and gives rise to the vocalization of the current focused item.

Validation follows as a second stage only if the user holds his press after 800 ms (●). Otherwise, the user can validate with a slide towards right (→). Similarly, the back command is a continuous gesture to the Left, also without colocation constraint (←), which implies that no back button is displayed on the N interface. When an uncolocated Tap ◉ is executed (anywhere on the screen), then it repeats vocalization of the focused item. It enables user to ask for information as much as he wishes. In order to perform a meaningful comparison of the two tactile interaction modes, an equivalent Classical tactile interface was also designed (C interface). On the C interface only the standard Tap is available to achieve selection/validation in a unique phase. Hence, it implies colocation constraint. Back action is available on the screen with a back button on the top left part of the screen. C and N interfaces were designed with the same functionalities enabling making a phone call from number dialing, or from contact selection, or from call log history. Contact list is available and contacts are editable. The only visual differences between these two interfaces are the existence in the N interface of a focus element (a rectangle), some visual feedbacks associated to continuous manipulation, and the lack of any back button.

4.1 Comparative User Test

This novel interface optimized for novice user (N) was compared to its classical version (C). Experimental plan was the following: S10 < I2*T5 >. N and C interfaces (I factor) were both used one after the other by 10 elderly touchscreen expert users and 10 elderly touchscreen novice users (S). Users were from 70 years old to 74 years old. Novice users had never touched any touchscreen devices. Expert users were used to use a tactile mobile phone. 5 tasks (T) had to be carried out on each interface. For one of them the user was asked to operate device with only one hand. Experimentation duration was one and half an hour. N and C interfaces order was counterbalanced. And before each of them some explanations about the tactile interaction mode were given to user. For the N interface the explanation was: «you have to move the focus in order to select the item you want. Then validate it doing a right gesture». If the user had some difficulties to execute this pre-test, then it was added: «you can also do a long press to validate». For the C interface the explanation was: «you have to tap on the screen in order to validate the item you want». Time, task achievement, and errors were recorded in log files, and on videotapes. Errors were coded accordingly to the type of generated errors. In the N condition 9 types of errors were coded. In the C condition 10 types of errors were coded. They were gathered into 2 categories: pointing inaccuracies, focus adjustment difficulties, non-actable items (*common pointing* errors named *CP*) and some specific N condition errors about *laterality confusion* (very few) and *missed validations due to uncolocation (*named *MVU).*

Results. A Wilcoxon signed ranks test shows that in N condition novice global errors were as numbered as in C condition (novice N errors M = 1.75, SD = 2.37 and C errors M = 1.78, SD = 2.34) whatever was the task. This was not expected as in the N condition uncolocation and Selection-Validation dissociation were here to solve gesture inaccuracies and careless mistakes. Further analysis shows that this is due to the specific errors that appear in N condition (*MVU)*. Indeed, novice users are constantly stuck to direct pointing. It was awaited for expert users of standard tactile pointing interfaces as they are used to directly point at items, but it wasn't for novice users. Actually, uncolocated validation makes the user to validate another item without realizing that it is not the focused one. The Right gesture without colocation constraint (→) is a trap for novice and expert users until they have acquired that the focus can be used in an uncolocated manner. On the contrary the Left gesture (←) doesn't generate error at all as it is never dependent on location (it just goes back to the previous level).

Details results show that if one focuses on *CP* errors only, then the N condition fulfills its goal. Indeed Mann-Whitney test shows, that *CP* errors (gesture inaccuracies and pointing located errors) are significantly fewer in number ($z = 2.42$, $p < .01$) for novice in N condition (*CP* errors M = 0.31, SD = 0.77) than for novice in C condition (*CP* errors M = 1.78, SD = 2.34). To sum up, these usual errors due to short press gestures and tight coupling between validation and selection (classical Tap) are mainly done on C prototype but very few on N prototype, as expected (Fig. 3). Hence, N condition with its "Selection & Validation" dissociation and its numerous facilitations has served its purpose to substantially reduce expected usual errors.

One of the tasks asked users to operate one-handed with the thumb. Similarly, if one focuses on *CP* errors, results show that in N condition novice users (M = 0.5, SD = 0.93) are doing quite as much errors as expert users (0), i.e. almost zero. In thumb interaction mode, N condition made the glass ceiling disappear between novice and expert users. It is not the case in the C condition where novice users (M = 3.75, SD = 3.54) are doing significantly more errors ($z = 2.383$, $p < .02$) than expert users (M = 0.625, SD = 1.06). Usual pointing interface (C) is still a trap for novice users, and also for expert users that make errors in such one-handed situation. On the contrary, N condition is well suited both for novice and expert users in a one-handed situation, considering *CP* errors.

