

DPI

De Programmatica *Ipsium*

DE PROGRAMMATICA IPSUM

Issue 012: Being A Senior Developer

September 2nd, 2019

Table of Contents

Issue 012: Being A Senior Developer	5
At Least	9
Enough Agile	17
Not Being A Developer After 46	23

Issue 012: Being A Senior Developer



September 2nd, 2019

Welcome to the twelfth issue of *De Programmatica Ipsum*, dedicated to the subject of *Being a Senior Developer*. In this edition:

- Daniel Steinberg explains how forty years of experience in the field is just the beginning¹.

ISSUE 012: BEING A SENIOR DEVELOPER

- Graham argues that “agile” methodologies have reached senior status at the age of twenty².
- In this issue’s subscriber-only article, Adrian talks about leaving the industry altogether³.

Enjoy this issue! Please let us know if you have any feedback⁴ and get our free newsletter⁵ to stay updated about new releases. If you want to support us, subscribe⁶ for a month or a year, and let us know if you would like to write with us⁷.

Cover photo by Jon Tyson⁸ on Unsplash⁹.

REFERENCES

- ¹ <https://deprogrammaticaipsum.com/at-least/>
- ² <https://deprogrammaticaipsum.com/enough-agile/>
- ³ <https://deprogrammaticaipsum.com/not-being-a-developer-after-46/>
- ⁴ <https://deprogrammaticaipsum.com/feedback/>
- ⁵ <https://deprogrammaticaipsum.com/newsletter/>
- ⁶ <https://deprogrammaticaipsum.com/subscribe/>
- ⁷ <https://deprogrammaticaipsum.com/write-with-us/>
- ⁸ https://unsplash.com/@jontyson?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
- ⁹ https://unsplash.com/?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

At Least



By Daniel Steinberg

About ten years ago there seemed to be a slew of thirty-year-olds explaining what it was to be old.

“Now that I am thirty,” they would say, “I understand ...”

I am not sure what it was they understood.

I had stopped listening and faded off into my own head, thinking, “uh oh, what if they find out I am fifty. They will never hire me.”

I shaved my beard because that is where all the gray was. I even reasoned that it was ok to stay overweight and not eat as healthy as I should because somehow that extra fat smoothed out the wrinkles and made me look younger.

When I was in my early twenties – for reference, just after C++ and Objective C were created – I wanted to be a high school math teacher.

I wanted to change the world.

I was going to change how math was taught based on all of my experience.

A friend took me aside and said, “you are right.”

“About what?” I asked.

“About pretty much everything,” he said.

“But...” I prompted.

“But, you are young. They are not going to listen to you.”

And they did not.

“When you are older,” he said, “they will. Not because you know more but because you are older.”

Now that I am older – much older – I think it is also because I know more. I think it is because I am less angry. I am not in a rush.

But mostly, I think the same things I used to think.

I spent an hour this morning on code that did not work.

I say that in the passive voice as if it were not code that I had written that did not work.

It took me an hour to put in a breakpoint at the right position to discover that this line of code was never called. Should I have known that this line of code that had always been called in the past would never be called?

Maybe.

It was a beautiful line of code that did exactly the right thing. But it was never called. So it never did anything.

It was me in my twenties.

I did not give one of those “now that I am thirty” speeches.

I loved my thirties.

I was surrounded by smart people who could not learn fast enough.

No one was bored – we just did not have enough time to learn all that there was to learn.

I was becoming a mathematician and teaching at the college level.

In summers I taught in the Upward Bound program. It was a program designed to encourage and prepare students for success in college. These were low income or the first generation in their family to go to college.

I was the math teacher and the academic head of the program told me he thought it was more important that students write as much as they could, and that I should not assign any math homework.

I argued that math was important and convinced him to let me assign math homework.

I was wrong.

When I ran my own program years later I understood the importance of writing.

I began each day with the students writing for a half hour.

When they began college they could write fluidly. The thoughts in their head organized themselves and presented themselves on the page.

There is that passive voice again.

I mean that the students could write with ease and confidence. They could write with clarity and purpose.

Writing was more important than math.

What is it you do?

Whatever it is, you write for a living.

Memos, proposals, emails, commit messages, performance reviews, user stories, code – you write for a living.

The better you write anything – the better you write everything.

You write an email and then you decide that it is better not to send it. So you delete it.

You find the code that is never called and you eliminate it. It is just noise.

