Configurable Templates for Assistive Technology Mobile Apps: A Block-based Programming Approach

Luís Filipe Garcia[1], João-Paulo Barros[1,2][a] and Candida Viriato[1]
[1]*Polytechnic Institute of Beja, Beja, Portugal*
[2]*Centre of Technology and Systems-UNINOVA, Caparica, Portugal*

Keywords: Special Needs Education, Mobile Computing, Programming.

Abstract: Simple computer activities have been used for a long time for cognitive and physical training in the context of rehabilitation or mental stimulation. An example is training the skills needed by users with special needs to access the computer through scanning access, which is an access method used by persons with major motor difficulties. The increased availability of less expensive mobile devices with large displays provides an ideal platform for these applications. This work-in-progress paper presents an ongoing work to empower rehabilitation therapists and special education teachers with no previous computer programming experience with a set of highly configurable apps supporting several types of activities for scanning access training. The apps are made available as open projects written in a block-based programming language. This way, they may be configurable by non-programmers while also allowing further changes depending on the programming skills of each rehabilitation therapist or special education teacher. This study intends to validate this approach among this group of users and formulate a set of guidelines concerning software architecture and organization and user interaction, to be used in the development of this kind of application.

1 INTRODUCTION

People with special needs benefit from specially tailored solutions, and scanning access training is a common activity where customization is useful. This customization can allow rehabilitation therapists and special education teachers to provide adequate training for each particular end-user in the context of rehabilitation and cognitive stimulation exercises. Moreover, since the physical and mental condition of users can change over time, the possibility of adapting training activities to the current conditions of the user is extremely important to ensure the continuity of training and to avoid technology abandonment (Phillips and Zhao, 1993). In particular, touch screens provide a platform suitable for those activities. Additionally, mobile devices, most notably tablets, are available at various prices and sizes and provide out-of-the-box touch interfaces. These characteristics make them especially suitable for supporting simple touch-based apps that are frequently used for those activities. Here is proposed the use of mobile apps built with a block-based well-documented programming language to allow a flexible degree of customization by non-programmers. This study intends to validate this approach for this group of users and, as a result, to formulate a set of guidelines concerning block-based software construction and user interface design that may be used in the development of this kind of applications.

The following sections present the background and the related work while framing this proposal motivation. Next, it presents how the apps are used and configured and show three apps as examples. Finally, it presents some future work.

2 BACKGROUND AND RELATED WORK

This section starts presenting scanning access training, which the rehabilitation activity to be supported by the proposed approach. The approach proposed in this study tries to address software customizability through end-user programming. Therefore, main concepts and related studies in these two areas are presented. Special attention was taken to two studies that

[a] https://orcid.org/0000-0002-0097-9883

Garcia, L., Barros, J. and Viriato, C.
Configurable Templates for Assistive Technology Mobile Apps: A Block-based Programming Approach.
DOI: 10.5220/0011089600003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 2*, pages 293-298
ISBN: 978-989-758-562-3; ISSN: 2184-5026

apply end-user programming to the assistive technology area in the pursuit of strategies and results that may contribute to refining the proposed approach. Finally, we present the MIT App Inventor development platform used to develop the prototypes of these customizable mobile apps.

2.1 Scanning Access Training

Access to the computer by people with major motor difficulties may have to be carried out through a mechanism called scanning. To control this mechanism, users should activate one or more switches. In the case of regular scanning, only one switch is used. The first activation of the switch will start the scanning, and then the system will automatically step through all the available options in the user interface. A second activation of the switch will select the option highlighted at that moment. Other scanning alternatives, such as step-by-step scanning, require two switches. One of the switches is used to move between options manually, and the second switch allows the user to select highlighted option.

For effective use of scanning access, to control the computer or other electronic devices, users usually undergo a training program involving activities of the following types that should be carried out in a progressive way (Cook and Hussey, 2002):

1. Cause-Effect Training – Activities where activation of a switch (or touching the screen) causes certain effects, such as coloring an image or moving an element on the screen;
2. Hit on time Training – Activities where an element moves on the screen, and the user must activate the switch at the right moment. For example, in an activity directed toward children, it may be necessary to activate the switch when a football player passes in front of the soccer goal;
3. Symbol Selection Training – Activities where the user is asked to select a communication symbol using scanning access.

A past graduation project (Silva, 2003) was developed in collaboration with the Center of Cerebral Palsy of ANONYMOUS, a computer application containing activities to train users in these three steps.

In the following work, also another graduation project, a second application was developed to allow rehabilitation therapists and special education teachers to create new activities, specifically adapted to their students (Coelho, 2007). The creation of new activities was based on pre-existing models, which these professionals completed by providing the necessary data through a wizard-like configuration interface. For example, for a cause-effect activity, they could select the sequence of images to be displayed to the user.

