GUIDL as an Aiding Technology in Programming Education of Visually Impaired

Mario Konecki
Faculty of Organization and Informatics, Varaždin, Croatia
Email: mario.konecki@foi.hr

***Abstract*—Programming education is of vital importance because of growing needs for programming professionals. Even though efforts have been made in order to improve the quality of knowledge and acquired programming skill of computer science students the educational systems are still confronted with rather high failure and dropout rates. There are several possible approaches that can be taken in education of programming novices which are described and discussed in this paper. The problems in programming education are even more prominent in the case of programming novices who are visually impaired. Some of educational problems are similar to general problems in programming education and some are specific in the case of visually impaired students and require a suitable approach in order to be resolved. Several existing aiding tools and approaches are presented in this paper and a new solution in the form of GUIDL (Graphical User Interface Description Language) system is proposed as a suitable mean for aiding visually impaired in their programming education. The results of conducted research about the effectiveness of proposed solution are also presented and discussed.**

***Index Terms*—visually impaired, programming, education, aiding technology, GUIDL**

I. Introduction

One thing that is common knowledge among both teachers and students of computer science studies is that programming is hard to learn [17]. This fact is especially true when talking about introductory programming courses. Programming courses experience rather high failing and dropout rates [3; 20; 26] which have also caused a fear and unwillingness to deal with programming among general student population. For years there have been efforts to find a suitable programming languages and methods that would address this problem in a proper way but to this date the situation regarding programming education has not changed much. The programming concepts along with a need to grasp a rather large amount of different syntactic elements have proven to be hard for most of students.

Various research has been conducted in order to conclude about the reasons for this kind of state and in order to propose some means of its resolution. Experience has shown that students tend to lose track and stop following lectures in active way in some point in which more complex syntax and concepts are introduced. In order to try to bring these concepts closer to students various visualization techniques and tools have been proposed and they have proved to be beneficial in greater or lesser amount to overall student population [8; 10; 23; 24]. Another existing approach towards introductory programming problems has stated the importance of prerequisites to programming which include problem-analysis and problem-solving skills that promote algorithmic way of thinking that is the basis for programming activities [14; 16].

Another already mentioned problem that students encounter is the problem of not being able to keep up with lectures which include many programming concepts and a lot of syntax which are both abstract in nature. Most of introductory programming languages are rather large and of broad purpose. They also lack any real insight into the process of their programs execution [5]. The program code is translated into result omitting details of its execution process. Also, most of tasks that can be encountered in introductory programming are of mathematical nature and are not intuitive or in many cases motivating or interesting for programming novices.

There are 3 possible approaches that are typically used in programming education [4]:

- incremental approach
- the sub-language approach
- the mini-language approach

Incremental approach includes dividing the overall scope of programming course into incremental parts which are then taught one by one. In this approach every next part uses all elements that have been taught in previous part until the whole language has been presented and learned by students. The sub-language approach takes only a part of overall programming language in order to ease the process of learning for students. In this way the students are able to focus on concepts in more details since they are not burdened with quantity of elements they need to comprehend. Only when the students truly learn and master this sub-language it is possible for them to proceed to more complex tasks. The mini-language is similar approach as the sub-language approach but it differs in the sense that it includes the newly developed language which purpose is mainly to teach students about programming principles. One such language that is well known is Logo. This kind of languages usually offer two important aspects: simplicity and visual nature which both promote better focus, easier

doi:10.4304/jcp.9.12.2816-2821

comprehension and better understanding of programming concepts.

The programming education for visually impaired is faced with some similar and some specific obstacles compared to general programming education. Since visual component is not an option the simplicity of programming language and its use is of vital importance. Visually impaired programming novices often have troubles in perceiving the overall program code which consists of a lot of variables, loops and other programming constructs. Some of ways that are mostly used by visually impaired programmers to better mark certain program elements are using descriptive variable, array and structure names and using program comments to describe certain points and elements of various program's code.

In this paper GUIDL (Graphical User Interface Description Language) [11, 12; 13; 15] system is proposed as a mean of providing visually impaired with a suitable way of learning some programming concepts that enables them to develop graphical user interfaces by using an adapted syntax and some aiding concepts. The research results about the suitability of GUIDL system for usage in visually impaired programming novices programming education are also presented and discussed. Another important aspect of GUIDL system is that it enables visually impaired to translate the developed graphical user interfaces into chosen technology format that can be used in actual programming languages and their environments.

