On Efficiency of Adaptation Algorithms for Mobile Interfaces Navigation

Vlado Glavinic[1], Sandi Ljubic[2], and Mihael Kukec[3]

[1] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR-10000 Zagreb, Croatia
vlado.glavinic@fer.hr
[2] Faculty of Engineering, University of Rijeka, Vukovarska 58, HR-51000 Rijeka, Croatia
sandi.ljubic@riteh.hr
[3] College of Applied Sciences, Jurja Krizanica 33, HR-42000 Varazdin, Croatia
mihael.kukec@velv.hr

Abstract. Many ubiquitous computing systems and applications, including mobile learning ones, can make use of personalization procedures in order to support and improve universal usability. In our previous work, we have created a GUI menu model for mobile device applications, where personalization capabilities are primarily derived from the use of adaptable and adaptive techniques. In this paper we analyze from a theoretical point of view the efficiency of the two adaptation approaches and related algorithms. A task simulation framework has been developed for comparison of static and automatically adapted menus in the mobile application environment. Algorithm functionality is evaluated according to adaptivity effects provided in various menu configurations and within several classes of randomly generated navigation tasks. Simulation results thus obtained support the usage of adaptivity, which provides a valuable improvement in navigation efficiency within menu-based mobile interfaces.

Keywords: personalization, adaptation, algorithmics, m-devices, m-learning.

1 Introduction

Mobile learning (m-Learning), the intersection of online learning and mobile computing, promises the access to applications supporting learning anywhere and anytime, implementing the concepts of universal access [10]. Personal mobile devices and wearable gadgets are presently becoming increasingly accessible and pervasive, while their improved capabilities make them ideal clients for the implementation of many various mobile applications [8], among which m-Learning represents one of the most important and attractive ones.

However, the acceptance of new m-Learning systems is highly dependent on usability challenges, the most important of them being technology variety, gaps in user knowledge and user diversity [9]. The potentiality for including the widest possible parts of the population in the interactive mobile learning process implies particular emphasis on the user interface design and the quality of interaction [5]. These HCI issues become even more considerable within present day mobile device applications (MDAs), which have a firm tendency for increased complexity, sophisticated interfaces and enriched graphics. Hence, the development process for such MDAs must involve personalization procedures that are essential for tailoring them to individual users' needs and interaction skills. As the general framework of mobile interaction heavily bases on two interaction styles – menu-based and direct manipulation, we have focused our interest on personalization of MDAs through a transformable and moveable menu component with adaptable and adaptive features, introduced in [6] – see Fig. 1.

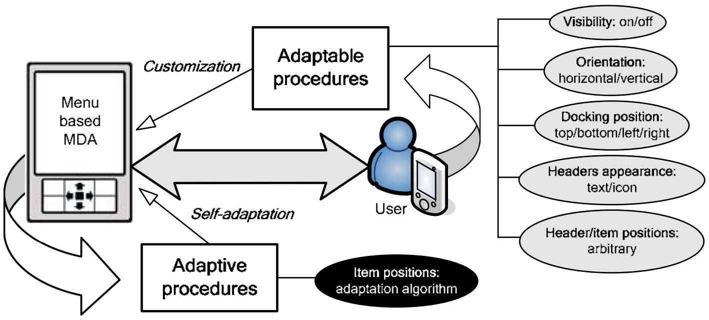
Customization
Adaptable procedures
Menu based MDA
User
Self-adaptation
Adaptive procedures
Item positions: adaptation algorithm
Visibility: on/off
Orientation: horizontal/vertical
Docking position: top/bottom/left/right
Headers appearance: text/icon
Header/item positions: arbitrary

Fig. 1. Adaptation algorithm usage in the general menu personalization process

In this paper we analyze from a theoretical point of view the respective menu navigation efficiency, in the case where automatic interaction personalization is provided by the usage of two different adaptation algorithms.

