ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20140623205344

USER INTERFACE ADAPTATION FOR ICT BASED ALTERNATIVE AND AUGMENTATIVE APPLICATIONS

Ivan Vučak, Željka Car, Marin Vuković

Original scientific paper

Support of ICT in Alternative and Augmentative Communication (AAC) has been recognized as the key enabler of better inclusion of persons with complex communication needs into everyday life. Due to complex, and often very individual, communication needs, such persons need the ability to quickly adapt graphical user interfaces according to their needs, skills, impairments and possibilities. The paper proposes initial automatic adaptation of AAC application user interfaces that is performed only once and then distributed across AAC applications through a specialized AAC platform. The adaptation is formally specified using timed Petri nets and evaluated on a group of users, aiming at producing components that can be re-used without the need for further evaluation, thus enabling rapid development of new applications without the need for real user testing, due to the nature and availability of AAC users. Finally, the paper proposes several methods of choosing symbols and weighs the benefits of each method according to evaluation results.

Keywords: *alternative and augmentative communication; automatic user interface adaptation; calibrator; communicator; Petri net*

Prilagodba korisničkog sučelja ICT aplikacija za potpomognutu komunikaciju

Izvorni znanstveni rad

Podrška informacijsko komunikacijske tehnologije za potpomognutu komunikaciju može olakšati uključivanje osoba sa složenim komunikacijskim potrebama u svakodnevni život. Takvim je osobama, zbog specifičnih i često individualnih potreba, nužno omogućiti jednostavnu prilagodbu grafičkog korisničkog sučelja u skladu s njihovim potrebama, sposobnostima, teškoćama i mogućnostima. U radu se predlaže inicijalna automatska prilagodba korisničkog sučelja aplikacija PK koja se obavlja samo jednom, a zatim distribuira ostalim aplikacijama PK putem razvijene platforme. Adaptacija je formalno definirana Petrijevim mrežama i evaluirana od strane grupe korisnika, pri čemu su rezultati evaluacije uključeni u sam model. Cilj modela jest implementacija programskih komponenti koje će se moći opetovano koristiti bez potrebe za evaluacijom od strane korisnika PK što je vrlo bitno obzirom na poteškoće i složene potrebe takvih korisnika. Ovakav će pristup u konačnici omogućiti brži razvoj usluga i aplikacija PK. Uz navedeno, u radu se predlažu i evaluiraju metode za odabir simbola.

Ključne riječi: *automatska prilagodba korisničkog sučelja; kalibrator; komunikator; Petrijeve mreže; potpomognuta komunikacija*

1 Introduction

Augmentative and Alternative Communication (AAC) includes various methods of communication used by *persons with complex communication needs* (CCN). They possess significant speech, language, motor and/or cognitive impairments which restrict their ability to participate independently in society [1]. Common causes of complex communication needs are severe intellectual disability, cerebral palsy, autism, childhood apraxia of speech, traumatic brain injury, stroke and degenerative diseases [2].

Historically, AAC has grown over the past 50 years from nonexistence to full acceptance as an area of interdisciplinary clinical practice that attempts to compensate, either temporarily or permanently, for communication deficits [3, 4]. AAC methods are based on the use of graphic and textual *symbols* (rather than written words alone) to represent certain objects, actions, or concepts. There are different commercial and non-commercial *symbol galleries* available, or collections of a large number of downloadable symbols.

Research [3] shows a great number of ICT initiatives for AAC. Number of AAC applications for mobile devices is growing extremely fast, practically on a daily basis, especially after the appearance of touch screen devices on the market. Each AAC user has a unique set of capabilities and needs and these sets significantly differ. Persons with CCN are a diverse group ranging from very young to very old, with a variety of diagnoses and health conditions, life experiences, skills, abilities and preferences. Some persons require AAC technologies for a short time only, while others need communication technologies throughout their lives. People with some degenerative conditions often have rapidly changing communication needs and technology solutions must accommodate these changes [4]. By using AAC software applications, users are able to communicate and learn at their schools, homes, worksites and communities. Different users may have different devices or use the same AAC application on different devices (for example while working at home or at school).

Different symbol users utilize symbols differently. Each symbol features properties such as colour and contrast, size, text labels and voice reproduction of the symbol meaning. Persons with hearing impairment may use both text and symbols and require visual feedback. Persons with visual impairment require high-contrast symbols, large symbols and voice reproduction of the symbol meaning. Persons with motor disabilities need large symbols and possibility of their easy sequencing in order to compose a message. Users either browse the symbol based content or select symbols from the given (often very large) set/gallery. Also, the purpose of symbol usage, as well as context, determines their display at the devices. So the basic feature of effective usable AAC applications should be the possibility to accommodate the symbol appearance according to the needs of the particular user, and the accommodation should be done in a simple way.

This paper presents original solution to the problem of initial automatic adaptation of the AAC application user interfaces (UI) considering the specific needs and possibilities of each user. The solution is implemented as user interface adaptation service for AAC applications for devices with touchscreen, particularly tablet devices.

The paper is organized as follows: Section 2 describes research background, underlying problems and related work. Concept of user interface adaptation for AAC applications, adaptation services architecture and Petri net formal model are presented in Section 3. Implementation of AAC adaptation services is described in Section 4. Usage case studies of AAC adaptation services are presented in Section 5, while future research directions are indicated in Conclusion.

2 Background, problem description and related work

The research described in this paper is part of the interdisciplinary cooperation in the area of ICT-based solutions for alternative and augmentative communication that have started in 2009 and has been intensified last five years, bringing together scientists and professionals from areas such as information and communication, computer science, education, rehabilitation, psychology and design.

These solutions provide support for communication, learning, and efficient execution of daily activities for persons with complex communication needs and may increase their social inclusion. In addition, these solutions are also aimed for family members, educators and therapists. Research consortium[1] contains the following University of Zagreb faculties: Faculty of Electrical Engineering and Computing, Faculty of Education and Rehabilitation, Faculty of Graphic Arts and Faculty of Humanities and Social Sciences – Department of Psychology. During the research a strong cooperation has been established with a number of stakeholders: parental and professional non-governmental associations, organizations and individuals presenting user needs, participating in the process of ICT solution development and taking part in the evaluation process of each developed ICT solution.