In this new project, a new approach will be tested, consisting of letting rehabilitation therapists and special education teachers change the program code directly to better adapt each activity to their students' needs and capabilities.

2.2 Software Customizability

Customizability is an important usability principle that consists in the modifiability of the user interface by a user (adaptability) or system (adaptivity) (Dix et al., 2003). This principle assumes even greater importance for users with special needs since it allows software and interfaces adaptation to this group of users' specific needs and capabilities.

Personalization through adaptivity is a strategy that requires a certain degree of intelligence on the part of the system. Instead of explicit configuration, the system must infer the users' needs and adapt its mode of operation based on collected knowledge. For example, options can be ordered in a menu according to their frequency of use, thus optimizing the user's effort. In assistive technologies, this strategy is followed by augmentative and alternative communication systems to predict words and sentences that users intend to speak.

In this work, we explore personalization through adaptability, meaning that the user will explicitly configure the software for training scanning access. However, instead of offering a graphical user configuration interface consisting of several dialog boxes for user configuration, an alternative is explored that allows users to change those options in the code.

This approach has been applied with some nuances in different application areas. (Barricelli et al., 2019) carried out a systematic mapping study about this strategy, considering scientific literature about End-User Development (EUD), End-Programming (EUP), and End-User Software Engineering (EUSE). They identified several different techniques that were adopted in those studies: component-based, rule-based, programming by demonstration/example, spreadsheet-based, natural language, template-based, natural language, workflow and dataflow diagrams, model-based, text-based, digital sketching, annotation-based, assertion-based, and gesture-based. The authors also identified several application domains: business and data management, web applications and mashups, smart objects and environments, games and entertainment, education and teaching, healthcare and wellness, mobile applica-

tions, interaction design, and robotics. Many studies considered a generic user because their main goal was to present a technique. However, a considerable number of studies were directed toward a specific class of users, such as office workers, managers, civil servants, architects, therapists, physicians, tourism operators, teachers, interaction designers, and researchers. The study described in this paper may contribute to improve the knowledge about the application of this approach in the area of assistive technologies.

2.3 End-user Programming and Assistive Technologies

EUP also has been applied to the assistive technologies area. Next are presented studies relating to these two areas. Methodologies and results obtained in these studies are going to be considered in the following stages of the project described in this paper. In (Barakova et al., 2013) an architecture is proposed to program robots to be used in therapies for training social skills to autistic children. To identify the requirements of this platform, therapists were engaged in the co-creation of scenarios where they would like to use robots as an augmentation to their practice. Reusability, modularity, affordances for natural interaction, and ease of use were the main principles identified to follow in end-user programming. They used the visual programming environment TiViPE (Tino's Visual Programming Environment) to create the end-user interface because it already had a modular structure and components for reading sensory information, an architecture capable of controlling robots, and meeting the challenges of the application domain. Preliminary tests showed that different user groups, including therapists with general computer skills, and adolescents with autism, could make simple training, or general behavioral scenarios, within one hour, by connecting existing behavioral blocks, and typing textual robot commands to fine-tune behaviors.

In (Carmien, 2005) authors discuss the development of a tool to guide users with cognitive difficulties in the execution of daily tasks, such as getting a bus. This tool allows familiars and professionals to create scripts presenting the steps required to complete a specific task. The tool has a dual user interface: one to be used by familiars or professionals to create or edit scripts; persons with cognitive difficulties use the other interface to accomplish specific tasks. The creation of new scripts can be based on a pre-existent script or built from scratch. The script editor follows the filmstrip metaphor to present each step of the task (prompt) graphically and allows several operations to be carried out over each prompt, such as adding an image, sound, or text note. Preliminary results showed that adopting a dual user interface to let familiars and professionals reconfigure the scripts could mitigate some of the causes for device abandonment.

Assistive monitoring systems are another category of tools that may require end-user programming. These systems are usually composed of a network of sensors and video cameras. For video and sensor output integration in a monitoring solution that end-users can program, (Edgcomb and Vahid, 2011) propose and evaluate a technique called feature extractors. This technique consists of a programming block structure that receives video input and produces the associate feature's level as output, typically a number between 0 and 100. Feature extractors were integrated into a system that authors have previously developed, the Monitoring and Notification Flow Language (MNFL) (Edgcomb and Vahid, 2012), and evaluated with users with different technological backgrounds in the construction of applications with varying levels of difficulty. Results showed that users quickly learn feature extractors, especially with very brief training via examples.

