**Coders Aren’t Assembly Line Workers**

* Published on March 27, 2017
* Featured in: [Editor's Picks](https://www.linkedin.com/pulse/feed/channel/editor_choice), [Software Engineering](https://www.linkedin.com/pulse/feed/channel/software)

[**David Max**](https://www.linkedin.com/in/davidpmax)

**Senior Software Engineer at LinkedIn**

Clive Thompson wrote a thought-provoking piece in Wired, [*The Next Big Blue Collar Job is Coding*](https://www.wired.com/2017/02/programming-is-the-new-blue-collar-job/?utm_content=buffer5cd23&utm_medium=social&utm_source=linkedin.com&utm_campaign=buffer).

The traditional definition of [blue collar work](https://en.wikipedia.org/wiki/Blue-collar_worker) is manual non-agricultural labor. Examples include coal miners, aircraft mechanics, longshoremen and assembly line workers. Some blue collar occupations require highly skilled workers with lots of formal training, while others require minimal skills.

By the traditional definition, my job would be considered white collar. If you can say that about a workplace environment that’s more like “collar optional.”

So when Thompson says that in the future coding will be a new category of blue collar job, he obviously doesn’t mean that future software developers will need to get their hands dirty. I think this is what he is saying:

* There is a growing category of coding jobs that will be well-suited to people who either currently have blue collar jobs or would have planned to gone into blue collar jobs in the past.
* These new jobs will be open to people without four year computer science degrees.
* These new jobs will offer long term stability and middle class wages for coders who will write the kind of everyday code that doesn’t require superstar abilities or an advanced degree.

**The image of the coding superstar**

When I ask people to picture a coder, they usually imagine someone like Mark Zuckerberg: a hoodied college dropout who builds an app in a feverish 72-hour programming jag—with the goal of getting insanely rich and, as they say, “changing the world.” *— Clive Thompson*

When I got started as a software engineer, if you were to ask people to picture a coder, they would also picture someone like Mark Zuckerberg, except instead of wearing a hoodie, he’d be wearing glasses and a button-down shirt, possibly with a [pocket protector](http://www.americanscientist.org/issues/pub/rise-and-fall-of-the-pocket-protector).

If software developers of the future shed some of their hoodied superstar image, that will be just a return to the past.

**Lower barriers**

My first programming job was at a technology company located along Boston’s famous Route 128 high tech corridor. It was a summer job I took after finishing grad school. I remember noticing at the time with some amusement that the HP Apollo computer workstation on my desk had probably cost the company about as much money as a year’s worth of my salary. It was definitely not the sort of machine a person would buy for their home to pick up programming in their spare time.

But that was the last time. At every software development job I’ve had since then, my main work desktop machine was a commodity PC running some version of Windows or Linux. At one time one needed to be at a university or work for a high tech company to get access to expensive computers and software development tools, but now it is easier and cheaper to acquire the tools necessary to learn programming.

**You don’t need a computer science degree to be productive**

I once worked at a company that produced software that banks used to create home mortgages. We were a small team that included several software developers, but also people who were experts in mortgage banking. Some of them had previously had jobs in banks as underwriters or loan officers.

There are lots of rules about how mortgages worked that our software had to reflect, like if this field was filled in one way, then the loan was not [FHA compliant](http://www.willitsnews.com/article/ZZ/20121119/NEWS/121116132), and then you needed to fill this other field over there, and so forth. If we had been a bank, we probably would have had the mortgage bankers write up a whole lot of specifications, and they would try to explain all these rules to us software developers, and we’d try to implement them.

But that’s not how we handled it. Instead we had a simplified programming language built into the application itself that handled business logic, and we showed our mortgage experts how to use it.

The original thought was that most of this code would be short expressions or one-liners, but as the needs of the application grew, and the level of comfort of the mortgage experts grew, more features were added to the embedded programming language, and the size and sophistication of the business logic code grew as well.

Effectively, everyone in the company was a coder, even though only some of us actually coded in C++ or had a formal computer science degree. Even the employee who had started off as an office and project manager became a coder too.

The real power of this arrangement was that our mortgage experts who really understood the business rules had the tools to actually implement them. They could see the effects of their changes immediately, and they didn’t have to go through a cumbersome process of trying to explain their domain expertise to a software developer to get every little change done.

If I had any advice to someone who wants to learn coding as a way to retrain themselves from whatever their current expertise is, I would say look hard for ways to leverage what you already know. What you may lack as a novice programmer can be offset by other deep domain knowledge you may already have.

**Agile Development is a form of Lean Manufacturing**

Many software developers practice a form of [Agile Development](https://en.wikipedia.org/wiki/Agile_software_development). Agile practices are themselves a subset of what are known as [Lean principles](https://www.lean.org/WhatsLean/), which were developed by Japanese manufacturers and whose practices have spread widely in the US automobile industry and elsewhere. If a blue collar worker at an automobile factory is currently practicing lean manufacturing, that worker may be surprised to find out that lots of software development [works by the same principles](https://www.forbes.com/2010/01/11/software-lean-manufacturing-technology-cio-network-agile.html).