The interest of visually impaired for perusing programming careers is present and something that needs to be taken into consideration. Over 130 blind programmers are members of the American Foundation of the blind programmers [1]. Programming is also stated in several European reports as a promising career opportunity for visually impaired [28; 29]. The same was also suggested by other authors since the very beginning of computer industry [25]. The importance of computer technology in creating equal opportunities in educational systems is also denoted as one of important aspects for visually impaired [7].

II. Existing Educational Tools And Approaches

There are several existing tools and approaches that are aimed at aiding visually impaired in programming education.

APL (Audio based Programming Language) [21] is a tool that is aimed at providing a mean of teaching visually impaired about the basic programming concepts rather than being a functional programming language for professional programming purposes. APL is made of two layers which are Audio Interface and Programming Logic. Audio Interface is made of a circular list of commands by which usage it is possible to create variables, conditions and other programming elements. Another part of Audio Interface is Query that is used to actually define the variables and input/output content in audio or text format. Programming Logic consists of four possible states that are used to create programs:

- Run program
- End loop/condition
- Delete last command
- Save Command and Verify next step

APL provides aiding tool for learning the most basic programming concepts in its special ecosystem which is not connected to other programming languages which are mostly used in practice. The basic elements of APL are shown in Fig. 1 [21].

Javaspeak tool [22] tries to describe the context of the written program in the sense of syntactic and organizational structure of the program. Syntactic structure denotes the way in which the program is composed in order to work and organizational structure denotes the way in which the programer arranges the program parts in order to keep the track of what the overall program does. Javaspeak uses audio chanel in order to describe both components. Javaspeak also uses several other functions that aid better understanding of program code such as for example adding certain description text at some keypoints of program code in order to better stress the significance and meaning of some program code parts.

WAD (Wicked Audio Debugger) [19] is a debugger made for Visual Studio in order to enable visually impaired to better understand and perceive the dynamic behavior of developed programs.

SODBeans (the Sonified Omniscient Debugger in Netbeans) [19] is a special module that includes compiler, debugger and accessibility features which is developed for NetBeans IDE. SODBeans enhances accessibility by adding a variety of narative features to its compiler/debugger. SODBeans is shown in Fig. 2 [30].

Although not strictly programming aiding tool TeDUB (Technical Diagram Understanding for the Blind) [19] is another tool that deals with an important aspect related to development of programming applications. TeDUB is a system that enables visually impaired to understand the given UML diagrams which are represented in a form that combines textual and audio representation.

Java Accessibility API enables development of Java applications which use swing classes that are compatible with aiding technology for visually impaired [6] and as such can be used for educational purpose. Java Accessibility API uses Java Access Bridge technology in order to be visible to Windows assistive technologies. Java Access Bridge Architecture is shown in Fig. 3 [31].

Figure 1. APL (Audio based Programming Language) tool.

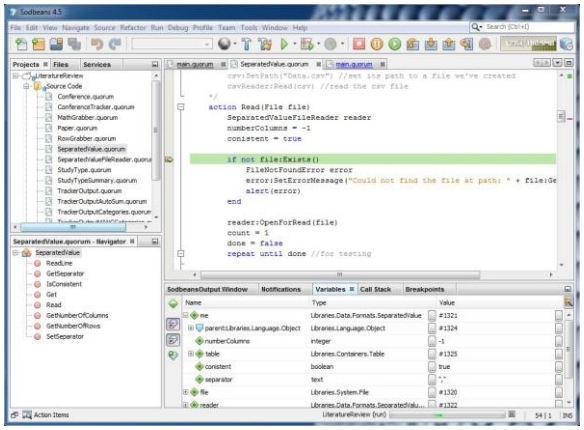

Figure 2. SODBeans.