2 Transformable Menu Component

In general, menus represent a core control structure of complex software systems, and therefore provide an interesting object for personalization research [2], especially in the mobile devices' environment. Here the focus is primarily on the speed of interaction between user and menu-based MDA, since this is considered to be one of the main factors in producing a truly usable system. Minimization of the interaction burden for the user, what is an important aspect of speed [1], can be accomplished both by avoiding a multi-screen menu hierarchy and by reducing the number of keystrokes. For that reason, our menu component has the usual well-known form, with size and shape adequately reduced according to mobile device display limitations (Fig. 2).

Because of the user diversity and high probability that different users will use different navigation patterns, even when working on very similar tasks, adaptation algorithms can generate various menu configurations [6]. A particular configuration thus personalized can be retrieved (at MDA startup) from and stored (at MDA shutdown) to the *Record Management System* (RMS) of the local device, or to a remote server by

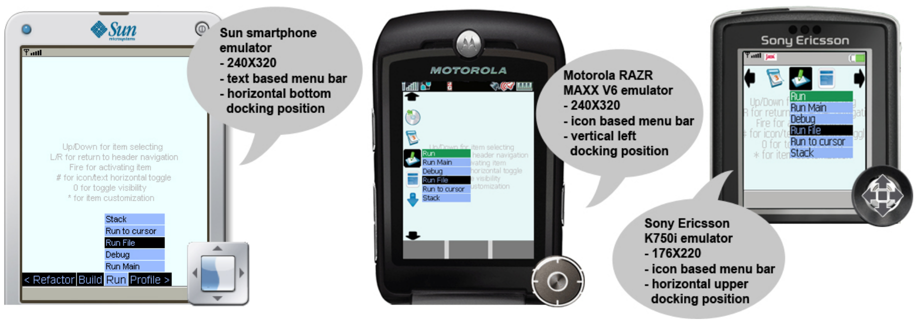
Sun smartphone emulator
- 240X320
- text based menu bar
- horizontal bottom docking position
Up/Down for item selecting
L/R for return to header navigation
Fire for activating item
# for icon/text horizontal toggle
0 for toggle visibility
* for item customization
Stack
Run to cursor
Run File
Debug
Run Main
< Refactor Build Run Profile >
Motorola RAZR MAXX V6 emulator
- 240X320
- icon based menu bar
- vertical left docking position
Sony Ericsson K750i emulator
- 176X220
- icon based menu bar
- horizontal upper docking position
Run
Run Main
Debug
Run File
Run to cursor
Stack

Fig. 2. Menu componet running on different device emulators

the respective *Servlet* application. RMS represents both an implementation and API for persistent storage on *Java ME* devices. It provides associated applications (of the *Java MIDlets* class) the ability to access a non-volatile place for storing the object states [7]. Since the RMS implementation is platform-dependent, it makes a good sense to guarantee redundancy by additionally storing menu configurations to the remote server and subsequently retrieving them from the server.

A personalized menu must furthermore provide an easy access to all of the existing menu functions, including adaptable ones. For that reason, our menu component is thoroughly modeled using state diagrams (*cf.* [11]) where all available state transitions are initiated through exactly one keystroke on the mobile device input, thus providing a platform for optimal interaction efficiency (Fig. 3).

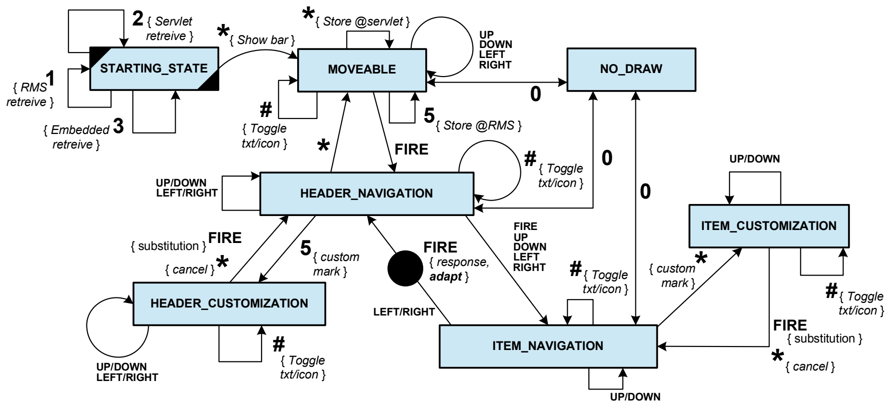

Fig. 3. Menu state diagram. The black node represents the spot where automatic adaptation is performed.

