Non-programmers are a poor fit for the Wikilambda project

None of the participants from the Wikilambda research could imagine substantive contributions for non-programmers beyond administrative tasks or language translation skills. These participants simply assumed that the project was for more technical people. They're not wrong. The Wikilambda project will depend on people who already know how to code.

Wikimedia is not the first organization to explore the potential for non-coders to become programmers. This position paper argues against that idea.

Section overview

- ➔ Programming logic is an obstacle
- ➔ Some people are non-technical by choice
- ➔ Why recruit non-programmers?
- ➔ Wikilambda diversity
- ➔ Conclusion

Programming logic is an obstacle

The Wikilambda study didn't focus on computer science education but one beginning-level coder spoke about the problem and theorized that programming logic was the key factor that separated programmers from non-programmers.

> **Maciej**: The difficult thing for beginners is understanding the logic process. If this happens, then do that. I think that people may have the same problem here at WikiLambda because of this lack of understanding. If something happens, then the program should do this.
>
> **Maciej**: It's the same thing in Excel. Most people don't like Microsoft Excel because they cannot follow the logic principle.

Others have articulated this idea in more detail. Are there habits of mind that are required for programming which are not found in the general population? If so, this could be an obstacle for non-technical users, regardless of the Wikilambda interface.

Quora: What are the basic characteristics of a computer programmer?

> Be able to **think analytically**. If you've been analyzing things since you were a kid you have the potential to become a great programmer.
>
> An algorithm is a set of steps describing a process. Lots of programming is simply describing algorithms to a computer. Having knowledge in data structure and algorithmic thinking means having good capacity for how to store and manipulate your data, being able to **think both iteratively and recursively**, and being able to understand code performance.
>
> You need the ability to think logically, programming is **mostly about decision flows.** The ability to visualize the process in mind (or on paper!) is important.

> You can acquire all the technical skills in the world but not be a good programmer if you lack the **creativity required for problem solving**.
>
> Able to **visualize the system design** in your head.
>
> **Desire to learn**, unlearn and relearn.

Quora: What are some reasons people fail to learn programming?

> Some people have **absolutely no logic skills** or very limited logic skills. These people will never be able to complete even a moderately complex programming task.
>
> Other people hate programming because of the **attention to detail** required.
>
> The core skill is being able to **think in abstract terms** down to step-wise detail and break a problem down into pieces.
>
> Programming is basically addressing one big problem as thousands of little problems. Being able to **mentally divide big tasks** into the tiniest imaginable steps.
>
> Programming is a useful exercise (as is Math), because it's **ALL about abstraction**. You can't escape from abstractions in programming. You're forced to create them and deal with them, so you're forced to learn them well.
>
> A lot of people who want to learn programming are often looking for easy outs and simple steps. **Patience and perseverance** are not in their makeup.

Some people are non-technical by choice

For non-coders who have the capacity to learn programming, there are already many resources on the internet to pursue this skill. Indeed many of the programmers from this research had taught themselves how to write code.

> **Richard**: Well, it's easy to teach yourself Python. There are plenty of tutorials and things. And of course I've many times learned how to use different programming languages.
>
> **House:** When I shifted to Python two years ago, I have found plenty, plenty of things online. So, it took me only six month to shift that, to learn all the stuff. It was quite simple.

Wikimedians who edit wikitext and utilize scripts and templates have demonstrated a capacity for technical challenges. If they wanted to learn code they could do so.

So why haven't they become programmers already? Many have told themselves for years that they are not technical people. This is a self-imposed obstacle arising from past experience. **Learning to code is simply not a part of their identity.**

Non-technical contributions are within their means and a few may try to write functions but the primary technical contributions will come from programmers.

Why recruit non-programmers?

The purpose of recruiting non-programmers for Wikilambda has always been to increase the demographic diversity of the initiative. Organizations traditionally value demographic diversity because this also brings a diversity of perspectives. But in this case diversity is an operational requirement. For Abstract Wikipedia to be successful it depends on participants who speak hundreds of different languages. The argument is that these languages are not likely to be found in the traditional programmer demographic.

But Wikilambda itself has no language requirements. Those only apply for Abstract Wikipedia. It's entirely possible to collaborate with native speakers of the target languages without requiring them to write code or participate in Wikilambda at all. A language-diagramming UI could be created that solicited input from native speakers and fed their responses into the system automatically. This would eliminate the need to write functions on a mobile phone and make it easier to reach non-technical language experts.

From a language perspective it's much more efficient to identify the target languages and then recruit native speakers. This would still be a significant challenge but it's a better strategy than recruiting thousands of non-coders who can't meaningfully contribute unless they happen to speak an obscure language.

Even if there were other reasons for Wikilambda to value diversity, recruiting non-programmers from the Wikimedia sister projects wouldn't necessarily solve this problem. These projects have their own demographic biases. There's no guarantee that non-programmers inclined to learn coding for this project would be any more diverse than the actual programmers already recruited from the same ecosystem.

Recruiting non-coders from outside the Wikimedia ecosystem would solve the bias problem but there's no reason to think they would be interested.

Wikilambda diversity

An alternate strategy is to consider the demographics of people who already know how to code but *do not think of themselves as programmers*. Biologists, mathematicians and information scientists bring a diversity of perspectives that reach beyond the developer subculture. This should be the focus when it comes to expanding the diversity of Wikilambda. The advantage is that these people don't need to be convinced to care about code. They just need to understand how Wikilambda can benefit their community of practice.

> **Katherine**: I write code as part of every day, but I do not consider myself a professional programmer. I wasn't hired just to do programming. I'm an information scientist at a cultural heritage organization. I have a PhD in information science.

> **Marc**: I needed Python to do my PhD. Actually, although I code professionally to develop my research, I do not consider myself a developer because I'm not creating applications in a developing pipeline. I code to obtain results, not to optimize, so I'm happy with that.

> **Krishna**: So I'm not a hardcore developer. Like, I don't do application development, but just a few things to get my work done.

The most obvious solution for expanding diversity is to recruit programmers from more diverse locations. Programmers exist worldwide. It should be straightforward to identify the Wikimedians who care about code in non-western countries and evangelize the Wikilambda project through direct outreach or by partnering with local groups.

Conclusion: Optimize for programmers

One of the advanced programmers from this study made a good argument about why Wikilambda is inherently meant for people who understand code.

> **Marc**: One of the basics of user experience design is speaking to the needs of the audience. This is speaking to the coder more than the user. The coder should be able to read all the different parts and understand how they connect. It's already assumed that they will code. To a *certain extent, we cannot hide what we are doing*.

> **Marc**: I think that for a non-programmer or someone non-technically savvy, this is not helping. I see that it's difficult to provide tools to non-programmers without a *technical minimal qualification*.

Wikilambda is a complex undertaking that requires input from highly technical people. There's no indication that non-programmers could make substantive contributions to the code or are even interested in trying. Entry-level tinkering is very different from developing a working application in concert with other programmers.

The project interface and architecture should be optimized for the people most likely to care about the subject matter. This means focusing on the needs of the programmer. Future research should include programmers from many different industries along with those who write code but don't identify as developers.

It's also important to reach beyond the Wikimedia ecosystem. Is it possible to create a coding system built on the MediaWiki platform that makes sense to outsiders? Are there ways to optimize the existing UI for a more technical audience or do they require a different solution in order for Wikilambda to succeed?

It's entirely appropriate for a niche project to have a niche audience and interface. Trying to serve the needs of everyone would be a recipe for failure.