Issue 030: Home Office

Graham Lee

March 1st, 2021

Welcome to the thirtieth issue of *De Programmatica Ipsum*, dedicated to the subject of *Home Office*. In this edition:

- Adrian explores the *conditio sine qua non* required for telecommuting to develop and thrive[1].
- Graham analyzes the economics that conspire against[2] working from home.
- In the Library section[3], Adrian talks about Gerald Weinberg's 1971 seminal book, "The Psychology of Computer Programming"[4].

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

Cover photo by Chris Montgomery[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/await-in-async-we-trust/
[2] https://deprogrammaticaipsum.com/your-place-or-mine/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/gerald-weinberg/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@cwmonty?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/work-from-home?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Await! In Async We Trust

Adrian Kosmaczewski

March 1st, 2021

In 1997, I moved back to Argentina to start a career in software engineering in a small web startup. They were in such a small office in the outskirts of Buenos Aires that my boss told me to go to the phone company, and get a new phone line installed at home. Then she gave me a US Robotics[1] 56K dial-up modem, got me a subscription to an Internet Service Provider, and with those, I worked from home for the first few months. I would only go to the office on Monday mornings for regular coordination, and on other occasions randomly, usually during team events. The company paid for both the phone line and the ISP.

I was 24 years old, and that experience marked my perception of work completely, forever. Writing ASP[2] pages in VBScript[3] using EditPlus[4], and then uploading them to a Windows

[1] https://en.wikipedia.org/wiki/USRobotics
[2] https://en.wikipedia.org/wiki/Active_Server_Pages
[3] https://en.wikipedia.org/wiki/VBScript
[4] https://editplus.com/

NT 4[5] server using WS_FTP[6], all of that was clearly something that could be done from home. No need to commute; no need to spend 80 minutes per day in the chaos of Buenos Aires' traffic, to achieve something you could do in your own place, sipping a good maté[7] while listening to Mario Pergolini[8]'s morning show on FM Rock & Pop 95.9[9].

I used a Pentium[10]-powered laptop running Windows 95[11] I brought from Switzerland, initially without an external keyboard, display, or mouse. For live coordination, there was good old e-mail (thanks to Outlook Express[12]), and at some point we started using ICQ[13] for chat. To operate on our server remotely (located in the USA), starting with Windows 2000 we used Terminal Services. I do not remember what we used to operate our older NT 4 server, maybe Citrix.

All of this was happening two years before MSN Messenger[14], six years before Skype[15], sixteen years before Slack[16] and Zoom[17]. SSH was released in 1995[18], but, of course, initially only for Unices, and we did not know about it.

By 2002 I was back in Switzerland, and the picture was much bleaker. Even though the country had affordable and quite fast ADSL[19] connectivity by then, and virtually everyone had a Nokia 3310[20] in their pockets, I did not hear of a single tech company working remote until the late 2010s.

Why was telecommuting[21] an option in 1997, and then not in nearly every other job I had until the pandemic stroke in 2020? Why did it seem like only companies such as Basecamp[22], GitLab[23], WordPress[24], Dropbox[25], or Spotify[26] could pull it off? Why is it that business people scoffed at The 4-Hour Workweek[27] as utter fiction? What about remote workers outperforming[28] office workers?

The most important factors blocking the development of telecommuting in Switzerland for the past 20 years were, without any hint of a doubt, *hierarchy and micromanagement*. To a large extent, for good or worse, most of my managers have *always* insisted on knowing all