Within this research an AAC software platform is implemented for component based development and deployment of interoperable and scalable symbol based services [5]. AAC adaptation services described in this paper are developed and integrated within this platform.

The research problems that are addressed in this paper and for which solutions are proposed and implemented are the following:

(A) *Initial automatic adaptation of the AAC service user interface conforming to the AAC user needs and capabilities*

Initial adaptation refers to the first user interaction with one of several available AAC applications within AAC software platform, and point in time when there is no information about particular user. Goal of the initial adaptation is to determine the earliest limitation point for user-application interaction because at that point the application has no data about user's needs and possibilities.

This could be solved by manually adjusting necessary parameters (for example symbol size, appearance, contrast, number and position of symbols, the most appropriate symbol gallery etc.) in the user settings of the particular service at the particular device. The main problem here is how to determine, in short and non-exhaustive way, the appropriate parameter value, keeping in mind that user's caretaker or rehabilitator performs the adjusting.

Other way of approaching this problem is creation of predefined categories of user interfaces for different user categories but all scientist and professionals who work in the field of AAC agree that it is not absolutely possible to categorize AAC users since each user has its own unique set of abilities related to visual and motor capabilities, as well as a level of previous experience in working with particular device and kind of application (communication, education, or assistive technology application) [6] [7].

(B) *Automatic determination of the rehabilitator preferences in terms of selection of the symbol/educational content from a set of possible symbols/contents within particular AAC service.*

Caretakers and AAC professionals, especially educational rehabilitators and specialized therapists, often work at educational institutions, specialized hospitals, polyclinics and day-care centres with a number of AAC users sequentially on the same device. They need to adjust content of particular AAC application for each user individually and they are doing this number of times during their work.

Also, very often AAC rehabilitators need to change a content of particular AAC application during the work with a specific user, for example, they add new symbols into the service, change the appearance of the given symbols by making new associated sound or text-label etc. All this should be done quickly and efficiently to avoid distraction of AAC user. So it is desirable that the application has the ability to automatically determine which method of selecting symbols best conforms to the particular AAC rehabilitator. This could be done by dynamic evaluation of AAC professional interaction with the user interface via calibration test.

Research focusing on effective and efficient input and output on mobile devices has always been an important topic as it addresses the inherent conflict between the desire for a small device size and ergonomic aspects defined by the physiology of user eyes, hands and fingers [8]. Analysis of the smartphone users touch behaviour by using a game published to the Google Play to determine the error rate for different target sizes and screen positions is presented in [8]. An adaptive on-screen keyboard that observes where the user is touching the display in relationship to the displayed key and using this information for adapting shape, size and location of virtual keys in order to improve error rate is presented in [9]. CARDIAC project reports [10] states that "accessibility problems of specific groups of users have been addressed through *Assistive Technology* (AT) based adaptations, and systematic *Design for All* approaches have been elaborated and applied in various domains at a research level. Still, the field is currently in need of a breakthrough towards the adoption in practice of design approaches, based on the accumulated knowledge, leading to accessible and usable inclusive interfaces". Also, mentioned report [10] states that "design techniques and methodologies able to address users' diversity, by means

[1] ICT Competence Network for Innovative Services for Persons with Complex Communication Needs, http://www.ict-aac.hr

of modelling and adaptation, exist but they are not enough known and used".

Studies have demonstrated a growing demand for TV-based entertainment and interaction for elderly people [11] and research described in [12] follows the approach of adaptive multimodal user interfaces that can automatically adapt to the needs and preferences of elderly users with mild ageing-related impairments, focus particularly on connected TVs. Prototype of a system which adapts user interfaces to achieve accessibility of web-based services and devices from their own handheld device in a personalized way is presented in [13]. Automatically generated interfaces which are tailored to an individual's motor capabilities and can be easily adjusted to accommodate varying vision capabilities based on a motor performance test are described in [14].

Project [15, 16] is addressing UI accessibility problems for elderly users when using web and TV platforms. Project presents developed framework which allows developers to efficiently integrate accessibility and personalization features into their applications, minimizing intervention with existing development process and tools.

ICT service/software with the ability to adapt to current circumstances is referred to as context-aware software [17]. Term context is very complex and maybe the broadest and largely accepted definition is the one given by Dey and Abowd [18]: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". A system is context-aware if it considers context when providing relevant information and services to the user, relevant to user's current task [18].

Context-awareness has been widely applied in different services and applications for different purposes – from content localization to achieving higher service security. In our research we are developing context-aware system with context based on the online evaluation of user abilities. Only by considering user context we can thoroughly achieve personalized and adjusted service.

Selected related work papers are all addressing same issue: adjusting input mechanism or user interface to users' preferences and abilities. Research papers [8] and [9] are dealing solely with touch screen devices and their input mechanism while papers [10, 12, 13] are analysing users' interfaces in a broader sense. Touch screen device researches are all starting from Fitt's law (modelling the time needed to select single target) with further improvements (adjusting to specific problem, e.g. selecting multiple objects) [9]. These papers present different approaches to the similar problem: improving users' experience based on different inputs such as device hardware and software features and purpose, users' abilities, etc.

Generally, there is lot of research in the area of adaptive user interfaces but so far according to the author's knowledge there is no application of this kind in the area of alternative and augmentative communication. This paper describes our research effort in this direction.

3 Concept of user interface adaptation for AAC applications

In order to adapt application *Graphical User Interface* (GUI) a user profile, describing user preferences and capabilities in terms of interaction with particular device and application, has to be defined. We propose the concept that employs **two calibrators** with the purpose to determine these user preferences.