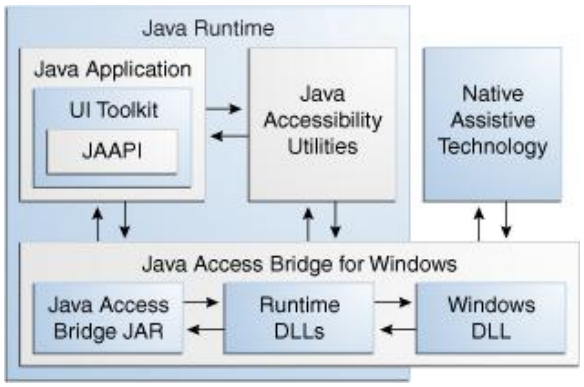

Figure 3. Java Access Bridge Architecture.

This solution uses MOST (The MObile Slate Talker) [2] which is a Java environment that is especially made to be suitable for visually impaired. It enables development of applications in which the graphical output is deliberately omitted and replaced with audio. In the same way the standard input is replaced by a Braille slate.

III. GUIDL SYSTEM

GUIDL system has been developed as an aiding tool that enables visually impaired to develop graphical user interfaces in a simple and easy way with a number of aiding concepts that make creation of graphical interfaces simpler and easier. The same principles are important in education of visually impaired since making things simpler and more visible and interesting are mentioned as important in all students education process and simplicity is the most important aspects in education of visually impaired about programming since visualization is not possible.

GUIDL system is based on GUIDL language that is developed as a simple and aiding programming language for development of graphical user interfaces for visually impaired. The aiding technology is in most cases developed based on one of models that support the development of assistive technology. The GUILD system is build based on the CAT (Comprehensive Assistive Technology) model [9] which provides assistive technology for selected activities to be performed by certain persons or users in a particular context of interest. The CAT model is shown in Fig. 4 [15]. GUIDL system is based on its core GUIDL language and appropriate translation mediators that are used to translate the GUIDL code into appropriate form that can be included into actual projects in the form of graphical user interfaces of chosen programming language and its development environment. The GUIDL conceptual model is shown in Fig. 5 [15].

All code written in GUIDL is processed by GUIDL scanner, lexer and parser in order to detect syntactic as well as logical layout errors. Such processed code is then translated into chosen programming language environment format which can be included into particular project. GUIDL system is not designed to be isolated ecosystem but rather aiding technology that aids visually impaired in creation of graphical interfaces which is something that is not sufficiently supported by development environments and existing aiding technology that visually impaired computer users use.

Compared to previously mentioned aiding educational tools GUIDL system enables creation of graphical user interfaces which has several important aspects and benefits for visually impaired programming novices:

- it includes visually impaired in the part of overall software development that has not been sufficiently accessible to them before
- it enables visually impaired to learn the basics of coding through writing the code that is used to build graphical user interfaces
- it produces visible results that visually impaired can share with their friends and colleagues which provides more substantial and interesting result for beginners than coding of purely textual programming assignments

Partial GUIDL grammar in EBNF [27] illustrates the simplicity of GUIDL syntax [15]:

project = projectcode, controlname, form;
projectcode = 'Project ' | 'project ';
form = formcode, controlname, formattributes, [controldeclarations], formend;
formcode = 'Frm ' | 'frm ';
formend = ('End' | 'end'), [eol];
controlname = qoute, word, qoute, eol;
word = alphabeticcharacter, {alphabeticcharacter | digit};
formattributes = frmcommonattributes, windowstateattribute, {colorattribute};
frmcommonattributes = textattribute, frmrestcommonattributes;
frmrestcommonattributes = frmsizeattribute, locationattribute;
frmsizeattribute = sizecode, (frmsize | frmwidth, ws ,frmheight), eol;
locationattribute = locationcode, xposition, ws, yposition, eol;
frmsize = 'frmsize1' | 'frmsize2' | 'frmsize3'
locationcode = 'Location = ' | 'Location=' | 'location = ' | 'location=';
xposition = 'left' | 'center' | 'right';
yposition = 'top' | 'middle' | 'bottom';

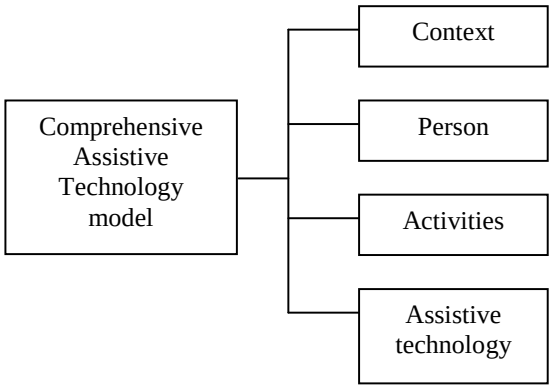

Figure 4. CAT (Comprehensive Assistive Technology) model.