[5] https://en.wikipedia.org/wiki/Windows_NT_4.0
[6] https://en.wikipedia.org/wiki/WS_FTP
[7] https://en.wikipedia.org/wiki/Mate_(drink)
[8] https://en.wikipedia.org/wiki/Mario_Pergolini
[9] https://fmrockandpop.com/
[10] https://en.wikipedia.org/wiki/Pentium
[11] https://en.wikipedia.org/wiki/Windows_95
[12] https://en.wikipedia.org/wiki/Outlook_Express
[13] https://en.wikipedia.org/wiki/ICQ
[14] https://en.wikipedia.org/wiki/Windows_Live_Messenger
[15] https://en.wikipedia.org/wiki/Skype
[16] https://en.wikipedia.org/wiki/Slack_(software)
[17] https://en.wikipedia.org/wiki/Zoom_Video_Communications
[18] https://www.ssh.com/about/history
[19] https://en.wikipedia.org/wiki/Asymmetric_digital_subscriber_line
[20] https://www.mobilephonehistory.co.uk/nokia/nokia_3310.php
[21] https://en.wikipedia.org/wiki/Telecommuting
[22] https://basecamp.com/books/remote
[23] https://www.forbes.com/sites/alexkonrad/2020/11/11/no-office-no-problem-software-unicorn-gitlabs-ceo-warns-youre-probably-doing-remote-work-wrong/
[24] https://scottberkun.com/yearwithoutpants/
[25] https://blog.dropbox.com/topics/company/dropbox-goes-virtual-first
[26] https://www.theverge.com/2021/2/12/22279951/spotify-remote-work-from-home-employees-choose-announcement
[27] https://en.wikipedia.org/wiki/The_4-Hour_Workweek
[28] https://www.inc.com/brian-de-haaff/3-ways-remote-workers-outperform-office-workers.html

of my tasks in excruciating detail, and I know many colleagues would agree with me on this matter.

It is important to understand that in Switzerland, social groups naturally revolve around and evolve into hierarchies. And hierarchies, based on command-and-conquer mentalities, work in synchronous mode. Orders and their execution happen synchronously, in tightly closed loops. Through a sad misinterpretation of the adjective "agile", this included Scrum best practices, such as standup meetings and retrospectives, which became prevalent and über-popular in the past 15 years.

On the other hand, proper telecommuting requires asynchronicity. And asynchronicity, in turn, requires trust.

Readers with some C# or TypeScript experience will undoubtedly have chuckled at the pun in the title[29] of this article. In these (and many other) programming languages, functions can be structured to not block the main thread, delegating some long-running tasks to a separate mechanism of execution; a thread, an actor, a coroutine, or whatever the fashion *du jour* may be. Usually, this involves a data structure called a "Promise" or similar, which represents... well, a promise for some result in the future. Promises can be fulfilled, fail with an error, or timeout after a while; even though proper timeout is a bit more complicated than expected in JavaScript[30].

But you get the core idea; in the long-running process of a manager, the delegation of tasks to employees happens through promises, trusting that they will be fulfilled at some point in the future. As Sharleen Spiteri once said, Future is Promises[31].

In the opinion of this author, lack of trust, and hence lack of true asynchronicity, is the main reason why there are so few software startup successes coming from Switzerland on the front page of Hacker News. Simply put, it is culturally impossible for many firms in Switzerland to have the level of trust required for startups to flourish. There are trusty banks, huge mechanical engineering firms, excellent fine electronics companies; but few, if any, software success stories. Things have changed quite a bit in the past 15 years, and thankfully so.

But many firms stay stubbornly stuck in trust-less environments, and even worse, MBA programs in some parts of the country still insist on "classic" hierarchical models of management, while some companies go as far as getting rid of management layers[32] altogether and still be leaders in their industries and worth billions of dollars.

Even though UN1A[33], the biggest worker union in Switzerland, keeps reminding[34] business owners of their legal responsibilities towards employees during this crisis, it is common to hear about companies forcing their employees to commute in the middle of a pandemic to perform jobs that can be safely performed from home.

Illegal and deadly, as it turns out.

As an anecdote, a decade ago, while working as an independent iOS developer in my own business, some customer asked me to commute every day to their office to write Objective-C code sitting in their noisy open space. Even though that meant an increase in my hourly rate, to cover both my travel expenses and my personal irritation at the request, they were happy to

[29] https://en.wikipedia.org/wiki/Async/await
[30] https://stackoverflow.com/a/39538518/133764
[31] https://www.youtube.com/watch?v=ZNyCsRwYJd8
[32] https://www.inc.com/samuel-wagreich/the-4-billion-company-with-no-bosses.html
[33] https://www.unia.swiss/
[34] https://www.unia.ch/fileadmin/user_upload/2020-04-Input_dt_en.pdf

oblige, knowing that they could peek over my shoulder and see what I was up to at any time. Or to invite me to some meetings they wanted me to attend. All of these things happened often enough as to generate delays in the project, a dangerous slippery slope.

But I digress.