2.4 MIT App Inventor

Mobile apps are usually developed by specially trained computer programmers using complex development environments and languages. However, a well-known development platform provides a much simpler way to create mobile apps: the MIT App Inventor. The MIT App Inventor is a freely available online development platform (MIT, 2022; Patton et al., 2019). It supports the development of mobile apps targeting Android and iOS systems. It uses a friendly block-based programming language built on Google Blockly (Fraser, 2015), allowing the creation of mobile apps in a much more quick and straightforward way than is possible with the traditional professional tools based on textual programming languages. It is a popular development environment. According to the site statistics, presently, it has 14.9 million registered users from 195 countries and around 286 thousand weekly active users (MIT, 2022). Anecdotal evidence shows it is one of the leading platforms for learning computer programming in different settings and across all ages. Also, there are many teaching and learning resources, including more than twenty books.

3 APPS USAGE

The use case diagram in Figure 2 identifies three actors: (1) the IT Specialist who, together with a rehabilitation therapist or special education teacher, has the skills to build new configurable apps; (2) the rehabilitation therapist and special education teacher that can configure existent apps; and (3) the end-user that simply has to use the configured and generated apps.

Two use cases were identified regarding the rehabilitation therapist or special education teacher: simple and advanced configuration. In the simple configuration use case, it is only necessary to change a minimal and documented set of blocks and know how to generate the app and deploy it to the mobile device. It is essential to notice that the changes in the set of blocks are simple enough to be quickly learned by any user with minimal computer usage knowledge, resulting from the regular use of a desktop browser. Only the advanced configuration demands programming skills, although in the context of a block-based programming language. This configuration can be as sophisticated as the user skills allow. In fact, it can even be used to build new apps: the user becomes an app developer, similarly to the IT specialist.

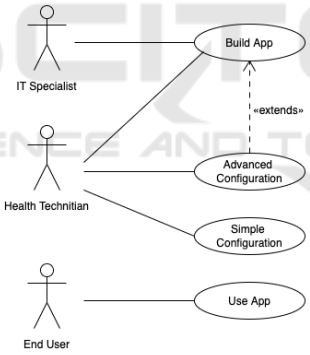

Figure 1: Use case diagram.

The rehabilitation therapist or special education teacher needs access to an MIT App Inventor online account. The projects should already be available, as in the screenshot on the top-right of Figure 2, or be imported from the local computer. Then, it is possible to choose one project as illustrated in the bottom left of Figure 2 where the ”Designer” screen is shown. This is where the user interface is built.

In the simplest use case, the rehabilitation therapist or special education teacher proceeds immediately with the ”simple configuration” by going to the ”Blocks” screen (bottom right in Figure 2) where the code blocks are. There, the user will change only a minimal part of the blocks, namely those that allow the simple configuration for the app that will then be generated. More significant changes can be made if the rehabilitation therapist or special education teacher already knows some block-based programming, as represented by the ”Advanced Configuration” use case.

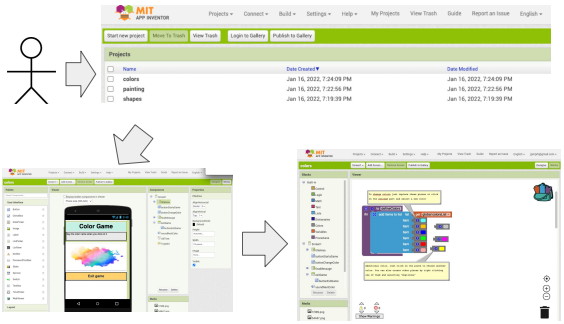

Figure 2: How the apps are configured.

Next, three simple apps are briefly present. The ”Shapes Game” app (see Fig. 3(a)) shows one shape, in full screen, sequentially. The user must touch the device’s screen to see the next shape.

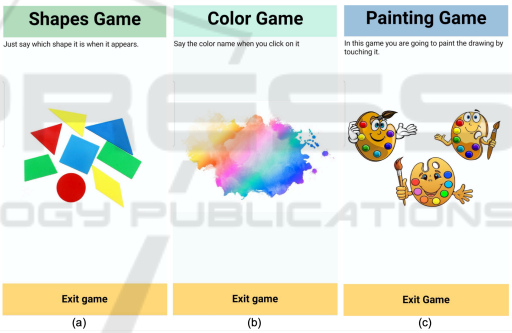

Figure 3: (a) Shapes, (b) Colors, and (c) Painting apps start screens.

The configuration is straightforward: the rehabilitation therapist or special education teacher only has to change the names of the item blocks in a blocklist as seen in Figure 4. These images are inserted in the ”Designer” page, which is effortless: select the files from the local computer using the desktop operating system user interface.

The ”Color Game” app is very similar: the user has to touch the device’s screen to see the next color. Figure 3(b) shows the start screen.

The configuration does not require external files: the colors are readily specified by color blocks that are visual values in the block-based programming language (see Figure 5).

A third example is the ”Painting App” (see Fig. 3(c)). In this app, a drawing is painted one touch at a time. The rehabilitation therapist or special educa-

Figure 4: Configuration for shapes identification app.