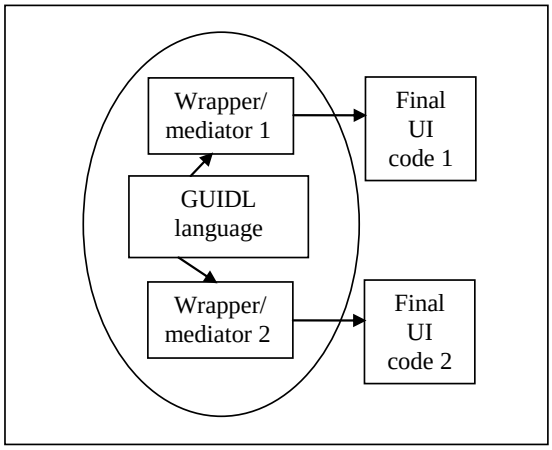

Figure 5. Conceptual model of GUIDL system.

GUIDL syntax is made to be simple and easy to use but another important aspect of GUIDL system are its aiding concepts that are used to enable visually impaired to created graphical layouts in a more suitable and easier way. These aiding aspects include [15]:

- Predefined gradual sizes of forms
- Predefined gradual sizes of graphical elements
- Predefined width/height attribute values
- Division of forms into quadrants
- Possibility to position graphical elements into one of form quadrants
- Possibility to define the position offset of forms
- Possibility to define the position offset of graphical elements
- Detection of problems with position of graphical elements (graphical element out of form boundaries)
- Automatic correction of problems with form dimension and position (form out of screen boundaries)

By using mentioned aspects visually impaired are able to construct graphical user interfaces in a quick and easy manner which promotes adoption of coding skills and gives an interesting result that is more concrete and engaging for programming novices than typical mathematical tasks that are usually used in introductory programming courses.

IV. GUIDL AS EDUCATIONAL AIDING TECHNOLOGY

Because of its simplifications and aiding nature as well as because of the results that GUIDL system produces in the form of graphical user interfaces that can be included into real projects in actual programming environments and shared with other sighted colleagues GUIDL system represents a suitable tool for visually impaired programming novices that is able to provide an appropriate first programming steps in a way that will maintain programming novices' focus and reduce their fear of programming by making things easier, more interesting and not overwhelming.

In order to test whether GUIDL system is suitable for visually impaired programming novices a research among 32 visually impaired participants has been conducted. 14 participants were women and 18 were men. Average age of participants was 25 with the youngest being 21 and the oldest 34. All included participants were active computer users but only 7 of them have had any programming experience. This fact was so because of the intent to see whether GUIDL system is simple enough for complete programming novices as well as for those with some programming experience.

GUIDL system has been given to all participants and has also been presented to them in the form of descriptive instructions that they were able to study on their own convenience as well as in the form of guided lectures and practical assignments that the participants were instructed to solve with step-by-step guidance.

Upon completing the introductory lectures and assignments all participants were given several tasks of gradual difficulty and different elements and aspects were included in every task. The observations were also made during the introductory lectures in order to see how well and how quickly are the participants able to do the assignments that were introduced to them.

The overall conclusion was that all participants were able to complete given examples during the introductory lessons with some minor difficulties that were quickly resolved by a few additional clarifications. The overall average of asked questions during the introductory assignments was 3 questions per participant. The participants were given 4 tasks to solve on their own and the results have shown that 27 participants were able to complete all 4 tasks without problems and 5 participants have solved the tasks but required some minor help on some syntactic details. The results of given tasks solving indicate that GUIDL system is indeed suitable for programming novices education and that it is also suitable as an aiding technology in graphical user interface creation for visually impaired. The observations have shown that participants have been successfully aided by all aiding aspects included into GUIDL system which have helped them to created elements with desired sizes and desired layout positions.