There are usually two types of end-users in AAC applications; AAC users and AAC rehabilitators (therapists and caregivers of people with complex communication needs). **First calibrator** - *AAC user interface (UI) calibrator* - is aimed for AAC users and its goal is to determine their preferences. The parameters determined within the first calibrator define the look and feel of AAC application experienced by end-user. **The second calibrator** - *Symbol selection user interface* (UI) *calibrator* - is aimed for AAC rehabilitators and its goal is to set preferences regarding AAC application GUI.

Prior to using any AAC application, both AAC users and AAC professionals should run specific calibrator. After the calibration is completed, parameters regarding GUI are stored in a centralized database. Any other communication or education application within AAC software platform may access the centralized database, read users' parameters and adapt its GUI according to the stored preferences and capabilities of particular user.

3.1 AAC adaptation service architecture

The proposed AAC adaptation service architecture is shown in Fig. 1 and it consists of several entities. As said in previous Section, AAC *UI calibrator* is intended for AAC users and its goal is to learn about users' capabilities through a *series of trials*. Data about learned capabilities may then be used for user interface adaptation in various AAC applications. *Symbol selection UI calibrator* is intended for AAC rehabilitators (caretakers and professionals), and its goal is to determine their preferences regarding user interface. Preferences relate to the way the symbols and symbol categories are presented and to enable easier administration of AAC application content. Similar to AAC *UI calibrator*, the *Symbol selection UI calibrator* holds a series of trials. Each calibrator is monitored by a *Software agent* that gathers trial results and forwards them to the *Coordination service*.

Coordination service derives information about user's capabilities and rehabilitator's preferences from the trial results. This information is then stored in a settings database for further use by various AAC applications. The process of data analysis, as well as communication with calibrators and applications, is orchestrated by a *Coordination agent* that receives trial results from the calibrators, initiates data analysis and responds to requests about user settings from other applications. AAC applications that use this data should also have an application agent responsible for communication with the *Coordination service*. Furthermore, AAC applications may indicate a change in user capabilities or user preferences and update the settings through a *Coordination agent*.

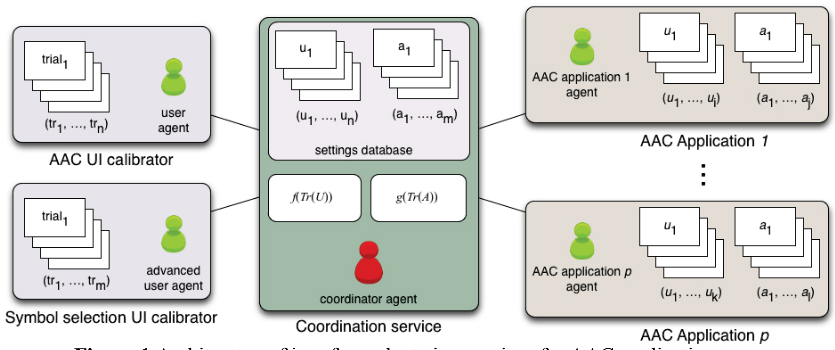


Figure 1 Architecture of interface adaptation services for AAC applications

In order to lay out the foundation for developing effective adaptation service that could be formally analysed, and to provide systematical approach for incremental development and future maintenance of AAC adaptation service, it is formally modelled and described as follows.

AAC user U runs AAC *UI calibrator* and completes a series of n trials, forming an array of n user trial results $Utr(U)$:

$$Utr(U) = \{utr_1, ..., utr_n\} \tag{1}$$

Similarly, AAC rehabilitator P completes a series of m trials on *Symbol selection UI calibrator*, forming an array of m professional user trial results $Ptr(P)$:

$$Ptr(P) = \{ptr_1, ..., ptr_m\} \tag{2}$$

User agents on each calibrator forward these arrays to coordinator agent that initiates data analysis. The analysis depends on trials and can be presented as two different functions $f(Utr(U))$ and $g(Ptr(P))$ that transform trial results into settings' parameters $S(U)$ and $S(P)$, respectively:

$$\begin{aligned} f&: Utr(U) \rightarrow S(U) = \{u_1, \dots, u_n\}_U \\ g&: Ptr(P) \rightarrow S(P) = \{p_1, \dots, p_m\}_P \end{aligned} \tag{3}$$

The number of derived setting parameters for both user and AAC rehabilitator corresponds to the number of trials completed, i.e. each trial results in a single user setting parameter. Finally, setting parameters are stored into database and any other AAC application can use or update them via the coordinator agent.

AAC applications that use derived user setting parameters can select all or some parameters from the database. The number of setting parameters per AAC application depends on the application functionalities and purpose. In general, AAC application p may use k AAC user setting parameters $S_p(U)$, where $k \leq n$, and l AAC professional's setting parameters $S_p(P)$, where $l \leq m$:

$$S_p(U) = \{u_1,...,u_k\},\ k \leq n \tag{4}$$
$$S_p(P) = \{p_1,...,p_l\},\ l \leq m \tag{5}$$

Presented architecture is client server architecture where multiple users are sending data to server when using AAC services. AAC services are client applications while *Coordination service* is a server application responsible for receiving, storing and processing client data.

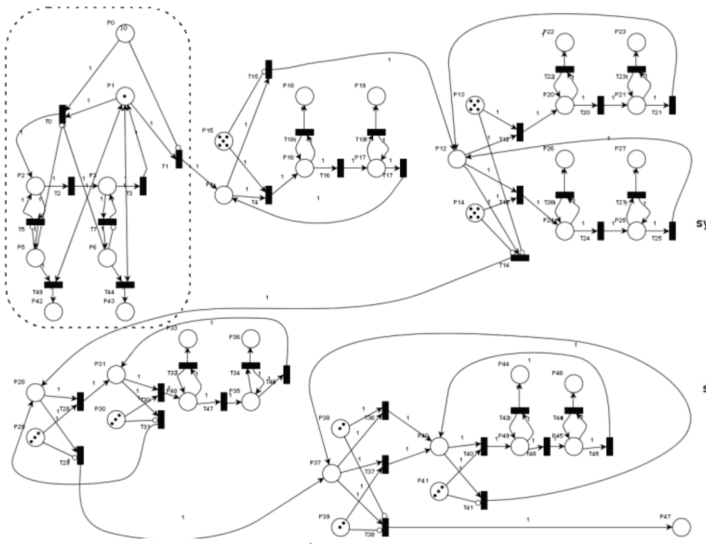
3.2 Petri net formal model

Formal specification of the proposed AAC services is done by using Petri net model. Petri nets are mathematical models for creating formal specifications of system that describes system as a set of strongly defined states and transitions. When used with graphical tools, Petri nets become very powerful technique for system analysis and process simulation.

