Ethical Considerations of the Typma Programming Language Project

Sacha Peterson
speterson@oxy.com
Occidental College

1 Introduction

As a project, the Typma programming language does not raise many ethical issues. Typma is based around variables being randomly typed when unspecified in print statements, function parameters, and any algebra/boolean expression. The Typma project is based around programming language theory with no focus on practical use that could impact society. However, there are some accessibility problems that could arise with understanding the code base for the parser and interpreter for Typma. These accessibility concerns are the general English language barrier that exists for most programming languages and a knowledge gap with functional programming concepts that are a basis for Typma's backend. In addition, while Typma is not meant to be used by anyone for practical programming, there is value in exploring problems that would occur if Typma was used in the real world.

2 Language Barrier

Computer science as a field has a massive bias towards the English language with the use of English for keywords in programming languages. Python, C, C++, Java, Javascript, Haskell, OCaml, and Rust all have English as the language of the language. Typma will be no different from these mainstream languages by also using English keywords. While this poses no issues for native English speakers, people for whom English is a second language or people who do not know know any English are at a major disadvantage when it comes to becoming an experienced computer scientist as seen in Ashok Kumar Veerasamy (2014) [1]. Although it may seem like non-English speakers are going to be excluded from understanding the parser and interpreter for Typma, everyone who knows how to program and uses a mainstream programming language already knows English keywords. The reality of computer science is that English is the standard language and the intended audience of Typma being people with at least a basic understanding of computer science, anyone who would want to look at Typma would not have a problem with English.

Although solving the language issue is not relevant and out of scope for the Typma project, an alternative project that I am considering as a backup to the Typma project would be a simple programming language based on keyword language inclusion from non-English languages. The project would be a lexer that would have different versions for a variety of languages for the keywords. This is similar to how movies, video games, and websites have toggles and options that support other languages. The scope of this project would support five languages: English, Japanese, French, Spanish, and German. Programming languages are based on concepts that are not tied to the English language so the language of the keywords can be changed without changing the function of a programming language. Focusing on five languages would provide enough of a variety to show that programming languages can become more accessible without implementing so many languages that would take more time than is allotted for COMPS projects. Special attention would be given to the Japanese component including different modes based on the three writing systems of Japanese. Another potential feature of this project would be a parser separate from running the code that could convert keywords to any desired language. This could be useful for a team that has members who read different languages better so a simple program could swap the keywords for language that is easier for any member to read. Although people who know how to program now already know English keywords and this project would not help them, this project is a proof of concept to show how in the future people would not have to learn some English just to learn programming.

3 Knowledge Gap

The Typma project is something that can be done in almost any programming language, with the language of choice for this project being OCaml. There is nothing significant about which programming language the Typma parser and interpreter are being written in except there are many design choices and features that do not translate to other languages outside of the functional programming paradigm. This is an accessibility problem because many programmers do not have the experience with functional programming to be able to understand some specific methods of the Typma project or may even have a difficult time reading and understanding the code in general.

Programmers have problems when learning how to use a new programming language in general as shown in Parnin (2020) [3]. Something that can get in the way of learning a new programming language is prior knowledge from another programming language. For example if you are a Java programmer trying to learning Python, a problem that could happen is the habit of placing a semi-colon at the end of every statement will make getting used to Python's lack of semi-colons with this usage more difficult.