Regarding the implemented adaptable (i.e. user-controlled) options (see Fig. 1), the user is provided with the ability (i) to easily control the visibility mode, (ii) to adjust menu orientation and respective docking position, (iii) to toggle the menu appearance both between character-oriented and iconic-based styles and (iv) to manually customize both the menu header and item positions within its hierarchical scheme. On the other hand, the use of adaptive techniques means that MDA user interface changes will be partially controlled by the system itself, providing usability enhancement through increased interaction speed while getting m-Learning tasks done.

3 Adaptation Algorithms

First of all, it should be noted that automatic adaptation is based on algorithms both monitoring a user's prior navigation behavior and rearranging menu item positions within a particular popup/pulldown menu frame.

In the following two adaptation approaches are compared with the original (static) menu configuration. While a *frequency-based* (FB) algorithm simply changes item positions according to their selection frequencies, a *frequency-and-recency-based* (FRB) algorithm refines the same idea by additionally promoting the most recently selected item [3]. The difference in related adaptation effects is visualized in Fig. 4.

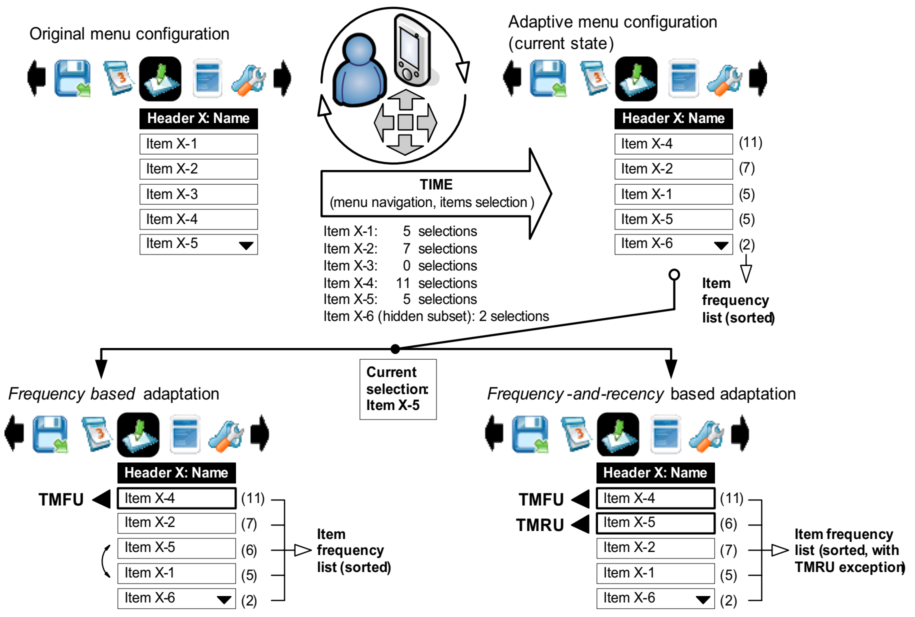

Fig. 4. The difference between *frequency-based* and *frequency-and-recency-based* adaptation: while the former promotes the most frequently used (**TMFU**) item only, the latter additionally promotes the most recently used (**TMRU**) one

As shown in the figure above, after a certain period of time and a related menu navigation pattern, the initial item positions are rearranged according to a sorted item frequency list. Denoting this as a current state of the adapted menu configuration, two outcomes are possible upon *Item X-5* selection. If FB adaptation is used, new repositioning is expected, based on the updated item frequency list. However, if automatic adaptation is ensured by the FRB algorithm, the currently selected item (*Item X-5*) will be replaced to the TMRU position, updating the frequency list and reordering its items. Using recency criteria, every menu item has a "fair chance" to quickly appear and be retained in the promoted part of the item set, regardless of its current frequency value.