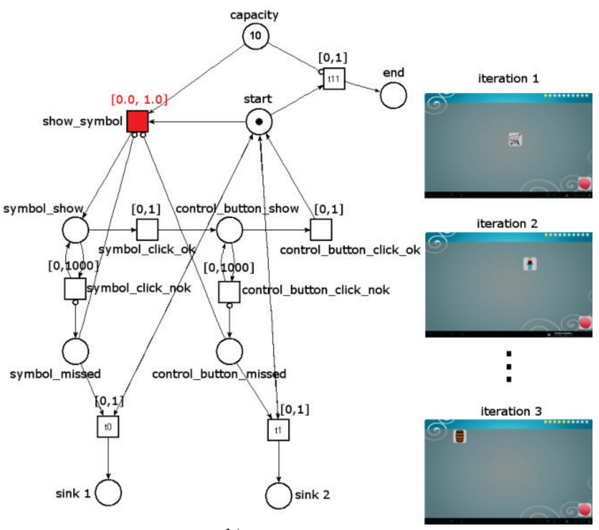
Specifying the proposed AAC service as a formal model described with Petri net makes it possible to precisely analyse the model through formal model verification which ensures model correctness. This means that all possible combinations of inputs and actions are considered, and therefore, all possible outputs are derived and evaluated.

In turn, such specification of the system simplifies possible future extensions to the proposed AAC services, as existing Petri net model can be updated with new functionalities and tested before the actual implementation.

As time represents crucial parameter in both calibrators, mainly because of timeouts in symbol selection process, user reaction and interaction time must be taken into consideration during model analysis. Each trial within both calibrators represents a basic user action (e.g. symbol selection, picking a symbol from various input methods, etc.) that may be used in various AAC applications. In order to obtain the times needed for completing these actions, and to introduce applicable timeouts into model and simulation, both calibrators' trials should be tested with real users. AAC rehabilitators do not present problem in this sense, but there are problems with including significant number of AAC users for evaluation purposes. However, testing a set of representative AAC users on the proposed trials just once would yield a representative timeout values for each trial, i.e. *basic user action*. Incorporating these values into model for each trial would make it possible to re-use and extend the model for any other AAC application that uses these *basic actions*. This would, in turn, speed up the process of designing new AAC applications and reduce the need for testing on real AAC users that may be unavailable.



a)



b)

Figure 2a Symbol Selection UI Calibrator Petri net model
Figure 2b Timed Petri sub model and user screens of the first trial within Symbol selection UI calibrator

Table 1 Symbol Selection UI Calibrator PN model state transition matrix

State	Capacity	Start	Symbol show	Symbol missed	Control button show	Control button missed	End	Sink1	Sink2
capacity	0	0	1	0	0	0	0	0	0
start	0	0	1	0	0	0	0	0	0
symbol show	0	0	0	1	1	0	0	0	0
symbol missed	0	0	1	0	0	0	0	1	0
control button show	0	1	0	0	0	1	0	0	0
control button missed	0	0	0	0	1	0	0	0	1
end	0	0	0	0	0	0	0	0	0
sink1	0	0	0	0	0	0	0	0	0
sink2	0	0	0	0	0	0	0	0	0

Due to the size and complexity of the proposed AAC services model, the modelling and simulation process is explained on a first trial within symbol selection UI calibrator. Complete model of symbol selection UI calibrator is shown in Fig. 2a, with highlighted first trial (which is in detail described in Fig. 2b.).

Since time should be incorporated into the model, the standard Petri net is not suitable for this purpose and an extension to the standard model is used. The most suitable extension is Timed Petri net [19], as shown in Fig. 2b along with examples of user screens through iterations.

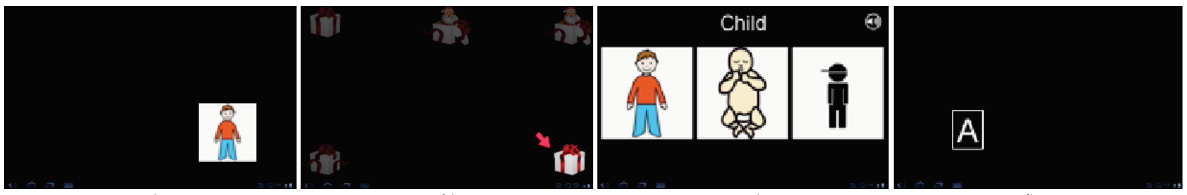
Timed Petri nets are Petri nets in which two times: a, and b ($0 \le a \le b$, $a \neq \infty$), are associated with each transition t_i. The times a and b, for transition t_i, are relative to the moment at which t_i was last enabled. Assuming that t_i was last enabled at time c, then t_i may fire only during interval $[c + a, c + b]$ and must fire at the time $c + b$ at the latest, unless it is disabled before by the firing of another transition [19, 20].

State transition matrix for the timed Petri net sub model (Fig. 2b) is given in Tab. 1. After the first symbol is shown (state *symbol_show*) user can touch symbol (transition *symbol_click_ok*) or he/she can miss it (transition *symbol_click_nok*). If the user didn't succeed, the symbol remains displayed and he/she must try again (transition from *symbol_missed* to *symbol_show* state). If user did succeed in touching symbol he/she should touch control button (*control_button_show*). Control button can also be touched successfully or unsuccessfully so the next possible states are *control_button_missed* if user missed control button or *control_button_click_ok* if user succeeded. If user failed in touching control button, button remains displayed until the user touches it. When user touches control button iteration is completed and if there are more iterations (i.e. marks in *capacity* state), next symbol will be displayed. There are two states with sink function (*sink1* and *sink2*) that are collecting "used" tokens after iteration enabling displayed sub model to act iteratively with the same state and transition conditions in each iteration.