The participants were also given a questionnaire in order to conclude about their personal experience and attitude towards GUIDL system. The results of conducted questionnaire research are given in Table I.

TABLE I.
RESULTS OF CONDUCTED QUESTIONNAIRE RESEARCH

Questionnaire item	Mean	Std. dev.
GUIDL aids me in understanding of programming skill	4.322	0.518
GUIDL aids me in designing of graphical user interfaces	4.804	0.435
I am able to create graphical user interfaces in a way that is easy and intuitive for me	4.671	0.393
GUIDL makes programming more interesting for me	4.455	0.467
GUIDL enables me to share my ideas about graphical user interfaces with my colleagues	4.838	0.294
I would use GUIDL in my real projects	4.358	0.672
I would recommend GUIDL to my colleagues that are interested in learning programming	4.744	0.429

The results of given questionnaire have confirmed that participants perceive GUIDL as a useful and appropriate aiding tool for the purpose of programming education and creation of graphical user interfaces that increases the motivation of its users to deal with programming and design activities.

V. CONCLUSION

Programming is more important than ever because of constant demand for more professional programmers. Programming courses however experience rather high failure and dropout rates. Various approaches and tools have been proposed in order to deal with this issue but all efforts gave limited results. Programming is recognized as a promising career opportunity for visually impaired with design of graphical user interfaces as an integral part of this profession.

Visualization is one of the most common means that have been proposed in order to bring programming closer to students. Since visualization is not possible in the case of visually impaired students a simplicity and intuitive aiding tools are the most important factor in promoting programming among visually impaired. Several existing aiding solutions that can be used in education of visually impaired students in the field of programming have been presented.

The solution named GUIDL (Graphical User Interface Description Language) that promotes simplicity and visual results in the form of graphical user interfaces that can be shared with other students has been proposed as an efficient mean of aiding visually impaired in their programming education. In order to test GUIDL system and to conclude about its suitability for programming education of visually impaired the research has been conducted in a form of testing how easy and how quickly it is possible for visually impaired students to create graphical user interfaces and learn about basic coding skills. The research included education about GUIDL system with guided examples and test tasks as well as questionnaire research that has been conducted in order to find about the attitude of participants towards the GUIDL system.

The results have shown that GUIDL system is useful and suitable aiding tool that is perceived as interesting because of its graphical output that can be shared as ones own idea in the area that was not sufficiently accessible before to visually impaired and that it can be used as an aiding tool in programming education of visually impaired. Adding new features to GUIDL system and expanding its possible areas of use will be a part of future research.

REFERENCES

[1] S. Alexander, "Blind Programmers Face An Uncertain Future", *ComputerWorld*, vol. 32, no. 44, pp. 86-87, 1998.
[2] A. Arato, Z. Juhasz, P. Blenkhorn, G. Evans, G. Evreinov, "Java-powered braille slate talker", in *Proceedings of ICCHP 2004: Computers Helping People with Special Needs*, 2004, vol. 3118, pp. 506-513.
[3] J. Bennedsen, M.E. Caspersen, "Failure rates in introductory programming", *ACM SIGCSE Bulletin*, vol. 39, no 2., pp. 32-36, 2007.
[4] P. Brusilovsky, A. Kouchnirenko, P. Miller, I. Tomek "Teaching Programming to Novices: A Review of Approaches and Tools", in *Proceedings of ED-MEDIA 94 - World Conference on Educational Multimedia and Hypermedia*, 1994, pp. 103-110.
[5] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko, P. Miller, "Mini-languages: a way to learn programming principles", *Education and Information Technologies*, vol. 2, no. 1, pp. 65-83, 1997.
[6] R. F. Cohen, A. V. Fairley, D. Gerry, G. R. Lima, "Accessibility in introductory computer science", *ACM SIGCSE Bulletin*, vol. 37, no. 1, pp. 17-21, 2005.
[7] G. Douglas, "ICT, education, and visual impairment", *British journal of educational technology*, vol. 32, no. 3, pp. 353-364, 2001.
[8] S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, K. Sutton, D. Tolhurst, J. Tutty, "Predictors of success in a first programming course", in *Proceedings of the 8th Australasian Conference on Computing Education*, 2006, vol. 52, pp. 52:189-196.
[9] M. A. Hersh, M. A. Johnson, "Assistive Technology for Visually Impaired and Blind People", Springer, 2008.
[10] M. Hu, "Teaching novices programming with core language and dynamic visualization", in *Proceedings of the 17th NACCQ*, 2004, pp. 94-103.
[11] M. Konecki, R. Kudelić, D. Radošević, "Challenges of the blind programmers", in *Proceedings of the 21st Central European Conference on Information and Intelligent Systems*, 2010, pp. 473-476.
[12] M. Konecki, A. Lovrenčić, R. Kudelić, "Making Programming Accessible to the Blinds", in *Proceedings of the 34th MIPRO International Convention on Computers in Technical Systems*, 2011, pp. 180-184.
[13] M. Konecki, "A New Approach Towards Visual Programming for the Blinds", in *Proceedings of the 35th MIPRO International Convention on Computers in Technical Systems*, 2012, pp. 1076-1081.
[14] M. Konecki, M. Petrlić, "Main problems of programming novices and the right course of action", in *Proceedings of the 25th Central European Conference on Information and Intelligent Systems*, 2014, pp. 116-123.
[15] M. Konecki, "Inclusion of visually impaired in graphical user interface design", in *Proceedings of the 17th International Multiconference Information Society - Intelligent Systems*, 2014, pp. 54-57.
[16] M. Konecki, V. Mrkela, "Algorithmic thinking and animated interactive presentation of sorting algorithms in