The core of the automatic adaptation algorithm can be specified through the following pseudocode, with the framed part referring to the case when the recency condition is active (can be omitted if the FB approach is used):

```
if (keyPressed=FIRE_BUTTON) then
  Update_Frequency_List(selected_item, item_freq++);
  if NOT position(selected_item, TMFU) then
    Update_ItemPositions(itemSet, freqList, noRecency);
    if NOT position(selected_item, TMRU) then
      Move(selected_item, TMRU_position);
      Update_ItemPositions(itemSet, freqList, recency);
    end if
  end if
  Application_Response(selected_header, selected_item);
end if
```

The abovementioned algorithm's usage is inspired by the work carried out in [2], where a similar approach was applied on adaptive split menus. In that particular research, the idea of using frequency and recency characteristics emerged from experiences gained working with the *Microsoft Office 2000* suite with dynamic menus, which adapt to an individual user's behavior. Whilst the related work is based in the desktop application environment (with the mouse as exclusive input device), we are dealing with an MDA setting and the corresponding mobile device navigation keypad. We believe that efficiency enhancement in mobile interfaces navigation, provided by automatic adaptation, exceeds the debatable benefits reached in desktop menu navigation.

4 Task Simulation Framework

As there is still a lack of evaluation studies capable of distinguishing adaptivity from general usability [4], we have developed a task simulation framework able to compare static and adaptive menu configurations and their respective navigation options.

Since the time required for the completion of menu navigation tasks directly depends both on the time to locate a target menu header in a root menu bar, as well as on the time to select an item from a single popup/pulldown menu frame, it is quite straightforward to specify the navigation performance level by determining the exact number of keystrokes needed for task fulfillment, within the input set of four navigation keys and a fire button (Fig. 5).

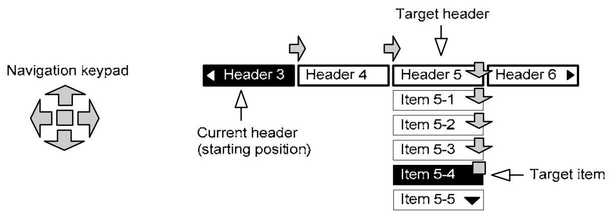

Fig. 5. If the general navigation keypad is used, it is a simple task to calculate the "*keypad distance*" from the starting position to destination. If *Header 3* is considered as the current menu position, selecting *Item 5-4* would require 7 keypad strokes: 2 RIGHT arrows, 4 DOWN arrows, and the FIRE button.

Various static and adapted menu configurations can be compared, based on the aforesaid calculation method and several simulation parameters which are introduced and thoroughly explained in Table 1.

Table 1. Parameters (and structures) used in the task simulation framework

Configuration	Parameter / Structure	Type	Characteristic
Menu configuration	Headers	User-defined	**Number** of first-order menu options (number of menu headers)
	Items_MIN	User-defined	Minimal **number** of items within each menu frame
	Items_MAX	User-defined	Maximal **number** of items within each menu frame
	Config	**Random**	Randomly generated menu configuration with *#Headers*, each of them containing between *#Items_MIN* and *#Items_MAX* items
Task configuration	Picks	User-defined	**Number** of randomly chosen menu selections
	Repetition	User-defined	Repetitive selections **percentage** (within set of *#Picks* selections)
	Pools	User-defined	**Number** of task subsets (for repetitive selections distribution)
	Task	**Random**	Randomly generated navigation and selection task, with a given number of randomly chosen menu selections and defined repetitive selections distribution

Simulations can be performed with different menu configurations, which are randomly generated according to a given number of menu headers (*Headers*) and an allowed number of items within each menu frame (*Items_MIN* and *Items_MAX* being the limitations). This way we are confronted with the option to analyze many various configurations that can afterwards be classified basing on menu sizes.