What about that code that is called all the time?

I hope it is good.

I hope it has been tested and profiled. This is where working your magic can really pay off.

My friend Adrian gave a talk about turning forty¹. How old he was. How wise.

He was wise. But I would bet he was wise at twenty.

My friend Graham challenges and educates me every time I talk to him.

They have given the community a gift with this publication.

They are both so wise and they are both so young.

I recently went through hundreds of pictures of my dad for his memorial service.

There was one of him at sixty.

He did not look so very old.

It is a month before my sixtieth birthday. I think I look younger.

I am still clean shaven. Now it is out of habit and not to hide my age.

I tell everyone how old I am.

I think it is important that you know. I think it is important that you know you work with people your parents' age and that we are still learning new things while we have the experience and context to put these new things in the right box.

Sure you can make us managers, but let us write code now and then.

I love being invited to give keynotes, but I also revel in the workshops and technical talks I am asked to present.

I was recently at a conference where one the well known speakers in our field was the closing keynote. He brought up the wrong slide deck and rambled in a way that I had once found charming.

The audience grew restless and heckled him.

He mistook it for them being engaged.

I leaned over and made the person sitting next to me promise that they would tell me if I turn into that.

That speaker and me – we are so lucky. We are that code that gets called all the time.

When I am asked to talk, my talk is tested and profiled. I have put in at least an hour for every minute of my presentation. If you are recording the talk, this will likely be the only time I give it.

Do I really spend a week to prepare for a talk I will only give once? At least.

I prefer to give a talk more than once. I learn something every time I present it. But times have changed.

When I was twenty I had so much to say.

It was raw and unfiltered.

What takes me that week to write the two hundred slides in a thirty minute talk is not thinking of what to say.

It is thinking of how to say it. It is leaving out that story I really want to tell because it will confuse you about what the point of the presentation is.

I was in Scotland earlier this year having coffee with a friend.

ISSUE 012: BEING A SENIOR DEVELOPER

He asked me my plan – you know – now that I am sixty.

“I do not know,” I said, “I would like to work another ten years. I love what I do.”

“Then why stop in ten years?” he asked.

I paused. I thought a moment. Then, I smiled.

“At least,” I agreed raising my glass.

“At least,” he nodded.

Cover photo by Florian Klauer² on Unsplash³.

REFERENCES

¹ <https://akos.ma/blog/being-a-developer-after-40/>

² https://unsplash.com/@florianklauer?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

³ https://unsplash.com/?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Enough Agile



By [Graham Lee](#)

Agile won. The practices encouraged by Agile transformation coaches the world over are practiced the world over. I do not remember a software team in the last decade who did not gather around for their regular standup meeting. A team who did not drag Jira stories across a board, in standard “as a user” format. Who did not have Jenkins builds kicked off by every commit. Who did not ignore, if not actively block, attempts to explain how their software worked with long sentences and UML diagrams.

Oh No It Did Not.

Agile lost. The values and principles ignored by those coaches and consultants are ignored by the teams who learned from them. Ignored because they were not relevant: the coaches said “successful teams do this” so we do this. They did not say “successful teams believe this, therefore they choose to do this”, so we do not believe.

My Engineering Organization Is Agile

On some teams I have worked with, that standup is weekly, or fortnightly. It is only for “the team”, which is engineering and their manager. They sometimes “invite” somebody from “the business” to join. Particularly when they want to blame “the business” for not giving clear requirements.

It is an interesting phrase, “the business”, especially when used by engineers to label the group of non-engineers¹ in the organisation. It implies that everybody else is engaged in getting customers and making money, while engineering is a cost centre funded presumably through altruistic motives.

But here is the thing. Had they not heard of agile, that team would not even have fortnightly standups. They would have trickledown briefings from their manager, who would get their information from their own manager. So the team is more connected with each other and its work. Agile won.

Again

I have been on teams where yes, there is a nightly build, but it usually fails, and is never in a state where we would want to show it to a customer. The question is whether it failed *more* than last night, and whether the tasks to fix things will get done before the ship date.

Had they not heard of agile, these teams would not even have that level of insight into the quality of their software. The build expert would do the build in the “integration phase”, after feature complete but before code complete. Only then would they discover the failures. Agile won.

And Again.

I have worked on teams where the work is organised as user stories in a Jira board, and the product owner is expected to think of all the stories upfront. The stories should be in priority order by the time the engineers see them, should be in the standard format, and should have clear acceptance criteria in Given-When-Then format. Do not call it a functional specification, but that is what it is.