education of students", in *Proceedings of the 17th International Multiconference Information Society - Education in Information Society*, 2014, pp. 105-112.
[17] M. Konecki, "Problems in programming education and means of their improvement", *DAAAM International Scientific Book 2014*, in press.
[18] N. Markus, Z. Juhasz, G. Bognar, A. Arato, "How Can Java Be Made Blind-Friendly", in *Proceedings of ICCHP 2008: Computers Helping People with Special Needs*, 2008, vol. 5105, pp. 526-533.
[19] S. Mealin, E. Murphy-Hill, "An exploratory study of blind software developers", in *Proceedings of the 2012 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2012, pp. 71-74.
[20] U. Nikula, O. Gotel, J. Kasurinen, "A motivation guided holistic rehabilitation of the first programming course", *ACM Transactions on Computing Education (TOCE)*, vol. 11, no. 4, art. 24, 2011.
[21] J. Sánchez, F. Aguayo, "Blind learners programming through audio", in *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, 2005, pp. 1769-1772.
[22] A. C. Smith, J. M. Francioni, S. D. Matzek, "A Java programming tool for students with visual disabilities", in P*roceedings of the fourth international ACM conference on Assistive technologies*, 2000, pp. 142-148.
[23] P. A. Smith, G. I. Webb, "The efficacy of a low-level program visualization tool for teaching programming concepts to novice C programmers", *Journal of Educational Computing Research*, vol. 22, no. 2, pp. 187-216, 2000.
[24] J. Sorva, V. Karavirta, L. Malmi, "A review of generic program visualization systems for introductory programming education", *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 4, art. 15, 2013.
[25] T. D. Sterling, M. Lichstein, F. Scarpino, D. Stuebing, "Professional computer work for the blind". *Communications of the ACM*, vol. 7, no. 4, pp. 228-230, 1964.
[26] A. Yadin, "Reducing the dropout rate in an introductory programming course", *ACM Inroads*, vol. 2, no. 4, pp. 71-76, 2011.
[27] "Information technology - Syntactic metalanguage - Extended BNF", ISO/IEC 14977, 1996, available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html, accessed: 24th October 2014.
[28] bfi Steiermark, "European Labour Market Report", available at http://eurochance.brailcom.org/download/labour-market-report.pdf, accessed: 24th October 2014.
[29] "The employment of blind and partially-sighted persons in Italy: A challenging issue in a changing economy and society", available at http://www.euroblind.org/media/employment/employment_Italy.doc, accessed: 24th October 2014.
[30] http://sourceforge.net/projects/sodbeans/, accessed: 24th October 2014.
[31] https://docs.oracle.com/javase/accessbridge/2.0.2/introduction.htm, accessed: 24th October 2014.