The basis of the simulation process is a randomly generated navigation and selection task, which consists of an explicit number of random selections (*Picks*), some of which are repetitive in accordance with a defined percentage (*Repetition*) and distribution (*Pools*). It is highly unlikely that the user will make the most of her/his repetitive selections at once, therefore these selections are evenly dispersed throughout the whole task, thus generating the desired distribution (Fig. 6).

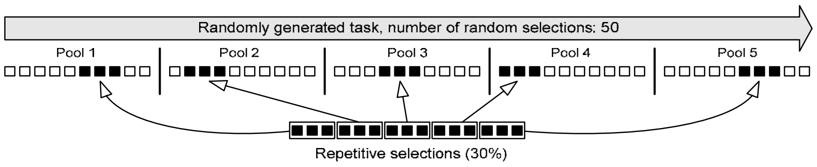
Randomly generated task, number of random selections: 50
Pool 1
Pool 2
Pool 3
Pool 4
Pool 5
Repetitive selections (30%)

Fig. 6. Distribution of repetitive selections within a randomly generated task. Parameter values for task configuration: *Picks* = 50, *Repetition* = 30%, *Pools* = 5.

Obviously, if items change their initial positions within a particular menu frame (according to algorithm instructions), a variation in *keypad distanc*e for selecting a particular item in both the original and the modified menu configuration will result. The overall difference between static and adaptive configuration in the total count of keystrokes (for completing the given task) represents the interaction speed enhancement provided by (automatic) adaptivity. In the task simulation framework, this criterion will be used to evaluate the usefulness of menu adaptation and to quantify efficiency of the used adaptation algorithms.

5 Simulation Results

The measure of interaction speed improvement derived from automatic adaptation is given by (X-Y), where X stands for the number of keystrokes required for completion of the generated task using the original menu configuration, while Y stands for the number of keystrokes using the adaptive one. Fig. 7 shows a sample result of an FRB adaptation simulation session.

Menu configurations that are used within the simulation process are categorized according to structure complexity, so we basically distinguish small, medium and large scale menus. It is easy to realize that wading through large scale menus requires increased user attention, because related headers can extend on several display screens. Because of the random characteristic of the task generation process, we used exactly 100 different instances of the generated task for every particular set of simulation parameters. Consequently, 100 simulation sessions are performed for a distinct menu configuration and specified task class, while the final simulation results are presented as a mean value of data thus collected.

Altogether 4500 simulation runs have been carried out for 9 menu configurations and 5 classes of randomly generated tasks. The mean values show an observable level of navigation efficiency enhancement for both FB and FRB approaches. The obtained simulation results are structured and presented in Table 2.

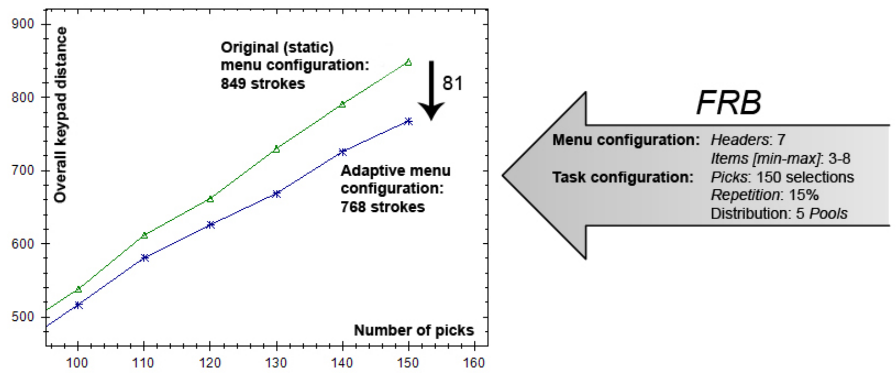

Fig. 7. Sample result derived from task simulation framework. In this particular case, FRB adaptation decreases overall keypad distance for 81 keystrokes, thus decreasing the input interaction burden for approximately 10%.