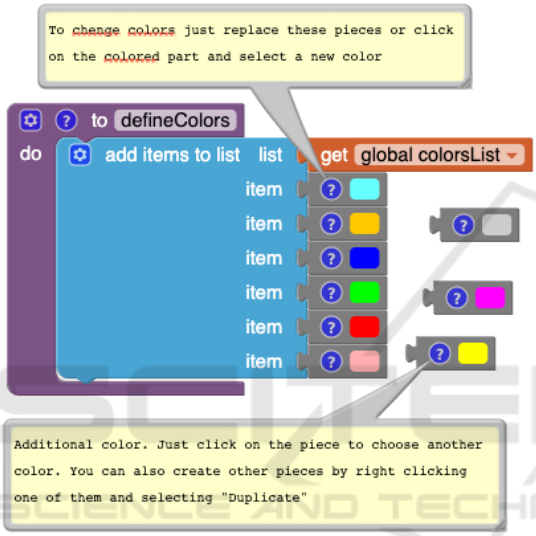

Figure 5: Configuration for colors identification app.

tion teacher has to configure the app by providing a sequence of increasingly painted images. Hence, the configuration is similar to the "Shapes Game" app: blocks with the image file names have to be added to a block list, as illustrated in Fig. 6.

4 CONCLUSIONS AND FUTURE WORK

The use of end-user programming in the context of assistive technologies is not new. Here, some works in the area are presented. However, no previous work has used block-based programming for mobile devices to our best knowledge.

The developed configurable apps already allow us to conclude that the MIT App Inventor online development platform can effectively build highly configurable mobile apps by users without computer programming skills. Also, those same users can take advantage of the platform proved adequacy for learning computer programming to progressively learn how to make the apps closer to the end-user needs.

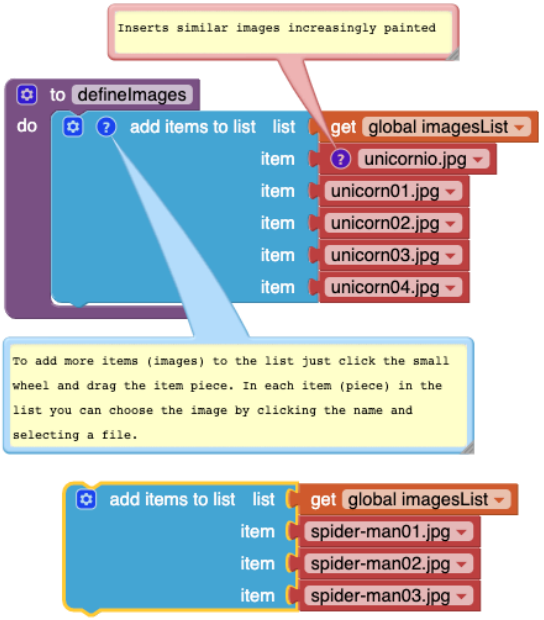

Figure 6: Configuration for painting app.

In future work, this approach will be tested with a more extensive set of configurable apps and a set of rehabilitation therapists and special education teachers with distinct levels of computer use proficiency. Those tests will provide additional input to improve the design and usability of this type of app.

REFERENCES

Barakova, E. I., Gillesen, J. C., Huskens, B. E., and Lourens, T. (2013). End-user programming architecture facilitates the uptake of robots in social therapies. *Robotics and Autonomous Systems*, 61(7):704–713.

Barricelli, B. R., Cassano, F., Fogli, D., and Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137.

Carmien, S. (2005). End user programming and context responsiveness in handheld prompting systems for persons with cognitive disabilities and caregivers. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1252–1255.

Coelho, M. (2007). Configurable scanning training (graduation project). Technical report, ***anonymous institution***.

Cook, A. M. and Hussey, S. M. (2002). *Assistive technologies*. Elsevier.

Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2003). *Human-computer interaction*. Pearson Education.

Edgcomb, A. and Vahid, F. (2011). Feature extractors for integration of cameras and sensors during end-user programming of assistive monitoring systems. In *Proceedings of the 2nd Conference on Wireless Health*, pages 1–2.

Edgcomb, A. D. and Vahid, F. (2012). Mnfl: the monitoring and notification flow language for assistive monitoring. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 191–200.

Fraser, N. (2015). Ten things we've learned from blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, pages 49–50.

MIT (2022). MIT App Inventor. http://appinventor.mit.edu/. Accessed: 2022-01-17.

Patton, E. W., Tissenbaum, M., and Harunani, F. (2019). Mit app inventor: Objectives, design, and development. In Kong, S.-C. and Abelson, H., editors, *Computational Thinking Education*, pages 31–49. Springer Singapore, Singapore.

Phillips, B. and Zhao, H. (1993). Predictors of assistive technology abandonment. *Assistive technology*, 5(1):36–45.

Silva, A. (2003). Scanning training (graduation project). Technical report, ***anonymous institution***.