The “team” (by which I mean the loudest person on the team) critiques the backlog once per sprint, in a grooming session. While “grooming the backlog” may seem like an elaborate ritual in which a Tudor monarch has their turds cleaned, it is in fact...well, it is that, without the monarch. Engineers complain that stories are too big, under specified, or too risky. Highly important stories are deprioritised in favour of easy, comprehensible, deckchair-rearranging tasks that got lower values in the Arbitrary Fibonacci Sweepstakes.

Had they not heard of Agile, none of this discussion would happen. The whole backlog would be signed off at the start, then the whole team would enter the death march until either everything gets finished, or exasperated managers fire the lot of them. Agile won.

“Agile” “Project” “Management”.

This is the issue on being a senior developer, so why am I talking about agile? Agile is a set of values, principles, and practices for making software, not a project management framework.

In fact, in most situations, the agile software development values seem antithetical to the existence of projects. We do not want to scope a project, because we respond to change over following a plan. Our team will not write a statement of work, because we collaborate with customers over negotiating contracts. We will not write down the requirements, because working software is better than comprehensive documentation.

Pushing a second layer of “in fact” onto the stack, the agile manifesto does not say any of that. It says we *value* comprehensive documentation, following a plan, negotiating contracts, and processes and tools. What it says is that there is *more*

value in working software, responding to change, customer collaboration, and individuals and interactions.

Agile Software Construction.

As a senior developer, a significant part of your role is to set expectations for how the software team and the experts in the rest of the organisation interact, and indeed to take a lead on that interaction yourself. You are a role model for the junior developers, and you are the point of contact for people from other departments.

If you want your team to run its project a certain way, or not as a project at all, it is your responsibility to take the lead. You must embody the principles you wish to see, and encourage the practices that support those principles. Whether your team is real agile, trademark Agile, or fauxgile, is on you.

In Theory, The Theory Works.

Agile software development started life as all of these things: as values (probably similar to the four in the manifesto), principles (the twelve that are below the fold on the manifesto website) and practices (see XP, Crystal, FDD, DSDM). All of these things exist today, but in different concentrations.

As Alan Francis once said to me, an idea is like strawberry jam. As you spread it, your strawberries stay in big lumps and your jam goes everywhere. Agile is like a three-tiered strawberry jam: practices are easier to see and adopt than principles, which are in turn easier than values. So we have plenty of people who have adopted practices they have learned about from “agile”, and maybe see the principles, even where they do not share the values. I still argue that they are better off than they were before, and that those of us who use software are better off as a result.

Conclusion

Agile software development has nothing to do with project management. It has everything to do with a defining set of values and principles that can guide you in making software. How you get there is up to you, and the people you work with.

If this is a disappointingly damp take on agile for you, then agile has won! Given two decades, it has moved the needle on the industry so far that an idea that was once shocking and controversial is now something everybody does. Even if they do not agree how and do not know why. Now the context is different, so it is time to tell new stories that help the people who are stood here, not the people who were there in 2000.

Cover photo by Holger Link² on Unsplash³.

REFERENCES

¹ <https://www.sicpers.info/2019/08/experts-around-the-table/>

² https://unsplash.com/photos/xdGCrwu2CWM?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

³ https://unsplash.com/search/photos/sydney-opera-house?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Not Being A Developer After 46



By [Adrian Kosmaczewski](#)

Three years ago, I gave a talk named “Being a Developer After 40.” Maybe you have heard about it. It was – and still is – my absolute bestseller article of all time. It struck such a strong chord among software developers, that many offered to translate it to the most incredible languages (among which Russian, Czech, Vietnamese and Persian) and got a lot of attention and reprints all over the place. I gave that

talk subsequently in other conferences twice. You can find it on my personal blog¹ if you want to read it. Go ahead, I will wait.

I now wish I had never written it.

The reason I say this is because I simply stopped enjoying being a software developer. In the few years that passed since I turned forty – I am about to celebrate my 46th birthday as these words hit the web – I grew tired of this industry. I still love to code, I still enjoy learning new programming languages, I still fiddle with my Raspberry Pi devices at home. I still consult about software, to a large degree, although I am not really involved in software development projects anymore.

I stopped recommending young people to become software engineers or to study computer science. I do not think anymore that learning to code is such an important skill for “the workforce” in this 21st century of ours.

