How do deaf or hearing impaired programmers perform in debugging java code?

Marcos Devaner do Nascimento[1], **Francisco Carlos de Mattos Brito Oliveira**[1], **Adriano Tavares de Freitas**[2]

[1]Computer Science Department – State University of Ceará (UECE)
Itaperi Campus, Fortaleza - CE - Brazil

[2]Computing Department – Federal Institute of Ceará (IFCE)
Maracanaú Campus, Maracanaú - CE - Brazil

marcos@projetolead.com.br, {fran.mb.oliveira, tfreitas.adriano}@gmail.com

Abstract. *People who are deaf or hearing impaired (DHI) often struggle with low-paying jobs. Access to education can change their perspectives. Many jobs in the information technology industry are left unfilled due to the lack of skillful candidates. Our lab offers distance learning Java programming courses to the DHI. We wondered how good our java DHI graduates are in finding and fixing program errors and changing program logic. We asked 5 DHI and 5 non-DHI programmers with the same amount of experience and training to perform debugging related tasks. We performed task and situated analysis and found that DHI programmer performance inferior to those without the disability. We argue that a debugging tool based on a mode more kin to the DHI should mitigate the disparity.*

Resumo. *Pessoas surdas ou com deficiência auditiva (SDA), muitas vezes possuem empregos de baixa remuneração. O acesso à educação pode mudar suas perspectivas. Muitos postos de trabalho na área de tecnologia da informação são deixados ociosos por falta de candidatos hábeis. Nosso laboratório oferece cursos à distância de programação Java para pessoas SDA. Nos perguntamos o quão bons nossos alunos SDA são em encontrar e corrigir erros em um programa. Cinco programadores SDA e cinco programadores ouvintes com a mesma experiência e treinamento executaram atividades de depuração de código. Realizamos uma análise situada e descobrimos que o desempenho dos programadores SDA era menor em relação àqueles sem deficiência. Argumentamos que uma ferramenta de depuração voltada para o público SDA deve atenuar essa disparidade.*

1. Introduction

According to the Brazilian Federal Census (2010), Brazil has 9.7 million people who are deaf or hearing impaired (DHI). DHI often struggle with low paying jobs. Brazilian Federal law 10.436/2002 requires that companies with more than 100 employees must have from 2 to 5% of their working force comprised of people with disabilities. Despite governmental incentives to promote educational and social inclusion, learning materials and teaching processes are still inneficient [Santiago 2011]. Thus, it is hard for the DHI to achieve competitiveness in the labor market. Jobs in technology present an opportunity to

improve the life of the DHI. There are many vacancies and the training time is relatively short. Moreover, distance education can rearch the DHI in remote areas or with mobility difficulties.

There are several learning environments for distance education of the deaf, however, just a few are meant for training DHI programmers. Our lab offers distance learning Java programming courses. The Java track is comprised of three modules: 1) Basic; 2) Intermediary and 3) Advanced and it takes twelve months to be completed. We have noticed that the DHI struggle to learn programming concepts and to get comfortable with integrated development environments (IDE), like Eclipse. For that matter, we have already proposed a highly cooperative and interactive IDE, strongly coupled with our learning platform. Such IDE, JLoad, embeds programming workshops into the learning platform itself and allows tutors to support pupils in every step, granting them access to the marked-up code, in a situated way. We describe JLoad in [Silva et al. 2014]. Although JLoad facilitates the learning of Java programming, it does not have support for debugging activities.

In the present work, we assess DHI performance in debug-related tasks and compare it with non-DHI programmers with the same training — all graduates from our Java courses. Debugging helps programmers to gain insight of program behavior, and is crucial for software maintenance and evolution. Labor market expects a computer programmer to be able to understand someone else's code and change its behavior. However, debugging sessions might be challenging. For [Le Goues et al. 2011], current debuggers are difficult to use, requiring skill and patience to understand the errors and failures presented. In this first study, the tasks are simple and so are the programs — none of them have even an object. All programs are procedural and well-structured. Debugging object-oriented programs are even harder [Lessa et al. 2010] and thus were left out of this initial study. We hypothesize that there is a difference in performance for the DHI and the non-DHI groups when related to debugging activities. We performed a study with five (5) DHI and five (5) non-DHI, aged between 23 and 38 involved in debugging activities to verify our hypotheses.