**On the other hand...**

“But any blue-collar coder will be plenty qualified to sling Java­Script for their local bank.” — *Clive Thompson*

The implication is that one can learn enough coding skills to get a job writing easy code, and then settle into a long stable career writing more of the same. Maybe, but I doubt it.

The world of software development changes rapidly. Even if we need a lot more developers, that won’t change the fact that keeping up is a continuous effort. Even if the barriers to entry are getting lower such that there are programming jobs that do not require a four year computer science degree, that is still just the foot in the door. Once on the job, you will need to keep actively improving to keep up like the rest of us.

“Somehow, they've forgotten that what software developers do best is learn.” — [*Jeff Atwood*](https://blog.codinghorror.com/the-years-of-experience-myth/)

To put it another way, there may be more entry level positions open to people with less training, but that training will likely grow stale. A boot camp program might prepare you for a job, but what you learn today in a boot camp will almost certainly be different from what you would learn two years from now.

“The more JavaScript code I write, the less secure I am in thinking I have a handle on the language. Just when I settle on a way of doing something, a startlingly better way occurs to me.” — [*The Object Slinger*](https://dreaminginjavascript.wordpress.com/2008/06/27/the-object-slinger/)

What about Devon? He’s the programmer mentioned by Thompson who helps maintain a ­security-software service in Portland, Oregon. He describes his job as stable and rewarding. I believe him, but if you look under the surface, you will see that his field doesn’t stay still for long.

Think about all the recent news items you may have heard about [Vault 7](https://en.wikipedia.org/wiki/Vault_7), [Russian hackers](http://www.cnn.com/2016/12/12/politics/russian-hack-donald-trump-2016-election/), [corporate data breaches](https://en.wikipedia.org/wiki/Yahoo%21_data_breaches), and the like, and you realize that there is nothing placid and static about running a [security software](https://en.wikipedia.org/wiki/Computer_security_software) service.

I would imagine that keeping up with the field of software security would be a continuous stream of new challenges, which is exactly the sort of thing that makes a job interesting over the long term.

**The elimination of toil**

Thompson is not the first one to observe that there is a lot of coding that doesn’t require superstar developers. It is tempting to extrapolate that in the future we’ll need a lot more of that kind of code, and therefore in the future we’ll have lots of people employed writing lots of easy code.

That prediction has been made before, and every time it failed to materialize. It could be that this time is different. But I doubt it.

“Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows.” — *Vivek Rau*, [Eliminating Toil](https://landing.google.com/sre/book/chapters/eliminating-toil.html)

For virtually every software development project, there is some part of the code that is so routine that one does not need a lot of skill to implement it. Once the project is a success, a new, more ambitious project is planned. If you were to use the same process as the original project, that new project might require 20 times as much of this routine code.

You might think that if the first project needed 2 developer to do the routine coding tasks, that now you are going to need 40 developers. Okay, go hire another 38 JavaScript slingers!

But that’s almost never what happens.

Instead, you take some of your best programmers and try to figure out a software solution to reduce your routine code problem to something more manageable. This effort might not have been worth it on the original project, but at the scale of the new project, it’s worth the effort of two expert developers to devise a way to eliminate the need for 38 more novice developers.

In fact, developers have been known derive a certain perverse pleasure in devising ways to automate any repetitive process. This is because doing something repetitive 100 times may be easier than building a tool to do it for you, but building a tool is fun, and doing something 100 times is boring. This is known as “[inspired laziness](https://swizec.com/blog/programmers-are-fucking-lazy/swizec/2648).”

“If you want it done the same way every time, and with any semblance of reliability, you want the human factor removed as much as is reasonably possible.” — *Jeff Atwood,* [*Get Me The Laziest People Money Can Buy*](https://blog.codinghorror.com/get-me-the-laziest-people-money-can-buy/)

This pattern has been repeated routinely through the history of software development.

If you are that novice developer writing routine code, and you hear that a project is coming down the pike that will give you 20 times more work to do, this is not the time to rejoice. Your job is not safe.

If the company you work for suddenly hires another 38 novice programmers to deal with the big new project they’ve got coming down the pike, this is *definitely* not the time to rejoice. Your company is not safe.

If you want to become a coder for the long term, your first goal may be to get a job, but then your next immediate goal is to graduate beyond the novice stage.

In the future, there will surely be more software developers coming from all backgrounds, blue collar included. But I don't see them joining a new permanent class of career “JavaScript slingers.” I expect that many of them will become the future’s experts and superstars too. Hoodie optional.