Electronic Portfolio

Miles Burkart

10 December 2024

Revised Annotated Bibliography

A. Tretyak, E. Vereshchagina, S. Fadyushin, T. Dobrzhinskaya; Cognitive ergonomic approach to programming language design: Developing an ergonomic syntax structure. AIP Conf. Proc. 13 October 2023; 2910 (1): 020040. https://doi.org/10.1063/5.0167001

This paper focuses mostly on programming language syntax structure by defining terms like clarity, reliability, flexibility, simplicity, and mobility. It then specifies a new language which aims to resolve some of the issues that are seen in current popular languages. This paper is peer reviewed and was written very recently. This paper would be useful to me in determining what kind of syntax should be used for a good programming language. The information from this paper could be integrated into the syntax section of "Programming Language Design Concepts" for a more in depth look at syntax design.

Chen, Zhe, et al. "A Smart Status Based Monitoring Algorithm for the Dynamic Analysis of Memory Safety." *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 4, 2024, pp. 1–47, https://doi.org/10.1145/3637227.

This paper addresses some of the greatest shortcomings of the C programming language, which is presently one of the most dominant languages for low-level software. The main shortcoming comes in the form of memory errors that occur due to the unsafe nature of the language's memory allocation system. The authors then propose a specification for a new monitoring algorithm that will be able to dynamically analyze a C program in order to locate and track down any memory errors.

"How Do You Write Your Own Programming Language? - Sololearn." Sololearn Blog, 3 Nov. 2022, www.sololearn.com/blog/write-programming-language/. Accessed 10 Sept. 2024.

This blog article provides basic considerations for newcomers to programming language design. This includes whether the language should be compiled or interpreted, basic syntax structure, and the target audience for the language. This article is written by a computer science and IT related blog, and includes basic yet valid theories about language design. This could be useful for introducing programming language design using simple ideas. This source could be used in conjunction with the following ones for a more comprehensive introduction to simple language design concepts.

Jung, Ralf, et al. "RustBelt: Securing the Foundations of the Rust Programming Language." *Proceedings of ACM on Programming Languages*, vol. 2, no. POPL, 2018, pp. 1–34, https://doi.org/10.1145/3158154.

This paper focuses on a newer programming language called Rust, a language that aims to provide the user with safe, low-level control over the computer alongside high-level features. The authors state that none of Rust's safety claims have been formally mathematically proven, so they are aiming to validate some of these claims by creating proofs for a reasonable subset of the language. These proofs would strengthen Rust's credibility as a more safe language than current low-level options, and would provide a good opportunity to talk about the future of safety in programming language design.

Kramer, Nimrod. "Create Programming Language: Design Principles." Daily.dev, 8 Apr. 2024, daily.dev/blog/create-programming-language-design-principles.

This article describes general considerations for amateur programming language design as well as some core elements such as purpose, readability, expressibility, efficiency, scalability, human-centric design, and extensibility. It then goes into detail about each of those elements and how each of them individually help to improve the design of the language. This article is written by a software development blog, whose author has experience in IT and Computer Science. This article would be similarly useful for an introduction, since it is written in a way that is simple to understand. The information from this article could be paired with another source in order to introduce the reader to language design.

Watt, David A. (David Anthony), and William Findlay. Programming Language Design Concepts. John Wiley, 2004.

This book is a complete comprehensive study of programming language design which includes insight into nearly every component of most traditional languages. It also states that languages should be both universal (can solve any solvable problem) and implementable (can execute any well-formed program). The book was written by two computer scientists, and while it was published in 2004, much of the information still applies to modern design–especially considering that many of the languages referenced in the book are still widely used today. This source will be particularly useful for general information and understanding of overarching concepts which require a more in-depth explanation. This source can be connected with most of the other sources given that it contains a good amount of general knowledge.

Two Artifacts

Continued Research Planning and Ideas – September 6th

In the field of Computer Science, one subject that I've always been interested in is programming language design. Designing a programming language is much like designing a spoken language; one must define grammar, syntax, rules, and patterns, while also considering readability, conciseness, performance, and overall user experience. I think that making one is an interesting exercise in logical design.

