Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study

Khaled Albusays
Computing and Information Sciences Program
Rochester Institute of Technology
Rochester, NY 14623
kla3145@rit.edu

Stephanie Ludi
Software Engineering Department
Rochester Institute of Technology
Rochester, NY 14623
salvse@rit.edu

ABSTRACT

Without understanding the programming difficulties faced by developers with visual impairments, the research community cannot begin to work on effective solutions to overcome these potential problems. This paper will describe our initial empirically based study to identify the problems blind software developers face. We analyzed 69 survey responses with blind developers in an effort to identify the aspects that are indeed a challenge to software development. The results indicate a number of difficulties, workarounds, and basis requirements that will serve as domain and stakeholder understand.

CCS Concepts

•**Human-centered computing** → **User centered design;** •**Social and professional topics** → Assistive technologies;

Keywords

Accessibility; Elicitation; Programmers; Visual Impairment; Programming Challenges; Blind Programmers

1. INTRODUCTION

The unstated assumption by industry and academia regarding software developers is that they are sighted. While sighted developers are able to quickly develop code, blind developers can face difficulties in programming activities [2]. The problem is that developers rely on the visual presentation of the code displayed within the Integrated Development Environments (IDEs). This programming software, which has features that require sight, presents obstacles for developers with visual impairments.

A screen reader is a common tool used by blind individuals to access information on the computer displays. The software was designed to present information displayed on the screen verbally, assuming that the software or the website is designed to accommodate the screen reader software.

CHASE'16, May 16 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897586.2897616

In the case of programming, a screen reader reads system menus, dialog boxes, tree views, as well as provides access to other system features. The software was designed in linear fashion that reads code one line at a time preventing blind developers from getting an overview of the entire code.

Our research is focused on developing tools to support blind programmers and students in the field of computer science. To initiate the process, we are working to understand the user population, as well as their needs, challenges, and strategies in terms of programming and related software engineering tasks. The process of elicitation was carefully designed and revised in order to accommodate both the users and the authors. We sought to answer the following questions about programming challenges faced by developers with visual impairments:

1. *What are the popular IDEs and programming languages that blind developers use?*
2. *What assistive tools do blind developers use when programming?*
3. *What are the difficulties faced by blind developers when developing code?*
4. *How do blind developers use workarounds to solve programming challenges?*

2. BACKGROUND AND RELATED WORK

Currently, the research that highlights difficulties faced by blind developers is limited. Mealin et al. [3] conducted an interview study with eight blind developers who have different level of experiences to uncovered programming challenges. The interview results indicated that several blind developers do not use tools and features built into the IDEs. It is unclear if they are unaware of the variety of tools offered within the IDEs, find the tools to be too complex, or if the tools were not easily accessible. In our study, we attempted a more detailed study of the programming difficulties encountered by 69 blind developers.

There have been previous efforts aimed at creating accessible tools for individuals with visual impairments. Stefik et al.[6] created Sodbeans, a programming IDE that relies on using audio cues to convey information to developers who are blind. The IDE was built to help blind individuals learn how to program using audible cues in browsable hierarchical manner. We believe this approach could be applied to any assistive tools to help blind developers understand and navigate more efficiently in IDEs. The authors of Sodbeans

did not evaluate the efficiency based on the APL, but rather on the student's ability to master the concepts of programming. The authors found that non-speech audio was very helpful and students were able to understand the majority of the cues. Vickers et al. [7] also confirmed the usefulness of auditory cues by sighted developers and how it can help find bugs.

Stefik et al. explored the usability of audio cues to show the lexical scoping relations between each statement in the program [5]. The relationship between statements was found to be dynamic, and different cues were played when a change in scope was detected. It seems important to consider the use of short cues and how to integrate them into the IDEs via an assistive plug-in.

Baker et al. [2] created an Eclipse plug-in called StructJumper to help blind developers quickly understand and navigate through large amounts of code. Within this plug-in are two features (TreeView and Text Editor) which are used to create a hierarchical tree based on the nesting structure of Java. It is the same concept used by Smith et al. [4] to help blind developers recognize code in hierarchical manner with the use of the tree-like structure. Our future research on code navigation will seek to expand on this work, but only after a more in-depth exploration of code navigation difficulties. We will also collect a list of user requirements and evaluate existing navigation tools.