This paper is structured as follows. In the Section 2, we briefly discuss the challenges of teaching Java to the DHI who use a highly spacial sign language and are called to express themselves in a linear English-derived programming language. In the sequence, we elaborate on the importance of debugging activities for learning and on making the DHI programmers effective members of real world IT teams. After that, in Section 3, we detail our study including: participants' profile, tasks performed and quantitative/qualitative evaluation. Our analysis, showed in Section 4, indicates that there is a strong evidence of correlation between the hearing condition and the ability of finishing Java debugging tasks successfully. The time spent in performing the activities also differs in each group. In Section 5 we argue that using visual-spatial representations is a good way to improve the performance of the deaf in different activities and we show some works which use this visual appeal to make the programming activities easier. Finally in Section 6, as a response to our findings, we propose the assessment of existing Eclipse Visual Debuggers plugins in a set of tasks similar to the ones described here. We also discuss the risks involving the development and validation of highly visual tool can pose on performance of the DHI especially overloading their super-necessary vision and their

short term memory. Such risks can undo any work aimed to improve DHI performance through the deployment of a Visual Debugger.

2. Teaching the Deaf

DHI students are regarded as having lower academic performance in when compared to their hearing counterparts in all educational settings, as we can see in [Gregory 1998], [Traxler 2000] and [Nogueira and Zanquetta 2009]. [Blatto-Vallee et al. 2007] even pose that the DHI undergraduate are at the same mathematical level as the lowest scoring hearing mid schoolers. However hearing loss might not be at the root of the problem. [Nunes and Moreno 1998] argue that the "poor mathematical performance but rather more of a risk factor related to the timing, type of instruction, and learning opportunities provided to deaf student." To corroborate that line of thought, [Zarfaty et al. 2004] showed that 3 and 4-year-old DHI children have spatial and temporal skills comparable to their hearing colleagues at the same age and even better spatial numerical skills. [Barbosa 2013] argues that in cognitive functions less dependent on linguistic stimuli DHI and non-DHI children seem to have similar performance.

[Boroditsky 2011] argues that bilinguals reason differently when on the same matter when required to do so in the two languages. That *language shift* also impacts memory. Finally, the author posits that "language shapes even the most fundamental dimensions of human experience: space, time, causality and relationships to others."

It is relatively straightforward to understand why teaching materials designed for non-DHI are not quite appropriate for the DHI. For [Perlin 2004], the "hearing culture" is essentially comprised of auditory signs and that DHI does not use hearing signs as they cannot fully understand them as they comprehend them in the realm of visual signs. It is not only the lack of appropriate learning material. The Brazilian sign language (Libras) is relatively young (13 years) and has a poor lexicon, when compared to Portuguese. Concepts used in Computer Science in general and in object-oriented programming languages like Java (eg. polymorphism, instantiation) simply do not exist in Libras. Our lab is working on another research aimed to help the DHI community to enrich Libras lexicon.

Any programming course must include debugging lessons in a hands-on approach in its syllabus as employers expect their DHI programmers to be fully integrated with the rest of the workforce, mostly non-DHI. We wondered how our DHI students would perform in debugging related tasks when compared to non-DHI using an industry-standard programming IDE like Eclipse. To our knowledge, no such study has been done. In the following section we describe our experiment.

3. Study

Ten students who had just completed our 150 hour basic java course participated in our study. They were divided into 2 groups each with five participants: the DHI group and the non-DHI group. Participants' age ranged from 23 through 38 and were all male. Four DHI had college degrees and one undergrad. Among the non-DHI, there were one graduate, two undergrads and two high-school degree holders.

In Table 1, we summarise relevant demographic information about the participants. As one can note, DHI participants are slightly better educated but experience with

Table 1. Relevant Skills

Profile	Skill	Results
DHI	Portuguese Comprehension	4 participants - Good 1 participant - Moderate
	English Comprehension	4 participants - Moderate 1 participant - Little
	Development experience with Eclipse	5 participants - More than a year
	LIBRAS Proficiency	4 participants - Excellent 1 participant - Good
Non-DHI	English Comprehension	4 participants - Little 1 participant - Good
	Development experience with Eclipse	5 participants - More than a year

Eclipse is similar across groups.