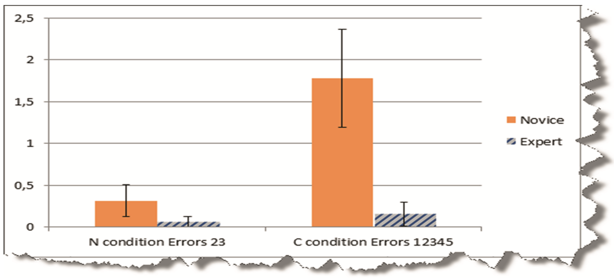

Fig. 3. Mean of expected errors (without *MVU* errors) in N condition vs C condition

Novice Profile Adjustment. According to this study, we have adjusted the novice interaction profile in disabling the Right gesture validation, which was the cause of missed validation due to uncolocation errors (*MVU*). This was done easily just by setting an option, taking advantage of our flexible architecture. Other adjustments are planned in order to improve display specifications (size, contrast, and legible "Accessible DfA" font[1]), audios and tactile feedbacks synchronization.

5 Discussion and Conclusion

Through this experimental study, an innovative touchscreen interaction optimized to novice users was defined, tested and adjusted. Right validation gesture, as source of many uncolocation errors wasn't kept in the final novice profile. Selection-Validation phases dissociation was confirmed as appropriate. Left, Up and Down gestures, and Simple Tap for getting information about the current focused item turned out to be well suited to novice users. Improvement of the novice profile was easily made after the test thanks to the general great flexibility offered by MenuDfA's architecture. A multimodal interface built upon list and focus, enabling continuous commands (gestures, vocals) enables to plainly deal with many profiles. Blind user profile is not so far from the novice one. The only difference is that they don't need colocated Press & Hold but need uncolocated Right gesture that is totally suitable in their case. Some motor impaired users using the back of their hand (fingers up) may be glad to use it in the same way. For them, Left and Right gestures may be an easy way to navigate inside the list hierarchy. Novice profile is also interesting in case of illiterate users because these latter are often novice in technology. The main difference is that they are not able to read the nature of the item, thus they need to ask for information more frequently. For them located Press & Hold with a long delay will certainly be useful. Other cognitive, expert, and low vision profiles are under study.

References

1. Casiez, G., Vogel, D., Balakrishnan, R., Cockburn, A.: The impact of control-display gain on user performance in pointing tasks. In: HCI 2008, vol. 23, pp. 215–250 (2008)
2. Constantine, S.: Adaptive techniques for universal access. User Model. User Adap. Inter. **11**, 159–179 (2001)
3. Francone, J., Bailly, G., Lecolinet, E., Mandran, N., Nigay, L.: Walvet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode. In: ACM AVI (2010)
4. Guide 71. Guide for addressing accessibility in standards. ISO/IEC Guide 71 (2014)
5. ISO/IEC 29138-1: User needs summary. To be published 2017. ISO/IEC/SC35/WG6
6. Karlson, A.K., Bederson, B.B., Contreras-Vidal, J.L.: Understanding single-handed mobile device interaction. In: Lumsden, J. (ed.), Handbook of Research on User Interface Design and Evaluation for Mobile Technologie, pp. 86–101 (2008)

[1] Available at https://github.com/Orange-OpenSource/font-accessible-dfa.

7. Kobayashi, M., Hiyama, A., Miura, T., Asakawa, C., Hirose, M., Ifukube, T.: Elderly user evaluation of mobile touchscreen interactions. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part I. LNCS, vol. 6946, pp. 83–99. Springer, Heidelberg (2011)
8. Lee, D.S., Yoon, W.C.: Quantitative results assessing design issues of selection-supportive menus. Int. J. Ind. Ergon. **33**(1), 41–52 (2004)
9. Norman, D.A.: The design of everyday things. Doubleday, New York, NY (1988)
10. Petit, É., Chêne, D.: MenuDfA: vers une navigation gestuelle tactile conçue pour tous. < hal-01188978 > (2015)
11. Petit, É., Chêne, D.: 27$^{\text{ème}}$ conférence francophone sur l'Interaction Homme-Machine., Oct 2015, Toulouse, France. IHM-2015, pp. d03 (2015)
12. Roudaut, A., Bailly, G., Lecolinet, E., Nigay, L.: Leaf menus: linear menus with stroke shortcuts for small handheld devices. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5726, pp. 616–619. Springer, Heidelberg (2009)
13. Singh, N., Pereira, A.: The culturally customized web site. Ed. Routledge, NY (2012)
14. Song, K., Lee, S.: Mapping user accessibility needs systematically to universal design principles. In: Lee, S., Choo, H., Ha, S., Shin, I.C. (eds.) APCHI 2008. LNCS, vol. 5068, pp. 446–456. Springer, Heidelberg (2008)
15. Trewin, S., Pettick, D.: Physical accessibility of touchscreen smartphones. In: Proceedings of the 15th International ACM SIGACCESS Conference on Computer and Accessibility, vol. 19, pp. 1–8 (2013)
16. Trudeau, M.B., Young, J.G., Jindrich, D.L., Dennerlein, J.T.: Thumb motor performance varies with thumb and wrist posture during single-handed mobile phone use. J. Biomech. **45**, 2349–2354 (2012)
17. Vanderheiden, G.: Fundamental principles and priority setting for universal usability. In: ACM (ed.) CUU 2000 Arlington VA USA (2000)
18. Vermeulen, J., Luyten, K., van den Hoven, E., Coninx, K.: Crossing the bridge over Norman's Gulf of execution: revealing feedforward's true identity. In: CHI 2013, April 27 – May 2, Paris, France (2013)