3. METHODS

3.1 Survey Design

We created an initial survey and conducted a pilot test in order to validate our questionnaire. Our pilot testers were experienced developers (+5 years) with no vision disabilities. The survey design was modified upon their comments and feedback. For example, the survey wording was changed to more familiar terms as well as adding other common programming languages to multiple questions.

We estimated the completion time of the survey at 10 minutes. There were 15 questions in our survey, 11 multiple choice, 3 open-ended, and 1 Likert Scale [1]. We defined the survey questions to inquire about how programming was learned, level of experience, visual acuity, visual perception, challenges, workarounds, assistive tools, and the type of development tools.

3.2 Sampling

In this work, we needed to find developers with visual impairments. As this is a subgroup of a limited and geographically dispersed population, we decided to use a snowball sampling technique. We decided to contact individuals who met the criteria and asked them to forward the survey to those who possess the necessary traits. The potential respondents were contacted via email invitations as well as posts in "blind individuals" groups on Google Hangouts, LinkedIn, and AppleVis (a community for blind and low-vision users of Apple's products).

3.3 Procedure and Response Rate

In order to eliminate geographic restrictions, we decided to set up an online survey through the Rochester Institute of Technology survey system. The system was designed to encompass all disabilities that affect access to the survey system. Participant response rates could not be calculated as we could only monitor the total number of responses submitted. The actual time for the survey completion could not be measured. The survey was open for more than two months.

3.4 Participants

The survey was taken by a total number of 69 participants, all of whom were blind developers. Nearly all 62 (89.86%) of the participants were male, 6 (0.09%) were female, and 1 participant decided not to answer. The mean age in our sample was 35.39 years, with a standard deviation of 13.55 years. The lowest age captured in our sample was 18 where the highest age of an individual was 68. Our survey sample showed variation in the visual acuity among the 69 respondents. About 29 (42.02%) of the participants were totally blind, followed by 25 (36.23%) who had light and shadow sensitivity, 12 had vision but needed corrective lenses (17.39%), 2 had macular degeneration (0.03%), and 1 was totally blind in one eye (0.01%). In regards to the visual perception of the 69 respondents, 43 had light perception, 26 had shadow perception, 22 had movement perception, and 16 had color perception.

4. RESULTS

The results section has been organized by the developers' background, tools used, assistive technology, and the challenges faced in software development.

4.1 Developer Background

Participants were asked to clarify the method used to learn programming. About 40 (57.97%) were self-taught, 28 (40.58%) attended schools, and 1 (0.01%) respondent did not answer the question.

We also asked participants to rate their levels of expertise in various programming languages. Table 1 shows respondents' experiences in various programming languages. The use of Python was expected as many undergraduate computer science use Python. It also has gained wider popularity among many STEM disciplines. The use of Python itself is interesting given that blind developers can dynamically inspect and change their programs.

4.2 Development Tools and Platforms

We asked participants to indicate their development tools, development platforms, and the target platforms for their work. About 49 (71.01%) use the Windows environment to write code, where 15 (21.74%) use Mac OS X, and 14 (20.29%) use Linux. Less common environments included IBM Mainframe, Motorola, micro-controllers embedded C, and Unix. In regards to the target platforms, 39 (56.52%) of the respondents developed applications that run on Microsoft. 13 (18.84%) people developed applications for Linux, 10 (14.49%) for iOS, 7 (10.14%) for Mac OSX, and 5 (0.07%) for Android. In terms of the development tools used, the most preferred editor is Eclipse (31 people, or 44.92%), followed by Microsoft Visual Studio (28 people, or 40.58%), Xcode (17 people, or 24.64%), Emacs (3 people, or 0.04%), and Netbeans (2 people, or 0.02%). Eclipse was expected due to its common adoption in undergraduate Computer Science programs, as well as it being open source.