Task classes with small designated number of *Picks* represent the user's interaction with the menu-based MDA in short interactive cycle. Conversely, tasks which include very large number of selections (e.g. Class #5 with 10000 *Picks*) correspond to longer usage of the application with menu component navigation options. Related to that, we can see that adaptivity effects considerably grow with task duration, so users can improve their navigation efficiency with the duration of adaptive menu usage.

Table 2. Simulation results. For every task class, we set a *Repetition* parameter value to 15%.

Menu configuration			**Task classes and simulation results**				
Scale	Headers	Items [min-max]	Class #1 **100** ***Picks*** 5 *Pools*	Class #2 **200** ***Picks*** 5 *Pools*	Class #3 **500** ***Picks*** 10 *Pools*	Class #4 **1000** ***Picks*** 30 *Pools*	Class #5 **10000** ***Picks*** 300 *Pools*
			FB / FRB	FB / FRB	FB / FRB	FB / FRB	FB / FRB
small menu	3	2-5	5 / 9	14 / 19	42 / 70	40 / 105	102 / 652
	4	2-5	5 / 8	17 / 23	43 / 67	71 / 152	98 / 584
	5	2-5	4 / 6	17 / 19	32 / 41	44 / 79	118 / 770
medium menu	6	3-8	14 / 18	25 / 31	50 / 71	133 / 255	330 / 1957
	7	3-8	13 / 16	37 / 43	99 / 132	116 / 217	257 / 1348
	8	3-8	16 / 19	52 / 59	100 / 123	142 / 241	503 / 2240
large menu	9	4-11	30 / 34	57 / 64	128 / 160	187 / 318	785 / 3027
	10	4-11	29 / 32	65 / 71	150 / 178	185 / 298	692 / 2767
	10	8-12	34 / 38	85 / 94	212 / 247	255 / 417	1081 / 4263

According to simulation results, the benefit of adaptivity implementation is on the other hand questionable in small scale menu configurations, especially within infrequently used applications. In such menus all popout/pulldown frames can be expanded on a single display screen, resulting in no need for navigation to hidden item subsets, hence rearranging items according to prior user's navigation patterns has no manifest significance.

Regarding the recency criterion, in most cases FRB adaptation resulted with better enhancement with respect to the FB approach, regardless of menu configuration scale. However, this difference in adaptivity effects becomes more prominent within tasks formed by a larger number of menu selections (e.g. within task class #5, FRB adaptation outperforms several times the FB approach). Hence, promotion of the most recently used items within a particular menu frame is preferable in MDAs requiring a frequent usage of a navigation-and-selection interaction style (as is the case in e.g. m-learning applications).

Generally speaking, simulation outcomes support the concept and confirm the usefulness of adaptive techniques implementation for menu navigation in the mobile application environment. Nevertheless, it should be noted that the abovementioned conclusions emerge from theoretically based results. Let us note that it is quite hard to model real application tasks by using random generators because actual navigation patterns contain to some extent more predictive sequences of menu selections, which is not the case in our task simulation framework. This is the reason we can expect even more enhanced results in real application adaptation scenarios. Users' possible mistakes in navigation, impressions and levels of satisfaction while working with adaptive interfaces are excluded from this analysis, as the groundwork for these indicators (e.g. usability testing) is yet not implemented.

6 Conclusion and Future Work

M-learning systems, one of our main research interests, will certainly become an additional advantage in the wide-ranging process of lifetime learning. When developing related m-learning MDAs, there arises a strong aspiration to completely utilize mobile device technology upgrowth, and to make these applications very powerful, graphically rich and usable. Hence, following the concept of universal usability, our efforts are focused on the quality of mobile user interaction. We make use of personalization procedures in order to enable users to work with MDA interfaces that are adjusted according to their preferred individual interaction patterns, thus making the users faster and more satisfied in performing assigned (m-learning) tasks.