4 Implementation of AAC adaptation services

This section describes calibrators, coordination service and a case-study AAC application *Communicator* in more detail. The typical usage of the proposed services is as follows. Before first use of any AAC application, both AAC users and professionals should go through a series of trials, designed as short games, within appropriate calibrator. These trials gather data about each user and send it to the coordination service for processing.

After the data is processed, the users can run AAC applications that are now adapted to their preferences/capabilities according to the trial results. If calibrators are not available, the users are still able to run AAC applications, but the user interface is set to default and needs to be manually customized by AAC rehabilitator. In case of any change in settings within AAC application, a coordinator agent is notified and new settings are stored in the central database.



a) b) c) d)

Figure 3 Four trials within the AAC User Interface Calibrator

4.1 AAC user interface calibrator

The AAC *UI calibrator* service is presented in [21]. It consists of four short trials as shown in Figure 3. Since calibrator is aimed to AAC users that use symbols, the goal is to determine four important parameters in this context:

- symbol size,
- optimal position of objects on the interface,
- symbol gallery type and
- background to symbol contrast.

In each trial different symbols are displayed in sequence and user's task is to select/touch them. Symbols have different sizes, backgrounds and positions on the interface, so by measuring time and precision of a user touch, the service can calculate the required parameters.

First trial (Fig. 3a) is about determining optimal symbol size; it is the smallest symbol size displayed on the user interface that user is able to select (i.e. touch) successfully. Optimal symbol size is determined by displaying symbols of different sizes through several iterations, from largest to smallest. Each iteration scales the symbol size of previous iteration down by 15 percent. This trial measures the time required for user to touch the symbol as well as the number of successfully touched symbols per iteration. Thus, the trials are constrained by user's ability to precisely select each provided symbol. If a user misses more than half of displayed symbols during any iteration within a trial, the symbol size trial is finished and user moves to another trial.

The aim of the **second trial** is to identify areas of the user interface that are most appropriate for the user with regards to his/her motoric capabilities. The trial is conducted by displaying symbols on predefined interface positions (corners and sides) through several iterations. This is done in such a way that one symbol the user needs to touch is highlighted at a single defined position during one iteration (Fig. 3b). Every symbol displayed in each position needs to be touched twice to be marked as *hit* (symbol appears in two different shapes at the same position to motivate the user to select it twice). Each iteration is limited with the timeout period.

Successful outcome of each iteration is touching the symbol twice in the defined time interval. **Unsuccessful outcome** is reaching the predefined maximum number of misses per iteration or timeout for that iteration. The obtained result is a

- *set of successfully touched positions* and
- *average time taken by user to touch the symbol at certain position*.

These results can be used to determine suitable positions for toolbar, menu bar, symbol gallery or other elements on user interface of any AAC application.

Third trial (Fig. 3c) determines the *most suitable symbol gallery*. It is conducted by displaying example term and group of symbols from various galleries through several iterations. Each symbol that is shown belongs to a different gallery, but depicts the same term (e.g. the term "child" shows various symbols for child from different galleries). Result of this trial is the choice of gallery from which the user has selected the most symbols and therefore showed that this *gallery has, in average, most understandable set of symbols*.

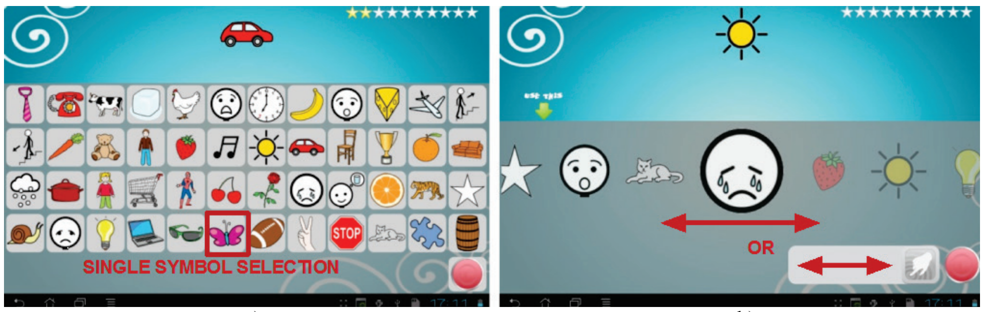
The fourth trial (Fig. 3d) identifies *optimal contrast between symbols and background*. Symbols are represented as framed letters and numbers. Each iteration shows a number of symbols on different screen positions but with a single symbol background combination. The most appropriate combination is derived from the highest number of touched symbols through all iterations. This trial also measures time needed for user to touch symbol and uses a predefined timeout value after which the trial finishes.

4.2 Symbol selection user interface calibrator

This service consists of five trials. **The first trial** is the simplest one and is most similar to the first trial from user calibrator described in previous subsection. Unlike the first, in other four trials the user should select a symbol shown in the upper part of the screen from the set of symbols in lower part of the screen. Each trial runs through several iterations in order to get average results for each trial.

There are two input methods available for symbol selection – *grid* (Fig. 4a) and *circular* (Fig. 4b). Grid editor is well-known form of choosing items (symbols, pictures, icons etc.) while circular editor became popular through touch screen devices. Symbol galleries vary from 1000 to 13 000 symbols so the need for a suitable input method is obvious. However, another problem is that some symbols may be rather similar, which makes grid editor inappropriate because of the large number of small sized symbols presented on the user interface. Circular symbol editor, on the other hand, shows only a few symbols at the time so user can focus but he/she must slide trough a possibly large number of symbols. When using circular input, the symbols can be selected by sliding through them (middle) or by sliding through a slide-bar (bottom right corner).