4.3 Assistive Technology

Table 1: Level of expertise in various programming languages

Level of Expertise	List of Programming Languages									
	Java	*C*	*C#*	*C++*	*Objective-c*	*Python*	*Ruby*	*Perl*	*JavaScript*	*Php*
None	22	16	30	22	42	22	44	38	20	26
Novice	16	15	16	16	11	18	11	14	17	13
Intermediate	17	18	8	16	4	16	3	5	22	15
Expert	12	14	9	9	5	9	3	5	7	10

The use of software and hardware-based assistive technology is integral to programming and related tasks. The use of a screen reader (e.g. VoiceOver, JAWS) is very common among participants, whereas several of the blind developers prefer to use refreshable Braille display when programming. A refreshable Braille display, a hardware device, translates a single line of text that is displayed on screen to a single line of Braille that can be ready by touch. Braille displays are very expensive and require the user to know Braille, which can be a limitation to blind developers.

In regards to the types of aids that respondents use for assistance with programming, all of the respondents indicated that they do utilize Screen Reader (69 people, or 100%). Braille Display is used by 30 people (43.48%), magnification software is used by 7 people (10.14%), and large fonts used by 6 people (8.70%).

4.4 Open-Ended Responses

The survey contained three open-ended questions designed to elicit responses regarding the use of text-editors, challenges, and workarounds. To analyze these results, we applied a coding scheme categorized by: limited accessibility aids in IDEs, code navigation, diagrams, debugging and user interface layout, seeking sighted assistance, and workaround techniques.

4.4.1 Limited Accessibility Aids in IDEs

Many participants reported that accessibility in IDEs is poor and limited. Participants P16 and P53 indicated that the use of a text-editor is necessary since IDEs are very complex environments. The following are some responses from participants:

> *"Using a text editor is completely necessary because accessibility for IDE's is so poor." (Participant 16)*

> *"Accessibility issues in IDEs like visual studio." (Participant 53)*

While participant P5 indicated that certain parts of the IDEs are difficulties to use due to unstable screen reader:

> *"Stability issues with the IDE's and the screen readers. Certain parts of the IDE's being more difficult to use than my sighted counterparts have to deal with." (Participant 5)*

4.4.2 Code Navigation

Writing code requires moving or navigating through it in order to revise it or to track down mistakes. Movement through code using arrow keys is not enough due to the layout and the structure of code. Because of these difficulties, the need for an effective solution to overcome code navigation challenges is crucial. Several participants tried to overcome code navigation challenge with the use of scratchpad and editors. Here are some comments from participants:

> *"I have another window open to serve as a scratchpad (notes to fix things, method/variable names, etc). Having that separate scratchpad allows me to avoid losing my place in the code if I need to go look something up." (Participant 1)*

> *"Some text editors allows you to jump between the start and end of the block you are currently in." (Participant 44)*

Participant P36 uses a screen reader to listen to the code and Braille Display for more detailed information:

> *"I find that I listen to code with the screen reader audio, then if I want more detail, including punctuation, I use the Braille display. " (Participant 36)*

We believe detailed follow-up is needed to better understand how navigation occurs in different languages, environments, and with various skill levels.

4.4.3 Diagrams

Software developers need to be able to access various diagrams during the development process. Providing textual descriptions for diagrams in a timely manner is challenging. Several participants discussed the problem of accessing UML diagrams and the need for UML assistive tools. Some of their comments include:

> *"It isn't easy to diagram, I have to keep things in my head when I'm designing program flow." (Participant 1)*

Where other participant reported that diagrams does help show how certain things work before coding.