In our previous work, we introduced a transformable menu model for MDAs, with personalization capabilities derived from the use of both adaptable and adaptive techniques. The model is implemented as a *Java ME* API extension, and can easily be reused in all likewise applications (not necessarily m-learning ones). In this paper we are dealing with system-driven personalization of the presented menu model and efficiency of adaptation algorithms. Various static and adaptive menu configurations are compared within a task simulation framework, and the results thus obtained confirm that the usage of adaptivity makes a difference, providing a valuable improvement in navigation efficiency within menu-based mobile interfaces.

Directions for future work include further improvements of our cognition on mutual influence between user diversity and automatic interaction adaptation. We would like to identify the conditions with respect to which the benefit of adaptation is more valuable than the eventual loss of control due to unexpected changes of the menu configuration. Results derived from the described task simulation framework will be substantiated with new research outcomes which base on running adequate usability tests. Moreover, for every presented and completed user task, appropriate time measurements will be carried out, within both static and adaptive menu-based applications. With results thus collected, we expect to get a better insight into the correlation between theoretical and empirical adaptation effects.

Acknowledgments. This paper describes the results of research being carried out within the project 036-0361994-1995 *Universal Middleware Platform for e-Learning Systems*, as well as within the program 036-1994 *Intelligent Support to Omnipresence of e-Learning Systems*, both funded by the Ministry of Science, Education and Sports of the Republic of Croatia.

References

1. Anderson, D.J.: Speed is the Essence of Usability (Editorial). UIdesign.net: The Webzine for Interaction Designers (1999), `http://www.uidesign.net/1999/imho/sep_imho2.html`
2. Findlater, L., McGrenere, J.: A Comparison of Static, Adaptive, and Adaptable Menus. In: Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI 2004), pp. 89–96. ACM, New York (2004)
3. Gajos, K.Z., Czerwinski, M., Tan, D.S., Weld, D.S.: Exploring the Design Space for Adaptive Graphical User Interfaces. In: Proc. 8th Int'l. Working Conf. Advanced Visual Interfaces (AVI 2006), pp. 201–208. ACM, New York (2006)
4. Glavinić, V., Granić, A.: HCI Research for E-Learning: Adaptability and Adaptivity to Support Better User Interaction. In: Holzinger, A. (ed.) USAB 2008. LNCS, vol. 5298, pp. 359–376. Springer, Heidelberg (2008)
5. Glavinic, V., Ljubic, S., Kukec, M.: A Holistic Approach to Enhance Universal Usability in m-Learning. In: Mauri, J.L., Narcis, C., Chen, K.C., Popescu, M. (eds.) Proc. 2nd Int'l. Conf. Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2008), pp. 305–310. IEEE Computer Society, Los Alamitos (2008)
6. Glavinic, V., Ljubic, S., Kukec, M.: Transformable Menu Component for Mobile Device Applications: Working with both Adaptive and Adaptable User Interfaces. International Journal of Interactive Mobile Technologies (iJIM) 2(3), 22–27 (2008)
7. Mahmoud, Q.: MIDP Database Programming Using RMS: a Persistent Storage for MIDlets. Sun Developer Network (SDN) - Technical Articles and Tips, `http://developers.sun.com/mobility/midp/articles/persist/`
8. Roduner, C.: The Mobile Phone as a Universal Interaction Device – Are There Limits? In: Rukzio, E., Paolucci, M., Finin, T., Wisner, P., Payne, T. (eds.) Proc. of the MobileHCI Workshop on Mobile Interaction with the Real World (MIRW 2006), pp. 30–34 (2006)
9. Shneiderman, B.: Universal Usability: Pushing Human-Computer Interaction Research to Empower Every Citizen. Comm. ACM 43, 85–91 (2000)
10. Stephanidis, C.: Editorial. International Journal - Universal Access in the Information Society 1, 1–3 (2001)
11. Thimbleby, H.: Press On: Principles of Interaction Programming. The MIT Press, Cambridge (2007)