After this pandemic situation is over, telecommuting will require more than another crisis to stay. It will demand a change of attitude from managers in the software industry. They should finally realize that if all else fails, there is a Git repository somewhere showing the work being done by the person you are suspecting of wrongdoing. You can *actually see their work*. You are paying for it, after all, but you do not need to interrupt them.

You might not understand the programming language used; you might not know all the names of the GoF design patterns at play; but, unless you are color-blind, you should be glad to see that the CI/CD pipeline shows up in green, and not in red.

Also, dear managers reading this piece: please let us all stop with that nonsense of "trust must be won". As a manager, you either have trust in your team, or you do not. If later on you discover that people abused that trust, please, let it be known to everyone involved and take any appropriate measures; but always start with trust.

Telecommuting works, and not only because this isolates developers from the foosball table and the noisy coffee machine. There is extensive[35] research[36] available[37] now. But, as it happens, it requires asynchronicity. Which, in turn, requires trust; a currency that is hard to come across in large swathes of corporate Planet Earth.

In retrospect, it is obvious my boss of 1997 really trusted me to do my job. There was no Jira[38] (it was released five years later), Bugzilla[39] was barely in the making, and no Subversion[40] (which reached 1.0 seven years later). We did not even use CVS[41] to track changes in our codebase (do not ask). But with my colleague (we were two in the technical team) we added features, fixed bugs, and collaborated so that the website was up and running at all times. She knew which features were needed, and she got them. She spent most of the time finishing deals and getting new customers. Which was, of course, crucial to get the company up and running. That company is still up and running, by the way.

Asynchronicity allows people to work at their leisure[42], concentrating on the what, not on the how.

Asynchronicity enables people to move to Barbados[43] or Helsinki[44] if they would rather not work from home, after all.

Closer to each and every one of us, asynchronicity means Zoom calls with the camera turned off[45], making it optional[46] for people to use, or not.

[35] https://www.atlassian.com/blog/teamwork/new-research-covid-19-remote-work-impact
[36] https://twitter.com/darrenmurph/status/1359523556376346625
[37] https://twitter.com/chris_herd/status/1359135080753614854
[38] https://en.wikipedia.org/wiki/Jira_(software)
[39] https://en.wikipedia.org/wiki/Bugzilla
[40] https://en.wikipedia.org/wiki/Apache_Subversion
[41] https://en.wikipedia.org/wiki/Concurrent_Versions_System
[42] https://twitter.com/imrayana/status/1361796033639051268
[43] https://www.visitbarbados.org/
[44] https://www.helsinkibusinesshub.fi/90-day-finn/
[45] https://twitter.com/dudewithsign/status/1361869434927910916
[46] https://twitter.com/Carnage4Life/status/1359913085666947076

And for those waving the Agile flag at every breath of their working life, reflect upon all the value that asynchronicity brings to the items on the left[47].

Cover photo by Bernard Hermant[48] on Unsplash[49].

[47] https://agilemanifesto.org/

[48] https://unsplash.com/@bernardhermant?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[49] https://unsplash.com/s/photos/trust?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Your Place or Mine?

Graham Lee

March 1st, 2021

It is no great secret, and no great surprise, that the latest SARS outbreak has greatly reshaped the world of work. Particularly in software engineering: where the work can be done anywhere with an internet connection so codes can be pasted from Stack Overflow, and the practitioners generally have a dislike of meetings. Your average stereotype of a software engineer would rather build the wrong thing for eight hours in a flow state, than have a 15-minute conversation in which they find out what direction they should be going in.

An aside: I actually enjoy meetings. My view of professional software engineering is one where I get to find out about people's work and the problems they have, and try to solve them, and discussions are key to this project. If what I wanted to do were to have some uninterrupted time to discover how to shovel a Haskell into a BEAM on Kubernetes so I could scalable actor lambda, then yes, I could understand why understanding what the deliverables are would get in the way.

Unfortunately the best sorts of meetings are the serendipitous ones, the ones where you bump into Becky who leads the Windows platform team who mentions a new API coming from a supplier next month, and Drew from marketing who says that at last week's trade event all the customers were asking about support for that API.

