Edward Li
English 3307
Tom Akbari
Unit 1 Rough Draft
12 September 2019

Convincing a Skeptical Real World

In 1989, John Hughes of the University of Glasgow published *Why Functional Programming Matters* in *The Computer Journal* with the unenviable task of convincing the world that functional programming, a style of programming ridiculed for its "simplicity" and use only in "academic" settings, pushes aside limits imposed by modern languages on the modularity of programs, and allows programmers to create more powerful programs with less components. When the paper was released, object-oriented programming languages (another programming paradigm) were quickly taking over and seemed to be the direction that the industry would take in the future. Hughes attempted to make a new argument as to why functional programming's unique properties lent it the ability to make cleaner and shorter code. While in the end, object-oriented programming still "won," the paper has garnered enough attention to become a "must read" in the Computer Science world, making it a worthy specimen to study. In the paper, Hughes makes a compelling case for his argument, all the while spending language on convincing skeptical programmers through compelling hooks, and simple language, and globally recognized examples.

The paper's opening salvo is "This paper is an attempt to convince the 'real world' that functional programming is vitally important, and also to help functional programmers exploit its advantages to the full by clarifying what those advantages are. This immediately shows the doubt of functional programming."[1(p.1)] Here, Hughes lays out the first issue he perceives he will encounter with his potential audience; they think as "real world" programmers, the contents of the paper would probably be useless to them. It's the first of many mentions of the struggle in attempting explain to skeptics functional programming's benefits. This opening line, and the context under which the paper was written, reveals that Hughes thinks of his view as the underdog, as something that he'll have to fight for. It's something to keep in mind while going through the rest of the paper; Hughes perceives his audience as people who wouldn't be familiar with the concepts he's presenting, and people who would want to ridicule his paper. This is clear in his language, with the sarcastic quotes around 'real world,' followed up with "this shows immediate doubt of functional programming," he knows that people reading the paper are going to doubt his claims.

Hughes then makes the claim that what people traditionally consider the strengths of functional programming aren't the true benefits of functional programming. "Even a functional programmer should be dissatisfied with [their] arguments… Clearly this characterization … is inadequate. We must find something to put in its place – something which not only explains the power of functional programming, but also gives a clear indication of what the functional programmer should strive towards."[1(p.1)] This quote shows that the intro's purpose is to not only convince skeptics to push forward, but also to hook those familiar with functional programming to gain new insight on the argument for functional programming. It also serves as a call-out to other functional programmers, saying that they aren't focusing on the right things when they're trying to argue for the benefits of functional programming, as seen in the aggressiveness of his tone. He strengthens this by appealing to the criticisms other people also have of functional programming, by making sarcastic claims about functional programmers appearing as "medieval monk[s], denying [them]sel[ves] the pleasures of life in hope that it will make [them] virtuous."[(1,p.1)] He appeals to both skeptic readers by building an understanding, telling them "hey, I get where you're coming from, why you doubt us, but hear me out," drawing in the reader more. By appealing to the audience, Hughes shows a level of empathy that causes the reader to want to join him in exploring the topic he has on hand, and satisfies Hughes' want to bring in more interest.

After getting common myths, misconceptions, and weaker arguments out of the way, Hughes iterates the basics of his point of view — instead of focusing on the features of the languages themselves, he focuses on the glue that holds parts of the language together. Hughes first spends time discussing how it's "generally accepted that modular design is the key to successful programming,"(1,p.2) and even lists examples of non-functional programming languages that provide some level of modularity. However, he then states that the key advantage to functional programming aren't the languages themselves, but the glue that holds them together. And for that, he uses an analogy about how "a chair can be made quite easily by making the parts… and sticking them together in the right way. But this depends on the ability to make joints and wood-glue. Lacking that ability, the only way to make a chair is to carve it in one piece out of a solid block of wood, a much harder task."(1,p.2). Hughes started to home in on the idea of glue throughout the intro, but he chose a chair analogy specifically to show the difference between the difficult of assembling vs. carving something monolithic. The chair analogy flows perfectly from his claims about the glue of languages – being about literal glue and parts of a chair rather than the metaphorical glue and parts of a language. It keeps the paper's linguistic sensibilities, while also giving a perfect example on how difficult he perceives programming in other styles to be, and why – in his brain – functional programming wins out.

Section 3 and 4 spend time explaining the concepts Hughes lays out in the abstract. However, instead of filling pages with jargon, Hughes uses a combination of English and diagrams to notate his point. Even through section 4 and 5, when Hughes is explaining the "practical" examples, he resorts to using more diagrams[1,pp.2-7] and English than code. This is on purpose – using code of a language that not many people know would scare off reader. He knows that even though he made the claim that the language his examples in is "relatively easy to understand," people would still run away if they just saw syntax of a language they didn't know. However, by using a combination of diagrams that most computer scientists are familiar with, as well as clear concise language, Hughes finds a universal method of communicating to readers the advantages he wants to semantically. This, in tandem with using practical examples and even diagrams to show off trees, and tic-tac-toe boards, things recognized universally, allows Hughes to dive into technical topics without scaring newcomers to his paradigm, edging him closer to achieving his goal of not scaring people off, and drawing in new readers.

Hughes' ability to address a skeptical audience, make simple analogies to highlight small points, and draw non-experts in the field in with easily understandable, general examples allows him to create a more convincing argument than his peers' attempts in the same vein. The real worth I found in the paper isn't in the content itself, but how it's able to make an argument in computer science. CS is a field which is currently riddled with an issue of the inability to communicate opinions. From what I've experienced, when two people disagree on something in CS, they resort to throwing large, complex phrases and reasons at each other to make each other's argument look "smarter," and "more reasonable" than each other's. However, in this paper, Hughes breaks down his argument and viewpoint in a clear, concise way, that's easy for almost anyone to understand, using a combination of intriguing language and well-placed diagrams, as well as using interesting examples, and adding a huge bow-tie on it all with a nailbiting hook. It’s a paper I will revisit later in my career whenever I want to be able to articulate a point and turn discourse to my favor that I hope to be able to apply.

Reference

1. John Hughes. Why Functional Programming Matters. November 1988. (accessed 09/12/19) Found at https://bit.ly/2kLIXpJ