> *"It's not possible to look at class diagrams to have a quick idea of how some stuff you did not code works." (Participant 7)*

4.4.4 Debugging and User Interface Layout

Features in many IDEs include the support for debugging and also user interface layout. Respondents difficulties accomplishing both debugging and UI layout. Developers indicated the use of basic debugger utilities such as breakpoints, stepping through code, and print-f. Participants also said that debugging tools are difficult to use. A sample of comments includes:

"The challenges I face more often concern interacting with errors and warnings and consulting documentation or tooltips." (Participant 1)

"Debugging and interface design need visual development tools and they are not accessible and compatible with screen readers." (Participant 12)

4.4.5 Seeking Sighted Assistance

Many respondents indicated the need to seek out help from sighted developers for certain tasks. Several respondents feel embarrassed when working with other sighted teammates. For example, participants P10 and P49 rely heavily on the assistance of a sighted person to help them overcome some programming issues.

"Asking a sighted colleague for assistance." (Participant 10)

"Many times I use the assistance of a sighted person." (Participant 49)

4.4.6 Workaround Techniques

In the elicitation process, a focus on challenges can result in the omission of positive information (e.g. workarounds that are used to complete tasks). Many respondents presented a myriad of workarounds for diverse development tasks. For example, P7 found an alternative way to access UML digammas. However, they did not provide detailed information on the approach used. Other comments include:

"I have found alternative ways to access UML. A blind person to perform a software engineering job must know their access tech in side out." (Participant 7)

Participant P16 uses a text editor to overcome the complexity of IDEs.

"I have met the challenges by using a text editor to write code, attempting to run the code, and continuing to edit and revise until I achieve the result I want." (Participant 16)

5. LIMITATIONS

Although our study achieved its goals, we recognized that there are some unavoidable limitations. The study is limited by the snowball sampling technique, which resulted in uneven participant categories. The technique was used in order to maximize the number of responses in the time allotted from a population that is challenging to recruit.

6. CONCLUSION

In this work, we explored the challenges faced by developers who are blind. The goal was to understand the blind developers' programming problems as well as their workarounds to overcome these challenges. Some of the results were expected such as the lack of accessibility in IDEs as well as the use of a screen reader. We were surprised to see that blind developers use text editors as preferred tools to write code. It is not clear whether blind developers are unaware of the variety of features offered within the IDEs or find them difficult to use. For example, some participants noted that Eclipse or Visual Studio were not accessible while other respondents did use these tools.

We have discussed several implications, but further investigation is needed to determine what our conclusions can be generalized. For example, the survey analysis indicated the difficulty of code navigation where blind developers find it hard to navigate quickly through large amounts of code. The participants did not express sufficient details to such a problem and to generate requirements. A further study is needed to illuminate this particular subject with a well-defined user profile. We aim to conduct observational and interview studies with blind developers in a remote setup (using Skype or Google Hangouts). Thus, the geographical distribution can be overcome while providing a representational sample of computer science students as well as professional software developers for the needed subject.

Acknowledgment

We would like to thank our participants as well as our anonymous reviewers for their suggestions and feedback.

7. REFERENCES

[1] Eliciting programming challenges for developers with visual impairments survey. https://people.rit.edu/kla3145/Research/pdf/BlindProgrSurvey.pdf. Accessed: 2016-02-29.

[2] C. M. Baker, L. R. Milne, and R. E. Ladner. Structjumper: A tool to help blind programmers navigate and understand the structure of code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3043–3052. ACM, 2015.

[3] S. Mealin and E. Murphy-Hill. An exploratory study of blind software developers. In *Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on*, pages 71–74. IEEE, 2012.

[4] A. C. Smith, J. S. Cook, J. M. Francioni, A. Hossain, M. Anwar, and M. F. Rahman. Nonvisual tool for navigating hierarchical structures. In *ACM SIGACCESS Accessibility and Computing*, number 77-78, pages 133–139. ACM, 2004.

[5] A. Stefik, C. Hundhausen, and R. Patterson. An empirical investigation into the design of auditory cues to enhance computer program comprehension. *International Journal of Human-Computer Studies*, 69(12):820–838, 2011.

[6] A. M. Stefik, C. Hundhausen, and D. Smith. On the design of an educational infrastructure for the blind and visually impaired in computer science. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 571–576. ACM, 2011.

[7] P. Vickers and J. L. Alty. When bugs sing. *Interacting with Computers*, 14(6):793–819, 2002.