A Randomized Controlled Trial on the Impact of Polyglot Programming in a Database Context

Phillip Merlin Uesbeck
University of Nevada, Las Vegas
Las Vegas, NV, USA
uesbeck@unlv.nevada.edu
https://orcid.org/0000-0001-8182-9942

Andreas Stefik
University of Nevada, Las Vegas
Las Vegas, NV, USA
stefika@gmail.com

Abstract

Using more than one programming language in the same project is common practice. Often, additional languages might be introduced to projects to solve specific issues. While the practice is common, it is unclear whether it has an impact on developer productivity. In this paper, we present a pilot study investigating what happens when programmers switch between programming languages. The experiment is a repeated measures double-blind randomized controlled trial with 3 groups with various kinds of code switching in a database context. Results provide a rigorous testing methodology that can be replicated by us or others and a theoretical backing for why these effects might exist from the linguistics literature.

2012 ACM Subject Classification Software and its engineering → Software development techniques

Keywords and phrases human-factors, randomized controlled trial, polyglot programming

Digital Object Identifier 10.4230/OASIcs.PLATEAU.2018.1

Funding This work was partially funded by the NSF under grants 1644491, 1738259, and 1640131.

1 Introduction

Polyglot programming, using more than one computer language in the same programming context, is common. Evidence from Tomassetti and Torchiano suggests that 97% of open source projects on Github use two or more computer languages [19], with the number of languages per project averaging about 5 [11, 19]. Seeing how common the use of multiple programming languages is, the question of its productivity impact on developers at various levels becomes salient. An obvious research question might be, "Does polyglot programming have an impact on developer productivity, and if so, how large is it, what direction, and in what context?" Given that software is a $407.3 billion industry [1] and that the median salary for a software developer is $103,560 per year in the U.S. [12], productivity impacts are expensive at scale. This broad question, which we investigate one specific aspect of in this paper, guides our broader research direction.

One aspect of polyglot programming is repeated switching between languages, which we will call code switching. Using the study presented in this paper, we aim to evaluate whether code switching impacts developer productivity. Our running theory as to why this might be is guided by evidence-based research in the field of linguistics, which investigates

the phenomenon of code switching in natural language [8, 10, 21]. Research in that context has shown that there is a time cost to switching between the use of natural languages [13], which begs the question of whether a time cost is measurable for switching between computer languages. Given that neurological studies on program comprehension and natural language comprehension seem to use the same areas of the brain [14], such a hypothesis seems plausible.

The specific aspect of the broader research line we are investigating is in the common case of polyglot within a database programming context. Thus, we present here a small pilot study of a double-blind randomized controlled trial with six programming tasks. To do this, we created three different versions of a querying API, 1) one without polyglot (non), 2) one where raw SQL is embedded (polyglot), and 3) a hybrid approach of our own design where the embedded query is similar to the host language (hybrid). Our null-hypotheses is that there is no relationship between code switching and productivity in the context of database programming.

With guidance from this pilot study, our goal is to build up a research line in a systematic way, increasing the sample size as we go and replicate the work rigorously each time. Finally, many of the experiments our lab conducts are designed off of the work of Austin Bradford Hill [6], who led pioneering work in medicine on how to design experiments and which later helped lead to the CONSORT evidence standard. To our knowledge, the field of computer science does not have an evidence standard for empirical work, so we adapted CONSORT after discussions from Schloss Dagstuhl [15]. Put another way, while scholars sometimes disagree about what should be reported with empirical work, or how it is reported, we followed an existing evidence standard to provide an evidence-based structure on what we should and should not report.

The rest of this paper is going to be structured as follows: First, we will discuss related work. Then, we will go over the design of the experiment in section 3. Results will be shown in subsection 4 and then discussed in section 5, finally the paper will end in a conclusion in section 6.

2 Related Work

The benefits and drawbacks of polyglot programming on a human-factors level are under-explored in the scientific literature. The main argument in favor of polyglot programming is about *"[...] choosing the right tool for the job"* [2], a view that seems to drive the field of domain specific language research which proposes using specialized languages for better productivity and maintenance [20]. Claims have been made that the use of a more appropriate language for a task leads to better productivity and easier maintenance by reducing the lines of code of a project [7], but also that the need to learn more languages creates a strain on the developers and that introduction of more languages to a project can reduce the pool of developers able to maintain the project [7].