3.1. The Procedure

The study consists in finding and correcting one error in two Java classes as illustrated in Figure 1. All the participants (non-DHI and DHI) used the Eclipse Debugger to perform some study tasks. The main idea of the experiment is to compare the way the two groups performed each task. Basically we measure the time spent to complete the task, the number of requests for assistance and if they successfully complete the task.

The seven tasks given to the participants in the experiment are:

- Task 01 - Modify to the debug perspective;
- Task 02 - Correct the errors in the code;
- Task 03 - Add breakpoint on a line of the code;
- Task 04 - Check the values of the variables;
- Task 05 - Use the stepover functionality;
- Task 06 - Loop through all the lines of the code;
- Task 07 - Modify to the java perspective;

They are typical activities in a process of debugging. There was no time constraint for the participants to finish. None of the tasks was mandatory. A video providing guidance was available in LIBRAS and Portuguese. A LIBRAS interpreter and an expert were available to answer technical questions and give support.

The experiment was recorded and later analyzed to check variables of measurement.

4. Results

Table 2 displays the data collected. Despite having attended to the same Java course, Non-DHI and DHI had significative differences in performance.

There is evidence of correlation between the hearing condition and the ability to complete tasks related to Java debugging in the Eclipse IDE — $\chi^2(1, N = 70) = 17.43$, $p < 0.001$. There is also evidence that DHI take long to complete (when they do

```
1
2  public class Account {
3
4      public double balance = 400;
5
6      public double draw (double value){
7
8          if (this.balance < value){
9              this.balance = Double.parseDouble(value);
10             System.out.println("There aren't sufficient funds.");
11         }else{
12             this.balance = Double.parseDouble(value);
13             this.balance = value;
14             System.out.println("Successfully performed.");
15         }
16         return value;
17     }
18
19 }
```

```
1
2  public class AccountTest {
3
4      public static void main (String[] args){
5          Account c = new Account();
6          c.draw(100);
7      }
8
9  }
10
```

Figure 1. Java classes with errors

complete) the tasks — $t(44) = 2.54$, $p = 0.0153$. We ran the *Student's t-test* at 90% confidence interval to mitigate the risk of committing a type II error, due to low number of participants.

Table 2. Collected Data

	Mean Time to complete (mm:ss)		# requests for help		completed the task	
	Non-DHI	DHI	Non-DHI	DHI	Non-DHI	DHI
Task 01	00:31	03:59	0	2	5	5
Task 02	19:38	17:37	3	1	4	1
Task 03	02:43	15:59	1	1	5	1
Task 04	03:39	-	2	0	3	0
Task 05	00:50	22:45	0	3	2	4
Task 06	03:15	-	2	0	5	0
Task 07	01:29	06:52	2	4	4	5

Interestingly, we could not find any relevant difference in the number of requests for help, despite the fact that there was a translator and an expert available. In our situated analysis, we found that in many occasions the DHI get completely lost, they seem not to understand what is asked of them, despite the fact that the instructions are available in LIBRAS. It seems that they do not understand the context. They particular get lost when they have to read the messages conveyed by the IDE. As for the Non-DHI, they easily find the appropriate widgets on IDE and have a better understanding of the system messages. They also explore the IDE with more resourcefulness.

5. Towards a Visual Accessible Java Debugger

[Blatto-Vallee et al. 2007] point that the use of visual-spatial schematic representations is a strong positive predictor of mathematical problem-solving performance for the deaf stu-

dents. For [Pinto et al. 2014], visuality seems to represent, to the deaf, the main channel for thinking and processing schemes that naturally enable the acquisition, construction and expression of knowledge, values and experiences that otherwise would be incommunicable. The visual channel allows the reading of the deaf world and is the support of their mental processing. The authors report improvement on self-esteem, interest and engagement among the deaf when drawings, images and visual manipulatives were used in the teaching of science, geography, arts and history.

The discussion above suggests that the visual programming techniques should be investigated. This is not quite new research area since, visual programming dates back at least to the seventies. *Pygmalion* [Smith 1975] allowed programmers to visualize programs' arithmetic operators. Although there are several visual Java debuggers, to our knowledge, none of them had been tested with the deaf. Before we proceed with the discussion of the available visual java debuggers, we must caution that it seems that sign language users are capable of holding less information in their short term memory (4 ± 1 items) when compared to non-DHI (7 ± 2) [Bavelier et al. 2006]. This might be partially due to the visual-spatial nature of the information required for the DHI [Wilson and Emmorey 2006]. Therefore, one must to aware of not overloading the visual mode when designing a java visual debugger for the deaf, or adapting an existing one. The other relevant aspect of the tool we seek to deliver to the deaf programmer is that is has to be free, portable and flexible enough to be used different workplace configurations since we want our DHI graduates to take them to their future jobs for productivity proposes.