Question:

- What characteristics define a good programming language?
    - Readability
        - Users are able to quickly grasp the function or idea of a piece of code when first reading it
        - Programmers spend a lot more time reading code than writing code, so this is an important quality
    - Functionality
        - Language must ship with a decent feature set and common quality of life features
    - Learnability
        - Should be quick to learn; not too many reserved words or memorized patterns
        - Should contain familiar patterns for experienced developers

A separate topic that I'd also like to research could be martial arts. I've been practicing Taekwondo for over ten years now, and I think that the experience has had a profound effect on

me both physically and mentally. I could look into the effect of martial arts specifically on confidence and discipline.

Question:

- What kind of effect does martial arts have on the human psyche?

Project 2 Outline and Visions – October 21st

Project 2 Paragraph Outline:

I. Introduction
II. Event that caused object to be made
III. Description of object, explain significance to my topic
IV. Event that caused text to be written
V. Description of text, explain significance
VI. Possibly compare and contrast the two items chosen
VII. Conclusion

In general, this topic doesn't get a ton of publicity, so there aren't many popular forms of media that cover it. There are, of course, videos online about it, but so far, I haven't found anything that's fully developed and would be of use in this essay. I'll be sure to look further into this as I continue to work on this project.

The audience for this topic in a popular format is certainly going to be programmers or programming enthusiasts, since that's essentially the only group to which this topic is relevant. The primary audience for the text that I chose, which is "The C Programming Language," differs depending on the time period which is being analyzed. When the language was first created, it

was a high-level language designed to make programming easier, making it simpler to do more powerful things with the computer. Nowadays, however, C is known as a low-level programming language, used primarily by programmers whose use case requires high efficiency and increased control over the computer's hardware.

"The C Programming Language" was written to help with the creation of the Unix operating system, so I would say that was the moment that caused its creation. It didn't provide any immediate extreme changes to the development process, but over time it made the process much more streamlined.

Reflection

What have you been focusing on and discovering throughout the semester?

I was surprised to learn that we would be able to choose our own topic of writing in this class, and I decided to use this opportunity to learn more about a topic that I'm interested in, which is programming language design. Creating my own programming language was one of the first large projects that I started when I was first getting introduced to software development, and I've since discovered much more than I knew back then. However, at the time of starting this class, I wanted to gain a more comprehensive understanding of these languages: what made some languages better than others? Why have some languages been made obsolete? What are new languages doing to fix the problems seen in the old ones? I currently have a much greater understanding of this subject and I feel more confident in my ability to create and understand programming languages and their design.

What questions are still unanswered?

I currently have a pretty good understanding of the main topics that make up this subject, but I would have liked to look more into some non C-like languages since they make up a large portion of the field and I don't have much experience with them. Specifically, I'd like to learn about database related languages like SQL as well as functional languages like Haskell or Erlang.

New communication style learned in Project 3?

I decided to use a tool called Manim to create the video for Project 3, which is a tool that allows for the creation of animations using only Python code. I hadn't used it prior to this

project, so it took a bit of time to learn it, but I learned a lot and I'll now be able to use it in future projects. I hadn't created an actual edited video in quite some time, so this was a good opportunity to brush up on my skills. My main concerns when creating the video, aside from the content, was the sizing and timing; just making sure that the video was properly paced and that I wasn't cramming too much information onto the screen at once. I also chose to build the website from scratch to improve my frontend development skills, and that also took a bit longer than I'd anticipated. It gave me a chance to learn more about the styling and layout of a website, something that I had limited experience in. Overall, I'm glad that I opted to focus on things that were new to me for this project, and I now have some new tools to utilize my projects in the future.

How do you work effectively?

I find that I work most effectively if I work on a project incrementally. The quality of my work often suffers if I try to complete a large project in a single session over multiple hours. I'd rather work on a project for a couple hours at most every day so that I have more time in between to fully flesh out my ideas. As for my working environment, I'll always prefer a quiet place with few distractions, and I usually find it difficult to work if there are any auditory distractions, including music.