Programmer productivity is studied in a variety of aspects of programming. Studies range from programming language features such as syntax [16] and type systems [9], over API design [17] and the effect of errors [4] to studies trying to investigate the cognitive processes involved [14, 5]. These studies on APIs, syntax, or others provided guidance to this work, although there is little in the literature that contains measurements of polyglot programming itself.

Listing 1 Task 1 as presented to the participants

```
package library;

import library.*;

public class Task1 {

  /**
   * Please write this method to return a
       ↪ Table object containing
   * all columns for all entries with an id
       ↪ smaller than 32
   * and sorted from high salary to low salary
       ↪ .
   *
   * Table information:
   *
   * - prof -
   *
   * id (int) | firstname (String) | lastname
       ↪ (String) | salary (int)
   *
   *
   *
   * Use the technique shown to you in the
       ↪ samples given
   *
   */
  public Table query(Table prof) throws
       ↪ Exception {

    // Your Code here

    return null;
  }

}
```

Listing 2 Task 1 Solution polyglot

```
Query q = new Query();

q.Prepare("SELECT * FROM professors" +
" WHERE id < 32 ORDER BY salary DESC");

Table r = prof.Search(q);

return r;
```

Listing 3 Task 1 Solution non

```
Query q = new Query();

q.SortHighToLow("salary");
q.Filter(q.Where("id").LessThan(32));

Table r = prof.Search(q);

return r;
```

Listing 4 Task 1 Solution hybrid

```
Query q = new Query();

q.SortHighToLow("salary");
q.Filter("id < 32");

Table r = prof.Search(q);

return r;
```

3 Methods

Study Design. We conducted a double-blind repeated measures randomized controlled trial in which participants were randomly assigned to one of three experimental groups between April and May 2018. Each participant was asked to solve 6 programming tasks.

While we chose this particular design for our study, where participants used a host language with an embedded code, there are several other ways that it could have been designed and we are looking to expand testing to such designs at a later time. Under consideration was to have participants study one language in a task, then another in a second. We also considered testing smaller code samples to isolate certain aspects of switching, like just the one method where embedding occurs for polyglot. Ultimately we settled on a design that we thought was balanced. Since embedding SQL into a host language is common, it has some face validity as a common polyglot task and thus seemed like a reasonable place to investigate first.

Study Population & Setting. Eligible were participants over the age of 18 who had some programming experience. Recruitment occurred during class time via advertisement pamphlet at the University of Nevada, Las Vegas. The pamphlet contained the URL to the website used for the experiment which guided participants through the entire process of the experiment. On the website, participants give consent, fill out a survey, and then solve the programming tasks. A time limit of 45 minutes per task was given to limit the overall time commitment for the experiment.

Intervention. Participants were randomly assigned to one of three groups and received a different code sample depending on their group. Each group's code sample demonstrated enough code to let participants infer the solutions needed for the tasks. We designed the three different groups to require different amounts of language switching while writing database

queries in Java. The first group (polyglot) uses SQL strings as parameters in method calls to create a query (see listing 2 and 5), requiring switches. The second group (non-polyglot) had to create a query by building it from objects in a series of method calls in a more object-oriented approach (see listing 3 and 6), requiring no switches, and lastly, the hybrid group's approach mixes the use of parts of SQL-style strings and method calls, using SQL syntax for conditions and column names and method calls for the rest (see listing 4 and 7).

To solve the tasks, participants had to fill in the code provided to them in a way that satisfied all unit tests. If an incomplete solution was submitted, compilation output was displayed and work on the tasks continued until a successful solution was submitted or the time ran out. The participants had to solve 3 $SELECT$, 1 $UPDATE$, 1 $INSERT$, and 1 $JOIN$ task to cover a range of different uses of queries. While the $UPDATE$, $INSERT$, and $JOIN$ tasks were kept standard, the 3 $SELECT$ tasks varied in the complexity of their conditions and whether a $ORDER\ BY$ component was needed.

An example of what the tasks looked like can be seen in listing 1, which shows the empty first task of all three groups. The instructional comments describe the structure of the table object passed into the method. Possible solutions for each group to the first task can be seen in listings 2, 3, and 4.

Outcomes. The difference of the start time and the end time of a task decided the dependent variable time to solution. As a random factor, the platform also recorded the participants' self-reported experience in using databases.

Randomization. Randomized assignment to the groups was handled by the website and followed the covariate adaptive randomization approach [18]. The participants were assigned to a experience category based on their college year. Group assignment was then conducted randomly but balanced within experience categories.