In the **first trial**, the user is presented one symbol at the time, every time at different position on the screen. The goal is to select/touch the symbol in the shortest possible time and then touch the red *control button* in the lower right part of the screen. *Control button*'s purpose is to make sure that user's hand is always at the same starting position when the next symbol is presented. This trial runs through 10 iterations and its purpose is to determine the existence of areas on the screen where the user is having difficulties touching symbols and to determine user's average response time.



a) b)
Figure 4 (a) Grid and (b) circular symbol selection mode

The **second trial** uses single symbol selection with grid input. Each iteration shows symbols in different order, which is important because it prevents users from remembering the symbol positions in grid during trial, as this would affect the results significantly. The **third trial** uses single symbol selection with circular input. As well as grid input, totally 42 randomly selected symbols are used and each iteration reorders the symbols in circular array.

Since many AAC applications are used for learning, it is necessary to detect the most suitable way of composing symbols into phrases and sentences. The other two trials within this calibrator focus on phrase composition with symbols. In these trials, users must select three consecutively shown symbols. In this sense, **fourth trial** uses grid input for phrase composition while the **fifth trial** uses circular input for phrase composition.

Each trial measures the *time needed for user to select symbol* and records a *number of mistakes* (wrong symbol selection) in order to determine what the most appropriate way to select symbols is.

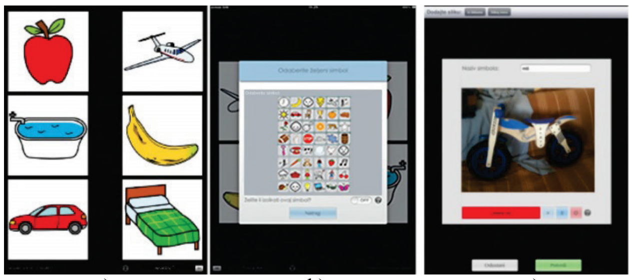
4.3 Communicator as an example of AAC application

An example of AAC application in the proposed AAC platform is *Communicator* [21]. Generally, communicator devices and software applications enable people with communication disabilities to engage into communication with others, thus providing better integration in the society. The main function of *Communicator* is the following: user is shown a set of symbols on the interface and each of them plays a specific sound when touched. This gives user the ability to present his/her wishes, needs or emotions even if he/she is not able to talk or communicate orally at all. There is a number of different communicators in the market today, stand-alone hardware devices as well as tablet and smartphone applications, but none of them support automatic adaptation of user interface.

Communicator within AAC platform shows from 2 to maximum 20 symbols at once, using three available open source galleries (ARASAAC [22], Mulberry [23] and Sclera [24]). Communicator may be used by single or multiple users.

After the user signs in, *Communicator* retrieves calibration data from the coordination service and adjusts its interface using symbol size, gallery and contrast settings that were previously determined by AAC *UI calibrator*. Symbols are positioned throughout the screen in order to show a maximum number of symbols at a time, so the position parameter is ignored in this application. Example AAC user interface is shown in Fig. 5a and AAC rehabilitator UI is shown in Fig. 5c.

When AAC rehabilitator signs in to the *Communicator*, his/her preferences regarding symbol selection mode are retrieved from the settings database (example is shown in Fig. 5b). These settings are used to modify the administrator's interface in *Communicator*, where AAC rehabilitators select symbols that should be presented to the AAC user, create new symbols, record sounds and enter text for each symbol (Fig. 5c). If any interface parameter is manually changed in the settings, online user settings are updated with new parameters through coordination service. Previous data is not deleted, so historical data can be accessed any time and user's progress can be monitored.



a) b) c)
Figure 5 (a) AAC User interface (b) Example symbol selection mode (c) Symbol editing

5 Usage case studies

5.1 AAC users case study

In order to illustrate AAC adaptive services' functionalities we evaluate them on two user case studies. First case study considers two persons with complex communication needs User A and User B while using AAC *UI calibrator* and *Communicator.* Default *Communicator* interface for both users A and B is shown in Fig. 6a.

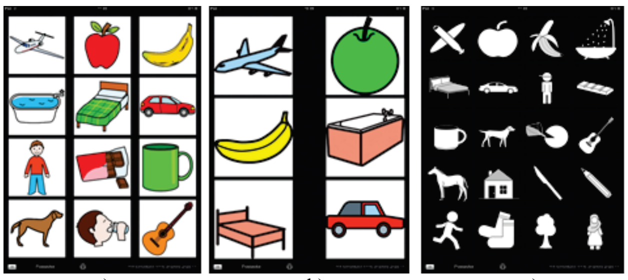


Figure 6 (a) Default Communicator user interface
(b) User A Communicator UI after calibration
(c) User B Communicator UI after calibration

User A went through calibration process first, and has failed to successfully select symbols with size 2 during the second calibrator trial, thus resulting with the largest symbol size chosen as most appropriate. He also showed best understanding of *Mulberry* gallery of symbols by choosing its symbol the most number of times during the third trial. After the calibration process is finished, *User A Communicator* interface looks as shown in Fig. 6b.

User B had no problem with selecting shown symbols but had some difficulties understanding them. *User B* passed first calibration trial with 100 % accuracy, and has selected *Sclera* symbols the most number of times in the third trial. After the calibration process is finished, *User BCommunicator* interface looks as shown in Fig. 6c.

5.2 Analysis of symbol selection method

In order to analyse symbol selection options that are offered to the AAC rehabilitators in applications, *circular symbol editor* and *grid symbol editor* are compared through *Symbol Selection* UI calibrator's trials.