The surface problem of learning a new language will happen for anyone trying to understand the code of the Typma project who does not know OCaml. But what makes this an accessibility problem is that Typma's implementation uses features of functional programming. One of these features is declarations of custom types. Type declarations are important to how the different types of statements, like a print statement or a function declaration, are categorized in Typma programs. Typma commands are represented with a type called "cmd" which has different variant depending on what type of expression it is. Here is what the cmd type will be declared as:

```
(* types in typma *)
type typma =
  | int
  | bool
  | str

(* function parameters *)
type param = string * typma option

type cmd =
  | Skip
  | Print of aexp
  | Ass of string * aexp
  | Seq of cmd * cmd
  | If of bexp * cmd * cmd
  | While of bexp * cmd
  | Do of cmd * bexp
  | Fun of param * cmd
```

Languages like Python, Java, and C++ do not have as expressive algebraic data types. Those languages do have objects, but objects are very different from type declarations and trying to understand type declarations as if they were objects defeats the purpose of trying to understand functional programming. Although the programming language being used would not normally affect accessibility to a large enough extent to be noticed, the Typma implementation of functional programming features would make it difficult for someone without a basic understanding of functional programming to understand the methods of the project.

4 Typma in the Real World

It is important to stress that the Typma programming language is not meant to be used for anything practical, but it is worth it to see how the logical extreme of a bad type system could cause ethical issues if it was ever used. These problems go past making the developer's life harder; if any unspecified data types make it into a released version of software written in Typma, people's lives could hang in the balance of whatever the random number generation decides.

For example, the paper Fitzpatrick (2021) [2] explores ways to ethically make mental health apps with unaccompanied migrant youth in mind. If some version of the Typma programming language was used to make this mental health app, it would be more difficult to make the app function, preventing mental health from being addressed. If the app featured some kind of notification that was time based, like a wellness notification at 9:00 a.m., and the programmer did not specify the type properly to avoid random typing, the notification only has a chance to be played at the right time depending on how the interpreter evaluates the unspecified type. Such a language that allows problems like this to slip by without being noticed could lead to the youth using the app randomly to not see the notification at the right time or not at all, thus failing to help the youth who need help.

A different language would have a type error that would flag this kind of issue. A language like Java requires types to be specified upon declaration or an error will be generated. Typma however will run whether or not the type is specified, leading to situations that programmers do not intend to happen. In a perfect world the programmer would not make the error of forgetting to specify types in accordance to proper Typma conventions, but that is unrealistic to assume that programmers are perfect and if Typma was used these issues would bleed into the real world. Typma is too dangerous to be used because of potential ethical issues it can cause by hurting the basic functionality of apps.

5 Conclusion

The Typma programming language parser and interpreter project does have some limited accessibility problems. Someone who does not read English would not be able to understand Typma programs, but someone who knows programming but does not read English keywords is unlikely. Someone who does not have any familiarity with functional programming would have a hard time understanding the code in the parser and interpreter for Typma. If Typma was ever expanded to be used in software that could have an effect on people's lives, one missed specification for a type could cause problems that adversely impact people's lives. The Typma project is intended to show the theoretical limit of type systems for people who are able to understand the

basics of programming and computer science. This limits the number of people for whom Typma would hold any purpose and ethical issues that could arise from mass usage.

The only consequential hurdle of Typma is the lack of widespread knowledge of functional programming. Although functional programming is not the standard for most fields in computer science, functional programming is becoming more and more common and computer science departments like the Occidental College CS Department should strongly consider either adding a class that teaches the basics of functional programming or add functional programming to an existing class. Functional programming can be very tricky to learn and is more complex than just teaching yourself a new language that uses familiar concepts. A course about functional programming is a great way to make the Typma project more accessible as well as other functional programming based projects.

References

[1] Ashok Kumar Veerasamy, Anna Shillabeer. "Teaching English Based Programming Courses to English Language Learners/Non-Native Speakers of English". In: *RMIT University* (2014).

[2] Fitzpatrick, Franziska Tachtler Reem Talhouk Toni Michel Petr Slovák Geraldine. "Unaccompanied Migrant Youth and Mental Health Technologies: A Social-Ecological Approach to Understanding and Designing". In: *ACM* (2021).

[3] Parnin, Nischal Shrestha Colton Botta Titus Barik Chris. "Here We Go Again: Why Is It Difficult for Developers to Learn Another Programming Language?" In: *ICSE '20* (2020).