Blinding. The assignment of participants to their group was done automatically and without intervention by the researchers. Since the experiment was done using the website, there was no direct interaction with the participants and therefore the proctors had fewer avenues to accidentally or intentionally bias them. The participants were not informed about which group they were assigned to or what the hypothesis of the study might be.

4 Results

Recruitment. We recruited 11 participants for this pilot study. Of the 11 participants, 5 identified themselves as female and 6 as male. On average, the participants were about 24 years old (M = 23.55, SD = 7.10). Six of the participants were sophomores, two were juniors, two were seniors, and one was a graduate student. Four of the participants reported that English is not their primary language. The polyglot group had 4 participants, the non-polyglot group had 3, and the hybrid group had 4. The demographics can also be found in table 1.

Baseline Data. All 11 of the participants completed all tasks. The data (all times in seconds) can be found in table 2. Figure 1 shows the average task completion times between the two groups. Figure 2 shows the task times by group in a boxplot.

Table 1 Demographics.

Metric	Polyglot	Non	Hybrid
N	4	3	4
DB Experience	25.00%	33.33%	25.00%
Female	50.00%	66.66%	0.00%
Age	23.50 (SD = 5.74)	20.33 (SD = 1.53)	26.00 (SD = 10.74)
Native	75.00%	66.66%	100.00%

Table 2 Times per task in seconds.

	Non			Polyglot			Hybrid			Total	
Task	N	mean	SD	N	mean	SD	N	mean	SD	mean	SD
Task 1	3	1371.67	545.68	4	1380.50	700.38	4	999.25	668.48	1239.45	614.05
Task 2	3	676.33	143.39	4	782.00	641.35	4	848.50	368.77	777.36	416.42
Task 3	3	1006.67	533.35	4	1870.25	419.55	4	1363.50	965.22	1450.45	722.35
Task 4	3	438.67	220.18	4	268.00	134.39	4	357.50	215.69	347.09	184.75
Task 5	3	154.33	39.95	4	702.50	945.13	4	258.75	107.21	391.64	578.24
Task 6	3	663.00	412.94	4	787.00	437.08	4	337.25	47.33	589.64	367.10
Total	18	718.44	507.60	24	965.04	751.49	24	694.12	615.21	799.27	645.89

Analysis. To analyze the results, we ran a mixed designs repeated measures ANOVA using the R programming language with respect to time to solution, using task as a within-subjects variable and group and database experience as between-subjects variables. Sphericity was tested using Mauchly's test for Sphericity, which shows that the assumption of sphericity was violated for the variable task. Following reported numbers are reported with Greenhouse-Geisser corrections taken into account.

There is a significant effect at $p < 0.5$ for the within-subjects variable task, $F(5, 25) = 7.065$, $p < 0.001$ ($\eta_p^2 = 0.479$), but no significant effect for group, $F(2, 5) = 0.889$, $p = 0.467$ ($\eta_p^2 = 0.110$) or database experience, $F(1, 5) = 0.973$, $p = 0.369$ ($\eta_p^2 = 0.064$). None of the interaction effects are significant. To test the differences between the tasks in more detail, we ran a Bonferroni test. There are significant differences between task 1 and 4 (p = 0.011), 1 and 5 (p = 0.041), 3 and 4 (p = 0.009), 3 and 5 (p = 0.008), and 3 and 6 (p = 0.035).

Listing 5 Solution 6 polyglot

```
Query q = new Query();
q.Prepare("SELECT id,
    ↪ firstname, lastname,
    ↪ clubname "+
  "FROM students JOIN clubmap
      ↪ ON students.id =
      ↪ clubmap.studentid");
Table r = students.Search(q,
    ↪ clubmap);
return r;
```

Listing 6 Solution 6 non

```
Query q = new Query();
q.AddField("id")
  .AddField("firstname")
  .AddField("lastname")
  .AddField("clubname");

q.Combine(students, "id",
    ↪ clubmap, "studentid");
Table r = students.Search(q);
return r;
```

Listing 7 Solution 6 hybrid

```
Query q = new Query();
q.AddFields("id, firstname,
    ↪ lastname, clubname");

q.Combine(students, "id",
    ↪ clubmap, "studentid");

Table r = students.Search(q);

return r;
```

5 Discussion

It is crucial to state that we found no significant differences overall in our pilot. This is what we would expect with a sample size of 11 unless polyglot had a very large effect size. We observed that $\eta_p^2 = 0.110$ for the group task, which means the polyglot effect explained about 11% of the variance. This needs to be confirmed or refuted at scale.