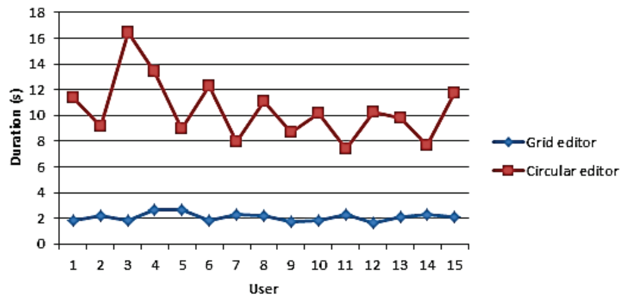


Figure 7 Average time needed for user to select one required symbol for circle and grid editor

The time of the symbol selection is measured in each trial. Fig. 7 shows results of the *mean time needed to find and select required symbol.* Total of 15 users participated in *circular and grid editors'* evaluation. Since symbol selection method is meant to be used by AAC rehabilitators, none of the evaluation users had any complex needs. They consisted of students and faculty staff, with an average age of 22,46 years, with 11 males and 4 females. Results in Fig. 7 show that the circular editor usage requires more time for selecting required symbol than the grid editor. Specifically, average time for selecting required symbol is 4,9 times longer for the circular editor. Also there is a greater variation of the mean time value for different users while using circular editor.

Mean time needed for user to construct given phrases and results for grid and circular editors are shown in Fig. 8. As expected, grid editor yielded better results here as well, with average time for constructing symbol phrase 3,4 times longer when using circular editor. The results show that circular editor requires more time for constructing given phrases and that for some users this time is significant. Also, mean time for different users varies significantly, leading to the conclusion that some users have obstacles if circular editor is implemented.

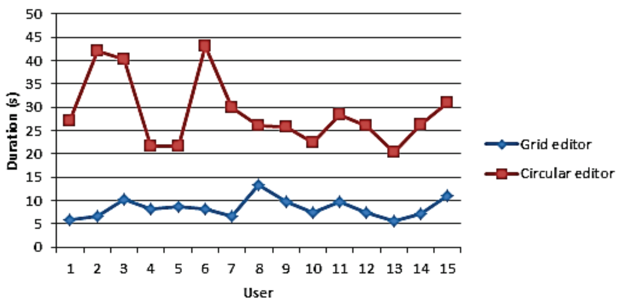


Figure 8 Average time needed for user to construct symbol phrase in circle and grid editor

The biggest difference between these editors is symbol visibility. Grid editor provides smaller symbols but they are all visible while circle editor's symbols are bigger but not all of them are visible to the user at once. This implies that grid editor should be implemented in user interfaces working with limited number of symbols that could be displayed within one screen. We assume that circular editor should be implemented in the user interfaces of AAC applications containing large number of symbols and further testing with a larger set of symbols will be done in future work.

5.3 Performance evaluation of AAC applications

In order to determine the time needed for user interaction, we performed trial involving the same 15 participants as in analysis of symbol selection method. In the context of this particular evaluation, it is necessary to say that all 15 participants were right-handed. The participants used 11,9 inch tablet devices during trial. During this trial, the participants had to select control button first (lower right hand side of the screen) and then select the symbol shown somewhere on the screen. These two actions form one iteration of the trial.

Test results for the first trial in symbol selection UI calibrator, presented by average times needed for each iteration over 15 users, are given in Tab. 2. Average time for single iteration is 932,89 ms, which corresponds to the time needed for one successful control button and symbol touch. The participants had to complete 10 iterations of trials where in each trial the symbol appeared on a different position on the screen. It is interesting to examine the average time required to finish the first iteration of trials, which is significantly longer than the other iterations. We explain this with the fact that users had no experience prior to the trials and they needed some time to understand what is required to do. As expected, after the first iteration the measured time for reaction became more balanced.

After result analysis, the timed Petri net model should be calibrated. In this sense, the total time required for the first trial completion (all 10 iterations) is an indicator of the model validity. The goal is to set the time parameters in such a way that total simulation time is approximately the same as average time from test results. To achieve this, the average time for iteration obtained during testing (approximately 1000 ms) was included in model in various ratios between transitions and total time for simulating 10 iterations varied. Best correspondence between simulation time and time obtained during testing is achieved by using the subnet shown in Fig. 2b.

Table 2 Average user interaction times per iteration in real user testing

Iteration	Measured average user interaction time (ms)
1.	1647,73
2.	895,93
3.	846,4
4.	910,93
5.	914
6.	893,47
7.	809,4
8.	784,4
9.	761,87
10.	864,73

As shown in Fig. 2b, only two transitions have fire time more than 1ms, which is considered immediate. These transitions are *symbol_click_nok* and *control_click_nok*, and they both represent loop transitions. After they fire, the system is back in the same state as it was before. Besides that, both transitions can be activated only once in single iteration. This restriction limits the number of symbol and control button misses, but time component of the misses still remained unmodified.

The simulation of the proposed Petri sub model, with incorporated timeouts, was performed with the tool TINA [25]. The simulation ran through ten iterations in order to see whether the simulation results correspond to the results for completing the first set of trials obtained during the testing. Average time from 15 simulation runs was 21,01 sec, while 15 test users averaged at 20,14 sec, which we consider a good correspondence. This leads to conclusion that the presented timed Petri sub model is able to simulate the behaviour of actual users within examined trial.

Once the complete AAC adaptation concept is modelled in the proposed way, the development of new AAC applications that use basic user actions covered in the proposed trials should be much easier and thus faster. Future AAC application designers should use the defined sub models as building blocks for new applications. Since expected user behaviour and results are simulated by the re-used sub models, the need for actual user testing is eliminated to a large extent, which is important because of the nature and availability of AAC users.

6 Conclusion

The paper proposes user interface adaptation for AAC applications which is intended for both AAC users and their caretakers. Due to the complex needs of AAC users, the focus is on initial adaptation where AAC users are required to do the calibration only once for all AAC applications and services they are using and the ones they might be using in the future. The process of adaptation and settings' synchronization is done through existing AAC platform, a central point for all AAC applications and services.

The proposed adaptation is modelled using timed Petri nets and evaluated on a set of users with specially designed application Calibrator and two symbol selection methods intended for caretakers. Evaluation results are used for comparison and tuning of a timed Petri net model which can be used to model the application usage by real AAC users, since it is very difficult to evaluate new applications on AAC user due to their complex needs. In this sense, using the proposed adaptation model will enable future AAC application designers to re-use the defined sub models as building blocks for new applications. Since real user evaluation results are incorporated in the model, the need for further user testing is minimized, which is important when working with persons with complex communication needs.