In a recent survey, [Sorva et al. 2013] point that "in the last three decades, dozens of software systems have been developed whose purpose is to illustrate the runtime behaviour of computer programs to beginner programmers." We refer the reader to that publication for more detailed text on these systems. We cite just a few of them as we debate the desirable characteristics of the DHI visual java debugger we look for.

Jeilot 3 [Moreno et al. 2004] was created to help Java learners on basic concepts of procedural or object-oriented programming. Its main feature is the visualisation either total or partial of code or flow control. With Jeilot 3, pupils can, at the same time, code and examine the visual representation of that code in execution time. During that process, students acquire mental models that will help them in the construction and understanding of Java programs. Jeilot 3 creators aimed to provide a tool to lower cognitive demands of the Java learner at their first contacts with the technology.

[Alsallakh et al. 2012] proposed a plugin that allows programmers to change the conventional Eclipse breakpoints to tracepoints and to gather information at runtime. With this tool debugging is done by means of diagrams using an interactive interface where the programmer can analyse the point-line pre-established code, plus the amounts and types of data. Compared with existing Eclipse plugins for visual debugging, this plugin provides a simple process to set and display the data to be tracked. Informal assessment showed that the stories and metaphor instance of line graphs are easy to understand. With that programmers will have scenarios where the visual tracking will help them in understanding and debugging of their programs.

jGRASP [Cross II et al. 2007] is a lightweight IDE developed to provide auto-

matic dynamic visualisations of Java data structures. Such visualisations are synchronised with the source-code allowing its user to walkthrough the code when the tool is set to debug mode. Such integration provides an unique and promising environment for the learning of data structures in Java. The authors claim that its use has a significant and positive impact on Java students. The visualisation techniques and animations used in debugging sections might have a positive impact on the DHI. The approach, however, is restricted to the teaching of Java data structures and not a general propose java debugger.

JIVE [Lessa et al. 2010] uses UML diagrams for status display and call methods within the contexts of the objects. This tool aims to provide the developer a clearer and more dynamic view of debugging code. JIVE allows the student to focus on specific regions of the diagrams, while the diagrams show responses to queries. JIVE is designed around two basic principles: present views of the state and execution history object, and support for declarative queries to search through recorded executions. It can be incorporated through the eclipse plugin, so that the student can use an IDE with wide acceptance in the developer community and debug his/her code more simple and didactic way.

As one can see, visual debuggers have extensively been used to help the teaching of programming languages. The two of our main research objectives remain: 1) to assess the impact of visual debuggers on the teaching of DHI and; 2) to provide a debugging tool that can be used both for training proposes and in the workplace.

6. Conclusions and Future Work

The IT industry presents a good opportunity for the DHI to access better paying jobs. However, for that to become a reality, the DHI performance might match those who does not have the impairment. Our laboratory offers distance learning Java programming courses for the DHI. Distance learning is an interesting option since the DHI are geographically disperse. We wondered how our DHI graduates perform when compared to the non-DHI who took the same 150 hour Java programming course in tasks related to debugging. We used the industry-standart Eclipse IDE for debugging. We argue that we should test the performance in such setting as it is the one that the DHI graduate will most probably find in real world industry. Knowing how to debug is important to the understanding of internal structures and behaviour of a computer program. Moreover, as a future IT professional, DHI programmers must be able to evolve and change a code written by a non-DHI coworker.

We noticed that there is in fact a difference in performance between the DHI and non-DHI groups. The DHI had poorer performance and some were not even able to finish the tasks, despite the fact that no there was no time constraint. This is an important finding as it hinders the DHI chances of getting employed. This difference, as addressed in Section 2, is due to language issues, since the DHI think differently about debugging as their mechanisms of thought are based in a visual-spatial language. As a response, we turned to discuss the use visual debuggers, the risks of overloading the DHI's super-important sight as they are capable of holding less information in their short term memory than non-DHI. Despite the risks, visual debuggers seem to be worth further investigating. We then discussed several visual debuggers and praised the fact the some of them are implemented as Eclipse plugins, as this feature might facilitate its adoption by the industry, or at least, among those companies willing to hire DHI programmers.