Besides the effect, which gives us clues toward what we might see in replications, we think observations about the individual tasks are interesting. Note that the non-polyglot group performed 3.50% slower than hybrid group across all tasks and that the polyglot group was

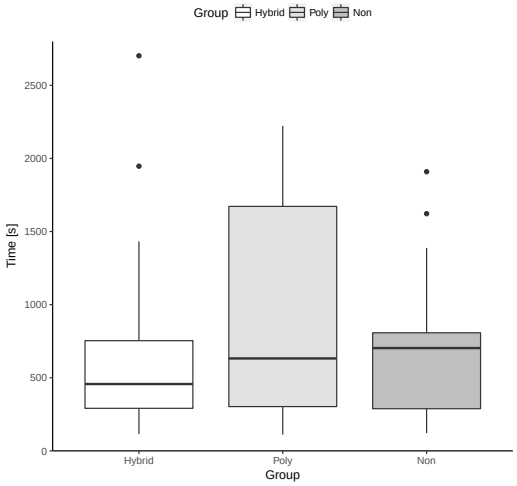

Figure 1 Boxplot of results between the groups.

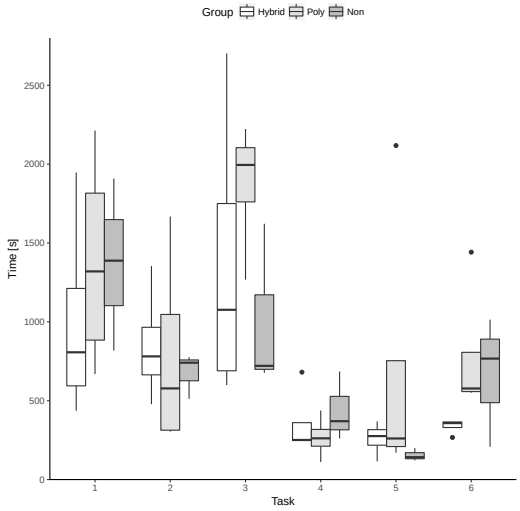

Figure 2 Boxplot of differences in time by task.

39.0% slower than hybrid. For some tasks, the effects were more pronounced. Consider task 6, where the non-polyglot group performed 96.6% slower than the hybrid group and the fully polyglot group was 133.4% slower than hybrid. This task is shown in listings 5, 6, and 7. We highlight this task not because of its result, but because the differences in the code are quite small. Given the large differences in human performance, this surprised us.

We think these observations teach us two lessons. First, polyglot programming is not a simple concept. Claims made in the literature about the practice being good or bad, without corresponding data, should be re-considered empirically. For example, after reading the linguistics literature on code switching, while we hypothesized the non-polyglot group might do very well, our polyglot hybrid seemed to have about the same impact. The study presented in this article is only a first step towards investigating the issue and we hope that the experimental design we presented can be a building block in future study of the impact of polyglot programming. Secondly, the polyglot group did quite poorly, despite the fact that we tested a commonly known case of polyglot programming. This might indicate, if our results are confirmed at scale and under more conditions, that a conditional polyglotism might be reasonable in programming languages. As the polyglot group uses two distinct languages and the hybrid group mixes languages while the last group stays within the same language, it appears that the distance from the host language could have an impact on results. This might suggest that switching between semantically and syntactically similar languages might be easier than switching between different ones.

Differences between tasks overall are explained by the differences in difficulty of the tasks and a learning effect. Task 3, presents a spike in times as it was the conceptually hardest task. It required participants to create a complex logical expression, compared to the ones for task 1 and 2, while most other tasks are what could be considered standard versions of their respective database commands.

Limitations. Every empirical study has limitations and ours is no exception. The most obvious to our pilot study is that it is a low sample size experiment on student programmers. We certainly make no claims about generalization, as this is not the point of a pilot study.

That said, it is important to discuss limitations not just in this sense, because generalization can mean many things, but also what limits must be overcome to get a generalized grasp of polyglot in practice.

First, we think to really understand polyglot in the wild, we would need a combination of measures to sort out the effects. For example, non-native English speakers writing code in English might have different results that need to be considered. On this same line of thinking, children learning to program, with various levels of understanding of their native tongue, would likely have different impacts as well. Similarly, professional programmers, or perhaps more specifically experts in polyglot programming in databases, might have different effects still. From our perspective, we think it is important to test all of these different kinds of people to sort out the facts over time.