In future work we will continue to develop new AAC services within the AAC platform with focus on recorded user settings' portability over various AAC applications and services. Another topic that will be addressed is the possibilities of automatic interface adaptation if the AAC users' needs or capabilities change over time. We will also examine the proposed symbol editors' performance on a larger set of symbols that might be more applicable to users with higher cognitive capabilities.

7 References

[1] Balandin, S. Message from the President. // The ISAAC Bulletin. 67, (2002), pp. 2.

[2] King, J. M. Complex Communication Needs and AAC. // Communication Sciences and Disorders. (2006), pp. 500-529.

[3] Stančić, Z.; Frey Škrinjar, J.; Ljubešić, M.; CarŽ., Multidisciplinary Collaboration and ICT Services for People with Complex Communication Needs. // Proceedings of the 34th Conferences on Microelectronics, Electronics and Electronic Technology / Opatija, 2011, pp. 265-270.

[4] American Speech-Language-Hearing Association Report: Augmentative and Alternative Communication. // ASHA. 33, Suppl. 5(1991), pp. 9-12.

[5] Car, Ž.; Vuković, M.; Vučak, I.; Pibernik, J.; Dolić, J. A Platform Model for Symbol Based Communication Services. // Proceedings of the 11th International Conference on Telecommunications ConTEL 2011 / Graz, 2011, pp. 141-147.

[6] ICT Systems for People with Complex Communication Needs. Alternative and Augmentative Body of Knowledge (in Croatian), University of Zagreb, 2012. http://www.ict-aac.hr/index.php/katalog-znanja (21.12.2016)

[7] Accessing Augmentative Communication for Children with Severe Physical or Multiple Disabilities. Complex Child E-Magazine, 2011, http://www.articles.complexchild.com/sept2011/00324.html (21.12.2016)

[8] Henze, N.; Rukzio, E.; Boll, S. 100 000 000 taps: analysis and improvement of touch performance in the large. // Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services. / Stockholm, 2011. pp. 133-142

[9] Himberg, J.; Häkkilä, J.; Kangas, P.; Mäntyjärvi, J. On-line personalization of a touch screen based keyboard. // Proceedings of the 8th International Conference on Intelligent User Interfaces. Miami, 2003. pp. 77-84. DOI: 10.1145/604045.604061

[10] Klironomos, I.; Abascal, J. An Introduction to the Key Issues Relating to Accessible User Interfaces. CARDIAC project. 2010. http://www.cardiac-eu.org/user_interfaces/key.htm (21.12.2016)

[11] Drobics, M.; Budweg, S.; ScheringS. Fostering social interaction with smart tvs: results from the FoSIBLE project. // Proceedings of AAL Forum 2012. Eindhoven, 2012, pp. 421-425.

[12] Jung C.; Hahn V. GUIDE - Adaptive User Interfaces for Accessible Hybrid TV Applications. // Second W3C Web and TV Workshop, Berlin, 2011.

[13] Aizpurua, A.; Cearreta, I.; Miñón, R.; Arrue, M. Adaptive User Interfaces to Assist Users Accessing General-Purpose Machines and Services. // 18th International Conference on User Modeling, Adaptation and Personalization, Hawaii, 2010, pp. 37-39.

[14] Gajos, K. Z.; Wobbrock, J. O.: Weld, D. S. Automatically generating user interfaces adapted to users' motor and vision capabilities. // Proceedings of the 20th annual ACM symposium on User interface software and technology, Newport, 2007, pp. 231-240. DOI: 10.1145/1294211.1294253

[15] Coelho, J.; Duarte, C. The Contribution of Multimodal Adaptation Techniques to the GUIDE Interface. // Universal Access in HCI, Part I, HCII 2011. LNCS. 6765, (2011), pp. 337-346.

[16] Coelho, J.; Duarte, C.; Biswas, P.; Langdon, P. Developing accessible TV applications. // Proceedings ASSETS 2011, Dundee, Scotland, 2001, pp. 131 -138.

[17] Brown, P. J.; Bovey, J. D.; Chen, X. Context-Aware Applications: from the Laboratory to the Marketplace. // IEEE Personal Communications. 4, 5(1997), pp. 58 -64. DOI: 10.1109/98.626984

[18] Dey, A. K.; Abowd, G. D.; Brown P. J.; Davies, N.; Smith, M.; Steggles, P.Towards a better understanding of context and context-awareness. // Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing / Karlsruhe, 1999, pp. 304-307.

[19] Merlin, P. A Study of the Recoverability of Computer System. Irvine, 1974.

[20] Popova, L. On Time Petri Nets. // Journal of Information Processing and Cybernetics, EIK 27(1991)4, pp. 227-244.

[21] Blagajić, I.; Šemanjski, I.; Šarić, T.; Janda-Hegediš, Ž.; Vuković, M.; Car, Ž. e-Accessible Service System: Calibrator and Communicator. // Proceedings of the 6th KES International Conference KES-AMSTA 2012 / Dubrovnik, 2012, pp. 241-250. DOI: 10.1007/978-3-642-30947-2_28

[22] ARASAAC Symbol Set. Aragonese Portal of Augmentative and Alternative Communication. http://www.catedu.es/arasaac/. (21.12.2016).

[23] Mulberry Symbol Set. OATS Open Source Assistive Software. http://straight-street.com/. (21.12.2016).

[24] Sclera Symbol Set. http://www.sclera.be. (21.12.2016)

[25] TINA Time Petri Net Analyzer, http://projects.laas.fr/tina/. (21.12.2016)

Authors' addresses

Ivan Vučak, M.sc
Sedam IT Ltd.
Koledovčina 2, 10000 Zagreb, Croatia
ivan.vucak@sedamit.hr

Professor Željka Car, Ph. D.
Faculty of Electrical Engineering and Computing,
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
zeljka.car@fer.hr

Marin Vuković, Ph.D.
Faculty of Electrical Engineering and Computing,
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
marin.vukovic@fer.hr