That sort of thing isn't a "home-working" meeting. It's a spontaneous chat in the break room. I've been known to hang around the smoking areas at break times in my work, not because I smoke but because I get to talk to people from across the organisation. And, damn it, I'm entitled to as many breaks as anyone else, even if I will live longer and thus get more opportunities for recreation.

Not so, the 30 or 60 minute videoconference meeting. I now have so many of these booked in my day that I tend to mute them, switch to a different virtual desktop, and carry on working:

my furrowed expression staring at the camera continuing to give the impression of being in deep thought about the current conversation over whether a second status meeting a day would help us get things done twice as fast. I'm now invited to so many meetings that even the people sending the invites out have conflicts and can't attend.

Of course, the alternative is the commute: an hour each way in the car listening to audiobooks, or on the train listening to men in suits who cough openly and spread their corona viruses throughout society as they're too important for things like tissues or nose breathing. Then a cycle ride across the city (there being no parking near the office) to work in one of those modern "open working environments", code for an employer who's too cheap to install office doors or walls. So then to sit for eight hours in a too (hot or cold; no matter, as we'll each alter the thermostat as we walk past) typing clumsily onto a £10 plastic membrane keyboard and listening to a dozen other people doing the same. And looking at them too, over the top of the too-low monitor from the previous generation that doesn't support the laptop's maximum resolution.

At home, I've got a decent mechanical keyboard (which I paid for), high-quality optical mouse (which I paid for), good monitor (which I paid for), fast internet which...you get the picture. For some reason, home working has become a license for employers to get their staff to pay for the right to have the correct equipment to do their jobs.

Not that this actually suits the rich as much as you might think. On the one hand, they don't have to pay as much for office space (you may remember 90s/00s fads such as hot desks, hotel desks, and corridor warriors: the employers already know about the cost savings and productivity increases). However, the ultra-rich are not the people who run universities, devops advocacy firms, or web agencies. The ultra-rich are the people who own the office complexes in London and Zurich. They rent 60% of the space out as modern, luxurious work spaces; 30% as chic eateries and after-work bars; and the remaining 10% as artisanal bean-to-cup coffee experiences.

These people need bums on office chairs, because those bums pop out to Pret for a sandwich at lunchtime, Costa for a latte in their edgy walking meeting, and Wetherspoons for a pint before picking their car up from the valet and filling up with petrol before heading out to the motorway. Every one of those businesses is on their property, every one of them is paying ground rent, and they need you to go back to work so that ecosystem continues to turn a profit.

There may well be concessions—maybe the dress code will be relaxed to include pyjamas and slipper socks—but we'll all be shepherded back to the office as soon as it's reasonable to do.

Cover photo by kate.sade[1] on Unsplash[2].

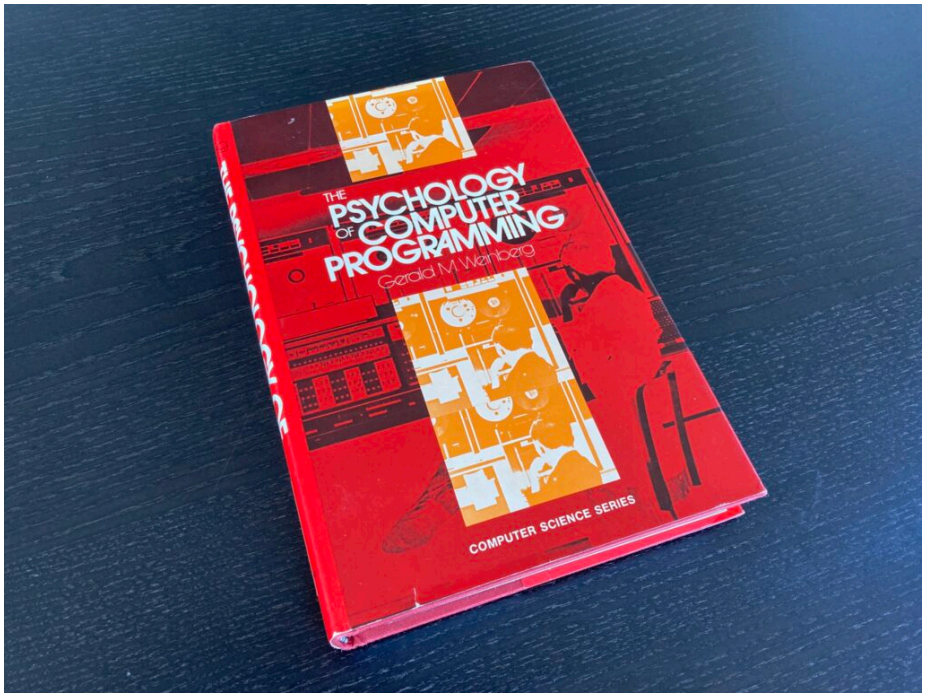
[1]https://unsplash.com/@kate_sade?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[2]https://unsplash.com/?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Gerald Weinberg