We now move on to assess several Eclipse Visual Debuggers plugins and evaluate which one(s) can improve DHI debugging performance. We are currently defining plugin inclusion criteria, but they all have to be open source projects as it is highly probable that DHI Visual Debugger plugin will emerge from this research avenue.

References

Alsallakh, B., Bodesinsky, P., Gruber, A., and Miksch, S. (2012). Visual tracing for the eclipse java debugger. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 545–548. IEEE.

Barbosa, H. H. (2013). Habilidades matemáticas iniciais em crianças surdas e ouvintes. *Cad. Cedes*, 33(91):333–347.

Bavelier, D., Newport, E. L., Hall, M. L., Supalla, T., and Boutla, M. (2006). Persistent difference in short-term memory span between sign and speech implications for cross-linguistic comparisons. *Psychological Science*, 17(12):1090–1092.

Blatto-Vallee, G., Kelly, R. R., Gaustad, M. G., Porter, J., and Fonzi, J. (2007). Visual–spatial representation in mathematical problem solving by deaf and hearing students. *Journal of Deaf Studies and Deaf Education*, 12(4):432–448.

Boroditsky, L. (2011). How language shapes thought. *Scientific American*, 304(2):62–65.

Cross II, J. H., Hendrix, T. D., Jain, J., and Barowski, L. A. (2007). Dynamic object viewers for data structures. *ACM SIGCSE Bulletin*, 39(1):4–8.

Gregory, S. (1998). Mathematics and deaf children. *Issues in deaf education*, pages 119–126.

Le Goues, C., Leino, K. R. M., and Moskal, M. (2011). The boogie verification debugger (tool paper). In *Software Engineering and Formal Methods*, pages 407–414. Springer.

Lessa, D., Czyz, J. K., and Jayaraman, B. (2010). Jive: A pedagogic tool for visualizing the execution of java programs. Technical report, Technical Report 2010-13). Retrieved May 15, 2010, from http://www. cse. buffalo. edu/tech-reports/2010-13. pdf.

Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004). Visualizing programs with jeliot 3. In *Proceedings of the working conference on Advanced visual interfaces*, pages 373–376. ACM.

Nogueira, C. M. I. and Zanquetta, M. E. M. (2009). Surdez, bilingüismo eo ensino tradicional de matemática: uma avaliação piagetiana. - deafness, bilingualism and traditional teaching of mathematics. *Zetetiké: Revista de Educação Matemática*, 16(30):219–237.

Nunes, T. and Moreno, C. (1998). Is hearing impairment a cause of difficulties in learning mathematics. *The development of mathematical skills*, pages 227–254.

Perlin, G. (2004). O lugar da cultura surda. *A invenção da surdez: cultura, alteridade, identidade e diferença no campo da educação*, pages 73–82.

Pinto, M. A. d. S., Gomes, A. M. d. S., and Nicot, Y. E. (2014). A experiência visual como elemento facilitador na educação em ciências para alunos surdos. *Revista Areté: Revista Amazônica de Ensino de Ciências*, 5(09).

Santiago, V. d. A. A. (2011). A participação de surdos no mercado de trabalho. In *Anhanguera Educacional*, pages 1–16.

Silva, L. C., Oliveira, F. C. M. B., Freitas, A. T., and Oliveira., A. C. (2014). Introducing the jload, a java learning object to assist the deaf. In *Proceedings of the 14th IEEE International Conference on Advanced Learning Technologies*, page to appear.

Smith, D. C. (1975). Pygmalion: a creative programming environment. Technical report, DTIC Document.

Sorva, J., Karavirta, V., and Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *Trans. Comput. Educ.*, 13(4):15:1–15:64.

Traxler, C. B. (2000). The stanford achievement test: National norming and performance standards for deaf and hard-of-hearing students. *Journal of deaf studies and deaf education*, 5(4):337–348.

Wilson, M. and Emmorey, K. (2006). Comparing sign language and speech reveals a universal limit on short-term memory capacity. *Psychological Science*, 17(8):682–683.

Zarfaty, Y., Nunes, T., and Bryant, P. (2004). The performance of young deaf children in spatial and temporal number tasks. *Journal of Deaf Studies and Deaf Education*, 9(3):315–326.