Second, while we used a randomized controlled trial in a lab setting because our hybrid approach does not exist in the field, it is important to recognize that effects from a lab setting may not match those in practice, although in programming languages they sometimes do. Just as one example, studies on syntax [16] seem to provide similar results to those on compiler errors in the field [3], which Bradford-Hill would call "coherence." That said, there is no panacea in empirical work. Rigorous data gathering over time, through multiple techniques, is often what settles difficult research questions.

6 Conclusion

In this paper, we described a pilot experiment on the impact of code switching on software development productivity, which was motivated by the prevalence of the practice in the field. Findings in linguistic research suggest that there is a time cost to switching between natural languages, but to our knowledge this is the first randomized controlled trial on the topic for programming languages. We conducted a pilot study with three groups, exploring alternative designs for database programming, including polyglot, non-polyglot, and a hybrid approach. Our study was a small pilot designed to evaluate our methodology, so the results are not conclusive. That said, they do appear to provide a hint that the syntactic and semantic distance between embedded languages, amongst other factors, could impact human productivity in practice.

References

1 Gartner Says Worldwide Software Market Grew 4.8 Percent in 2013. `https://www.gartner.com/newsroom/id/2696317`. Accessed: 2018-06-02.

2 Polyglot Programming. `http://nealford.com/memeagora/2006/12/05/Polyglot_Programming.html`. Accessed: 2018-04-25.

3 Amjad Altadmri and Neil C.C. Brown. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 522–527, New York, NY, USA, 2015. ACM. `doi:10.1145/2676723.2677258`.

4 Brett A Becker. A new metric to quantify repeated compiler errors for novice programmers. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 296–301. ACM, 2016.

5 Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. Eye movements in code reading: Relaxing the linear order. In *Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on*, pages 255–265. IEEE, 2015.

6 Richard Doll and A Bradford Hill. Smoking and carcinoma of the lung. *British medical journal*, 2(4682):739, 1950.

7 Hans-Christian Fjeldberg. *Polyglot programming. a business perspective.* PhD thesis, Master thesis, Norwegian University of Science and Technology, 2008.

8 Roberto R Heredia and Jeanette Altarriba. Bilingual language mixing: Why do bilinguals code-switch? *Current Directions in Psychological Science*, 10(5):164–168, 2001.

9 Michael Hoppe and Stefan Hanenberg. Do developers benefit from generic types?: an empirical comparison of generic and raw types in java. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2013, part of SPLASH 2013, Indianapolis, IN, USA, October 26-31, 2013*, pages 457–474. ACM, 2013. doi:10.1145/2509136.2509528.

10 Ping Li. Spoken word recognition of code-switched words by Chinese–English bilinguals. *Journal of memory and language*, 35(6):757–774, 1996.

11 Philip Mayer and Alexander Bauer. An empirical analysis of the utilization of multiple programming languages in open source projects. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, page 4. ACM, 2015.

12 Bureau of Labor Statistics. Software Developers - Summary. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm. Accessed: 2018-04-20.

13 Daniel J Olson. Bilingual language switching costs in auditory comprehension. *Language, Cognition and Neuroscience*, 32(4):494–513, 2017.

14 Janet Siegmund, Norman Peitek, Chris Parnin, Sven Apel, Johannes Hofmeister, Christian Kästner, Andrew Begel, Anja Bethmann, and André Brechmann. Measuring neural efficiency of program comprehension. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 140–150. ACM, 2017.

15 Andreas Stefik, Bonita Sharif, Brad. A. Myers, and Stefan Hanenberg. Evidence About Programmers for Programming Language Design (Dagstuhl Seminar 18061). *Dagstuhl Reports*, 8(2):1–25, 2018. doi:10.4230/DagRep.8.2.1.

16 Andreas Stefik and Susanna Siebert. An Empirical Investigation into Programming Language Syntax. *Trans. Comput. Educ.*, 13(4):19:1–19:40, November 2013.

17 Jeffrey Stylos and Brad A Myers. The implications of method placement on API learnability. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 105–112. ACM, 2008.

18 KP Suresh. An overview of randomization techniques: an unbiased assessment of outcome in clinical research. *Journal of human reproductive sciences*, 4(1):8, 2011.

19 Federico Tomassetti and Marco Torchiano. An empirical assessment of polyglot-ism in GitHub. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 17. ACM, 2014.

20 Arie Van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.

21 W Quin Yow, Jessica SH Tan, and Suzanne Flynn. Code-switching as a marker of linguistic competence in bilingual children. *Bilingualism: Language and Cognition*, pages 1–16, 2017.