Adrian Kosmaczewski

March 1st, 2021



Some books are like mirrors. By that I mean that reading them involves a great deal of looking at oneself, both for praise and loathing. Taking a look back in time, reflecting on all those times we thought we were right and we were wrong, bringing back memories of times long gone, some of them painful, most hopefully fun and joyful.

"The Psychology of Computer Programming" is one of those. In every one of its chapters, Dr. Weinberg reflects on the human aspects of programming. It is, most probably, the first book ever written on the subject of the personality, the interactions and the characteristics of software developers. Let me be clear: this book has been written in 1971, and it has been continuously in print until at least the end of the twentieth century, for almost 30 years. This is the stuff of classics.

There has even been a reviewed "anniversary" edition published in 1999, but I bought a copy of the first edition of the book, and besides the delightful look and feel of the pages, representatives of the typography and design of the seventies, the structure, the flow and the discussion of this book make it stand in a class of its own, and represent a hallmark in our field.

Dr. Weinberg analyzes programming from both individual and collective points of view, including the issues of education, human resources management, and even programming language design. The author actually devotes a section of his book to explain how the design of a programming language impacts the readability and the subsequent maintainability (or lack thereof) of the programs written in it.

Sounds familiar?

Regarding team interactions, the author highlights the issues brought from scaling up programming teams: do small teams face the same issues as larger ones? How do communication patterns emerge and evolve? The author takes pleasure in debunking common myths and misconceptions, some of them still held today by managers all over the world: what are the factors that can cause a project to break down in pieces? The answers will surprise you, and you will wonder why nobody had ever told you to read this book first.

Dr. Weinberg also pays close attention to the individual characteristics of programmers. What defines a good programmer? Why do they take pride in their jobs? What kind of incentives should a company offer to their developers? Nice offices or challenging problems? (I think you can guess the answer to that last question.)

In the humble opinion of the author of these lines, software is primarily a social process, and only later a technical feat. Software is no more a technical product than a book is just a printed object. Software is the result of interactions among people, and as such it will reflect all the contradictions, the failures, the wonders and the joy of the people involved in it. Every single piece of software ever written by humans reflects the underlying moods, psyche and interactions of a group of people. As such, studying the psychology of a programmer yields naturally in a process in which, invariably, the software will be better at the end.

I once saw a joke on Twitter, saying that managing programmers was subject to Heisenberg's Principle, in that observing programmers changes their behavior. I do not think it is a joke, for I believe that all social systems are, actually, quantum-like systems subject to this principle; the actions of the observer will invariably alter the behavior of the observed. And that is OK, as far as I am concerned. If managers are conscious of this fact, and if they can use this to their advantage, they will be able to build sustainable teams.

Maybe Dr. Weinberg took some inspiration from Melvin Conway, who in 1967 stated[1] that "organizations design systems mirroring their own communication structures". Now you start to understand why your microservice architecture is a mess, and no, neither Istio nor Prometheus is going to help you with that.

Finally, regarding recruiting: we all know how hard it is to find and recruit software developers, yet I am appalled to see how many companies do as much as they can to destroy the teams they spent so much time and money to build. Why is this? I can only recommend all human resources managers, all project managers, and all developers as well, to get a copy of this book and, as I said at the beginning of this chapter, to take a good look at ourselves in the mirror. We all need a little bit of introspection, and Dr. Weinberg can help.

Legacy buzzword warning: this book is mostly accessible to general audiences, but there are a few sections where the author assumes a certain experience with programming languages, compilers, hardware design or even project management; and in some cases, with the 1971 versions of those.

Cover photo by the author.

[1] https://en.wikipedia.org/wiki/Conway%27s_law