I grew tired of the lies and the misplaced optimism in the future. Technophiles all over the world are convinced that any problem can be solved by throwing technology at it. Experience shows that this is not true; and actually, all things considered, this attitude makes the world a worse place.

The whole “Agile” thing – about which Graham wrote extensively² in this issue – is another point of conflict. “Agile” has become a shadow of what it could have been. The ideas behind the Manifesto have been manipulated to create a new industry of consultants and teachers and coaches and managers and books.

And the common answer to the shortcomings of “Agile” (the now common and nonsensical “You are doing Agile wrong”) is an insult in the face of the whole industry. Teams all over the planet are having their “standup meetings” without purpose or direction, just for the sake of it. Losing time in retrospectives that are merely an exercise in pointlessness and hypocrisy. Some teams here and there, sometimes, somewhere, achieve some decent levels of productivity with some Agile practice (I assume XP being the one easier to gauge of all.) Somehow. We all have to endure those typical Agile (in particular Scrum) ceremonies, celebrating them as if we were part of a secret occult group, summoning the Gods Of The Agile Manifesto™®© so that we can finish all the tasks in the backlog for this sprint.

Then there is that continuous “reinvention of the wheel”; every ten to twenty years, some programming language appears and all of the wheels must be reinvented in that new language. I grew tired of it. From C++ we went to Java then to Ruby then to JavaScript then to whatever comes next, and the same ideas come along, and functional developers come back every time, loaded with hubris and disdain, to tell us how wrong we all were, even though these days you can do functional and OO programming in any language, yes, including good old PHP.

The software industry specializes in hubris and disdain. And rejection.

I had many managers throughout the years who literally told me “I do not hire women” or “I do not hire people of color.” I have witnessed with my own eyes, how unexperienced, grotesque, “opinionated” and disgustingly inappropriate white young men got jobs that belonged to knowledgeable, experienced, sane, human, empathic people from other cultures, nations, ethnic groups, or genders. I have been fired for questioning those choices. I have been made feel guilty for not being part of the “team.”

I suffered three consecutive events of burnout and depression in this industry. I have had no support from my superiors, supposedly very “committed” to the wellbeing of their employees.

And the list could go on and on. Micromanagement, particularly by non-technical leaders taking technical decisions. Lack of accessibility, security, privacy, education or basic human decency because of “business drivers” that take precedence above all logic and understanding. Open space offices. Obnoxious software platform vendors. Useless documentation. Hostile online communities. Carpal tunnel syndrome. Technology becoming obsolete by default every six months. And did I mention dwindling salaries, too?

I personally am finding ways to walk away from the software development world, slowly but surely. I started by not looking for any more software development jobs. It is incredible to watch the same recruiters call me 10 times per month, the same person, repeating to them that I do not work as a developer anymore, in writing or by phone, and to see them come back once and again. Because, oh yes,

I forgot to mention the beautiful recruiting industry and its lack of attention and its wonderful focus on bonuses at the end of the fiscal year.

But I still work in the industry, of course! One cannot shake 22 years of professional experience out of the window as simple as that. My resumé screams of software. But I decided to use my writing and speaking skills to become “Developer Relations” or “Evangelist” as they call them. This way I can use my knowledge, but for another purpose. And I speak openly about the issues in this industry, using the stages of conferences all over the world to state openly what nobody can (or want) to say.

And even better, I have decided to go back to school, and learn something completely different, and move on to other horizons. Maybe related to software and technology, or maybe not.

As you could read, this is not a hopeful article. I am tired of how this industry has wired itself. I do not think it is healthy, I do not think it is worth the effort anymore. I do think it is worthy to speak up and wake up people to some realities. I do hope this humble publication, reaching its first anniversary soon, will contribute to that.

Writing that article, “Being a Developer After 40,” was an effort to find some sense in an industry that had none. This is the reason why I removed the article from Medium, and hid it at the bottom of this magazine. This is the reason why I am writing a rebuttal of that article here.

The lyrics of “Goodbye Yellow Brick Road” come to mind as I write these words.

Cover photo by Gary Chan³ on Unsplash⁴.

REFERENCES

¹ <https://akos.ma/blog/being-a-developer-after-40/>

² <https://deprogrammaticaipsum.com/enough-agile/>

³ https://unsplash.com/@gary_at_unsplash?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

⁴ https://unsplash.com/search/photos/garbage?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText