Issue 051: Freelancing

Adrian Kosmaczewski

December 5th, 2022

Welcome to the fifty-first issue of *De Programmatica Ipsum*, dedicated to *Freelancing*.

In this edition:

- We learn the principles of running a healthy business[1] as independent software engineers.
- In the Library section[2], we review the software economics knowledge of Barry Boehm[3].
- In our Vidéothèque section[4], we learn from Mike Monteiro[5] how to ask nicely for payments.

We opened an account on Mastodon last month: follow us at @deprogrammaticaipsum@mas.to[6] to be notified of new releases!

We would also like to thank our patrons who generously contribute every month (or have contributed in the past) to our work and help us run this magazine. Thank you so much! In alphabetical order: Adam Guest, Adrian Tineo Cabello, Benjamin Sheldon, Christopher

[1] https://deprogrammaticaipsum.com/when-you-cant-create-you-can-work/
[2] https://deprogrammaticaipsum.com/category/library/
[3] https://deprogrammaticaipsum.com/barry-boehm/
[4] https://deprogrammaticaipsum.com/category/videotheque/
[5] https://deprogrammaticaipsum.com/mike-monteiro/
[6] https://mas.to/@deprogrammaticaipsum

Nascone, Jean-Paul de Vooght, Patryk Matuszewski, Paul Hudson, Roger Turner, and Szymon Licau.

Enjoy this issue! Please subscribe to our free newsletter[7] to stay updated about new releases, share the articles on social media, or contribute[8] if you would like to support our work.

Cover photo by Toa Heftiba[9] on Unsplash[10].

[7] https://deprogrammaticaipsum.com/newsletter/

[8] https://deprogrammaticaipsum.com/contribute/

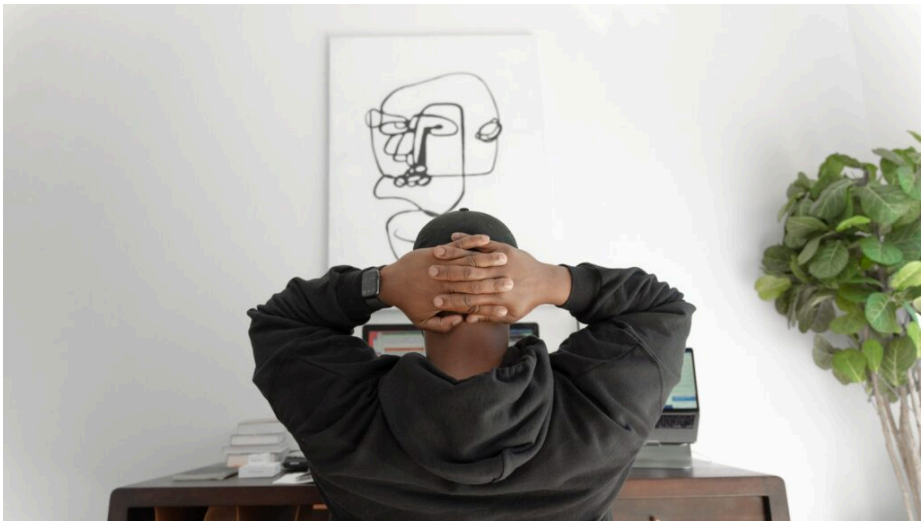
[9] https://unsplash.com/@heftiba?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

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

“When You Can’t Create, You Can Work”

Adrian Kosmaczewski

December 5th, 2022



It is hard to make a living in the software industry without crossing the path of a software developer dreaming of becoming independent. Imagine the bliss: no more bosses, no more timesheets, just you and your favorite programming language, day in, day out. Let us be honest: we all dream of building the SaaS or the mobile app of our dreams and living out of its monthly recurring income. Or, in the worst case, at least to have a nice consulting gig that pays for a full year of salary in just six months.

Let us talk about the latter case for a moment, which is, by all standards, the lowest-hanging apple of all freelancing scenarios. To start as a consultant, you need experience, a laptop, a customer, and a cup of coffee.

Such a prospect becomes ever more attractive at a time when the worldwide tech job market went from being at a standstill[1] to actively wiping out[2] tens of thousands of jobs. The situation reminds us of the post-dot-com crash[3] seen twenty years ago.

Plenty of blog posts, videos, and self-help books instruct people on achieving financial freedom by starting a consulting business. The usual narrative is similar in all of them. This article will not be wildly innovative in this respect, but this author can provide some ideas from his own experience.

The starting point mentioned above (experience, laptop, customer, and coffee) is just that: a starting point. In this text, I will give you the required information to become a professional independent software developer. Spoiler alert: your coding skills are necessary, but they are not the most important thing.

[1] https://newsletter.pragmaticengineer.com/p/the-scoop-big-tech-freezing-up
[2] https://www.trueup.io/layoffs
[3] https://www.cnet.com/culture/will-code-for-food/

Market

Picking your market carefully by specializing[4] to the maximum extent possible should be the first task to accomplish. It is also usually the simplest; to a more significant or lesser degree, we all are specialized in something when working as software developers.

The critical thing to realize is that we can change our specialization later. Many developers, particularly younger ones, might feel trapped by a particular decision. It does not have to be like that; you can migrate[5] to other technologies. You most probably will. Just give it time. If you watch for trends, you will realize your skills will become a commodity[6] in just a few years. It is unavoidable[7].

Learn economics; sadly, a subject seldom taught in computer science and engineering university degrees. It would be best if you understood how markets work because market laws will define your hourly rate. If your product (your skills and your brain) has more demand than offer (that is, people doing the same as you,) your hourly rate will be higher. It is quite as simple as that.

To make a profit, reverse-engineer the market: pick a vertical with lots of demand and very little offer. Spoiler alert 2: it might not be your favorite technology. Sorry about that.

You can work against the market mechanics in various ways: yes, you can earn a very comfy living even in a market where the rates are on the floor. The first strategy is differentiation; learn how to make people distinguish you from everybody else in the crowd. It takes time and patience, but you can learn the skill. Specialization, conference talks, and becoming an expert in social media are some ways to achieve this. You can also test the market by slightly raising your rates when negotiating with new customers. Doing it will also give you information about your market; keep doing it until the rate of refusals is too big, and boom, you have your market price. And if your rate is not enough to make a living, move to other technical "galaxies" as needed so that you do not have to drop your hourly rate too much.

The big question is the eternal one: how much should you charge? The rule of thumb is that you will spend 20% of your time coding and 80% doing business, so you should rate your time covering your costs and also making a profit at the end of the year. You have to earn per year at least the same as in a paid job in your location; otherwise, you are effectively losing money by being a freelance (Economics 101!) Remember, you will have months without income, and you should use that time to build your business.

And when should your customers pay you? For smaller projects (four to low-five figures in a currency like the US Dollar or the Euro,) there are two payments: 50% upfront and the other half right after the final delivery (invoice to 30 days; 90 days is for big businesses, you have to pay rent.) For medium to large projects (that is, medium-five to six figures,) consider splitting the final invoice in three, asking for one-third at the start, one-third in the middle, and the rest upon delivery. Also, do as Mike Monteiro says[8]: keep all rights to IP (intellectual property) as 100% yours until complete payment. Your customer must not have the right to use your work until you have been fully paid. This clause must be a part of your contract.

[4] https://deprogrammaticaipsum.com/specialization-is-for-insects/
[5] https://akos.ma/blog/migrating-from-macos-to-linux/
[6] https://en.wikipedia.org/wiki/Commodity
[7] https://www.youtube.com/watch?v=WU2XajpRdgM
[8] https://deprogrammaticaipsum.com/mike-monteiro/

Contracts

Speaking about contracts, hire a lawyer[9] and get a standard contract for your business. Such a request is not expensive because many lawyers already have templates ready for this. It would be best if you did this the same day you legally set up your business.

Since you are on a hiring spree, your second hire should be an accountant and ask them about your local regulations. Make sure to keep all tickets and invoices; you might be able to deduct lots of stuff from your yearly tax declaration depending on your location. Not having to deal with taxes will free you a lot of time, so do not make economies in this regard.

Make every customer sign a contract before you do any work. Also, do not sign NDAs. Most customers approaching freelancers instead of more prominent agencies do not have big budgets anyway, and ideas are always less original than they would like to think.

Contracts and plans take me to the following fundamental point: exit clauses. Your contracts should specify how a relationship can become sour and what to do in those cases. A good business lawyer will undoubtedly include such clauses in your draft, but make sure they are there.

One of the typical exit clauses, the simplest of them all, concerns scope changes. Any scope change in the initially agreed project makes the current contract void and demands a new one. It is as simple as that. Your business hangs on a thread because you are selling your brain CPU time by the hour, and most companies prefer fixed-time fixed-cost projects for planning purposes. Scope changes can break your business, which is why "Agile" project management is often the wrong choice for you. A healthy rule of thumb is that the smaller the project, the more "Waterfall-ish" its management should be.

Another common case for an exit clause: do not accept your customer adding third-party (or internal) developers to "help" in the project without your consent, especially after the project has started. Remember Brooks' law[10]. Collaboration is complicated in small projects; you work on a fixed-time and fixed-budget basis, and you are always one project away from bankruptcy. You cannot afford to lose time dealing with people you do not know.

Firing customers is a healthy activity. If you do not get along with your customer, consider canceling the project and moving on. Refrain from making decisions based on sunken costs: this is another of the things you should learn in business. The money, time, and effort you invested are gone; you should care more about the cash[11] you could lose in searching for a worthless pursuit than what you already lost.

Also, if you can, avoid intermediates (that is, companies hiring you to work on a project for a customer of theirs.) Your margin is thin enough already.

Customers

Keep a CRM database. Resist the urge to build one yourself; you have better things to do. Pay a monthly SaaS subscription and keep track of every email received from a customer, phone call, and opportunity that appears on the horizon.

Customers will come to you more because of your reputation than your open-source code. And your reputation is everything. *Always be on time, underpromise and overdeliver.* This

[9] https://deprogrammaticaipsum.com/mike-monteiro/
[10] https://en.wikipedia.org/wiki/Brooks%27s_law
[11] https://deprogrammaticaipsum.com/eric-sink/

simple rule will make you stand out from the crowd of the previous paragraphs, and your customers will start talking well raving about you. Others will hire you because of those positive words. No matter how many CRM systems you use, you cannot correctly measure these effects. But they happen, and when they do, it is terrific.

Regarding neurodiversity, becoming a freelance might work slightly better for extroverts: going out, meeting people, and shaking hands is part of the deal. You do not need business cards; you can exchange LinkedIn profiles from your smartphones.

You will have collaboration opportunities with other companies or professionals in your field at some point. To create a complicated software system, write a book, run a marketing campaign, or be a part of a consortium of companies. The golden rule for such collaborations stands in just one word: contract. Draft an agreement, review it, and sign it if you agree. Such arrangements are particularly critical when you deal with friends. The hard truth is that there are no more friends when money is involved.

Management

Forget about "Agile"[12] unless your customer temporarily hires you to work within an established team. If you are the sole contributor for a piece of software, you can (and you must) estimate your work correctly and provide a more "Waterfall-ish" project plan to your customer. Do not expect your customer to have a "standup meeting" every day with you; they are hiring you to solve issues here and now. Get as much information as you need before the project starts, write a detailed project plan, and make the project plan (including its deadlines) a part of the contract, signed and acknowledged by your customer.

Keep track of basic statistics of your past projects: programming language, lines of code, the time required from start to finish, and defect rates. Use those data to estimate future projects. You do not need machine learning models for this, just a basic knowledge of linear regression or COCOMO[13] will suffice, and your estimations will become so good after a while that you will barely believe it.

Remember the insurance trinity: health, accident, and legal. Get the three of them. Many insurance companies provide business packages, adding property damage (very useful if you have an office) and third-party liability.

Invest in suitable hardware: a standing desk, a decent chair, two screens, a laptop with a docking station, a good webcam, a headphone-mic headset, your favorite keyboard-mouse combo, and a fast internet connection. If you enjoy working with music, a nice set of stereo speakers and a subscription to your favorite streaming service will do wonders for your productivity. Use an office with a closing door. Only work 6 hours per day, max. Close the door after that.

Beware of burnouts[14]. Multitasking will become second nature, but you need rest and vacation days. Plan your projects accordingly and give yourself the required time off work. Enjoy what you are doing. Be happy here and now.

Code

Make your work transparent using a simple project management tool with an integrated Git repository, issue tracker, project manager, and a wiki. GitLab, Redmine, Gitea, GitHub, the

[12] https://deprogrammaticaipsum.com/you-are-doing-it-wrong/
[13] https://deprogrammaticaipsum.com/barry-boehm/
[14] https://deprogrammaticaipsum.com/business-as-unusual/

choice is today wider than ever, and the costs are as lower as a literal zero. Use one that provides CI/CD features to deliver ready-to-install software daily to your customer. Enabling your customers to self-service will make you appear more professional and give you fewer headaches. Make them open tickets instead of sending emails or (God forbid) calling you on the phone. Async[15] is the keyword, and not only for your C# or TypeScript code.

Do not think your code is the only deliverable, as essential as it is. Provide your customers with documentation about and around that code: API documentation, usage samples, build instructions, architecture diagrams, pipeline configurations, unit tests, helm charts, and whatever else is needed to understand, build, deploy, secure, and evolve the product in the future without you being around. Bundle all this information into a "software maintenance manual" and make it part of your final delivery. Consider yourself only a temporary step in a more vast process, and make yourself non-essential. Become transparent. Your customer will thank you for that and, ironically enough, will hire you again because of that attitude.

You should provide a warranty for your work; around 30 days is a healthy time, depending on the scale of the work you provide, but only for correcting bugs. As explained a few paragraphs ago, your work ends when your code runs in production with the requested features. Customers should have 30 days to use the system and report any issues, but they must only be able to request new features with a new ad hoc contract.

Refrain from committing too much time to open-source software. Remember that your focus must be generating cash; we do not need yet another web framework. You are a business now. Giving code away might or might not provide a return on investment, but the statistics show that the chances of this happening are maddeningly small. There is also this urban myth that customers will come to you because of your open-source code. That is simply a myth.

Conclusion

The most important realization of any freelance software developer is the sheer amount of non-coding work involved. You will be surprised, too. Building your own business will give you a different and very healthy perspective on your work and the dynamics of our industry. You will understand why software engineers worldwide must unionize[16] and actively support open-source projects[17] with either code or cash.

Henry Miller[18] once said[19], "When you can't create, you can work." He meant it for writers, but it works for software developers, too; when you cannot create new apps or software or fine-tune your algorithms, you can document, test, clean your desk, and get new customers for your business.

PS: a few more resources to read before you start your own business: The Beginner's Guide to Freelancing in 2022[20] by Ali Luke; How To Be A Rockstar Freelancer[21] by Collis and Cyan Ta'eed; The Positioning Manual for Indie Consultants[22] by Philip Morgan; Getting

[15] https://akos.ma/blog/the-great-idea-of-async-work/
[16] https://deprogrammaticaipsum.com/divide-et-impera/
[17] https://deprogrammaticaipsum.com/the-conquest-of-code/
[18] https://en.wikipedia.org/wiki/Henry_Miller
[19] https://www.themarginalian.org/2012/02/22/henry-miller-on-writing/
[20] https://smartblogger.com/freelancing/
[21] https://www.amazon.com/Rockstar-Freelancer-Collis-Taeed-Cyan/dp/B0025UN05U
[22] https://philipmorganconsulting.com/positioning-manual/

Real[23] by Basecamp; Manage Your Project Portfolio[24] by Johanna Rothman; Blogging to Drive Business[25] by Eric Butow; The Personal MBA[26] by Josh Kaufman; and last but not least Steven Silbiger's The Ten-Day MBA[27].

Cover photo by Keenan Beasley[28] on Unsplash[29].

[23]https://basecamp.com/gettingreal

[24]https://www.jrothman.com/books/manage-your-project-portfolio-increase-your-capacity-and-finish-more-projects/

[25]https://www.amazon.com/Blogging-Drive-Business-Maintain-Connections/dp/0789749947/

[26]https://personalmba.com/

[27]https://www.harpercollins.com/products/the-ten-day-mba-4th-ed-steven-a-silbiger?variant=32206243987490

[28]https://unsplash.com/ja/@keenanbeasley?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[29]https://unsplash.com/s/photos/freelance-coder?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Mike Monteiro

Adrian Kosmaczewski

December 5th, 2022

Erika Hall[1] and Mike Monteiro founded Mule Design[2], a design consultancy in San Francisco, around 20 years ago. Mike was (in)famously known around a decade ago on Twitter, where his profanity-laden rants about design, ethics (or lack thereof), unionization, and open condemnation of fascism, reached peaks of popularity and retweets.

Kids: those were different times in social network history.

During those golden ages, Mike gave a legendary talk called "F*ck You, Pay Me,"[3] with a title inspired by a quote[4] from the recently deceased Ray Liotta[5] in the 1990 movie "Goodfellas."[6] The talk was part of the "Creative Mornings" series of meetups held on Friday mornings in San Francisco, a series that had many spinoffs around the world.

What is interesting about this talk is that, despite being primarily directed to designers, its contents are immediately suitable for freelancing software engineers, a profession whose work is, at least since the return of Steve Jobs to Apple in 1997, inextricably linked to that of designers.

The video starts with one of the most excellent title screens ever made for a meetup (an idea this author plans to steal at some point in the future shamelessly.)

Then Mike asks the audience about everyday situations where designers end up in disarray: customers not paying in time, projects canceled, and changes of scope. At some point, he introduces his lawyer (who hid secretly as part of the audience until that point). He explains how a designer, particularly a freelance or a small agency, can get screwed in business negotiations.

[1] https://designinfluences.com/erika-hall/
[2] https://www.muledesign.com/
[3] https://www.youtube.com/watch?v=jVkLVRt6c1U
[4] https://www.youtube.com/watch?v=3XGAmPRxV48
[5] https://en.wikipedia.org/wiki/Ray_Liotta
[6] https://www.imdb.com/title/tt0099685/

The critical element of the talk is the word *contract*, a concept foreign to many independent software developers out there in the market, who, for some reason (most probably urgency, hunger, or ignorance), would not make their customers sign one before starting to work. In particular, it does not matter if you operate in the United States or another country. As an independent software professional, you (and your lawyer) must assume bad faith[7] from the other side of a commercial transaction. At least to a certain degree, of course.

As Mike says[8],

> You're at the point where you need a lawyer when you've decided to stop being a design amateur and become a design professional.

Also, having employed lawyers in a country as expensive (and as business-safe) as Switzerland, this author can safely assert that they and the contracts they crafted were worth the money. When running a business, keeping the lights on and meeting payroll is tantamount. As an independent software professional, you are a business with just one person on your payroll: yourself. Protect your interests accordingly.

Replace the word "design" with "code" throughout the video, and think.

After watching this month's video, "F*ck You, Pay Me,"[9] read more from Mike: his books[10], his stance against fascism[11], his guide about unionizing tech[12], a list of reasons why it is our fault[13] if customers do not understand what we are saying, and his explanation of why design is political[14].

What he says, and how he says it, is essential, simply because code is political too.

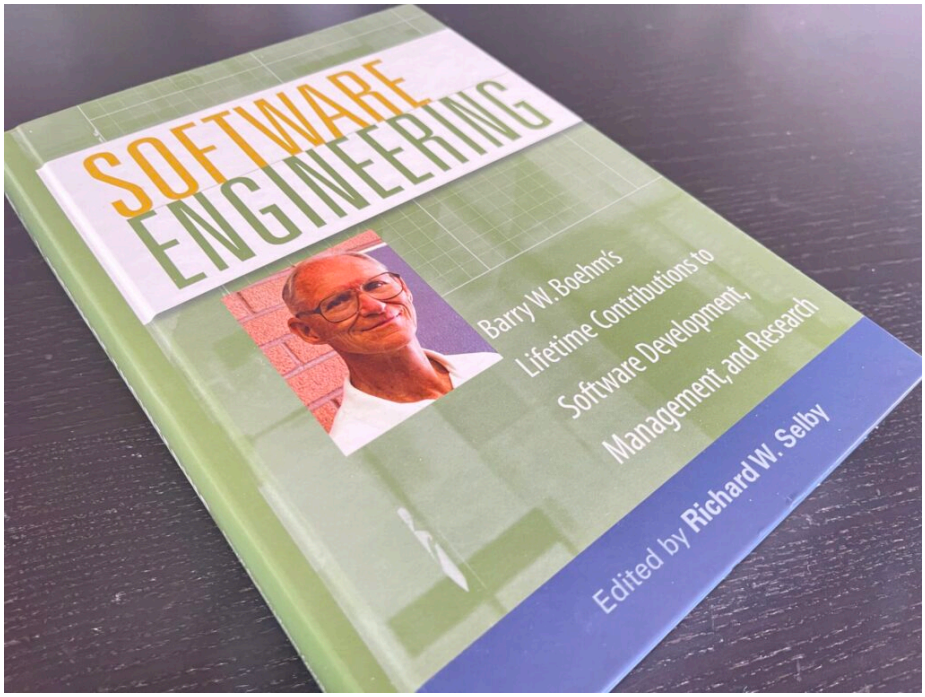
Cover snapshot by the author.

[7] https://youtu.be/jVkLVRt6c1U?t=845
[8] https://youtu.be/jVkLVRt6c1U?t=1043
[9] https://www.youtube.com/watch?v=jVkLVRt6c1U
[10] https://abookapart.com/blogs/press/get-to-know-mike-monteiro
[11] https://www.uxyall.org/presenters-collection/mike-monteiro
[12] https://modus.medium.com/dear-designer-the-union-organizers-guide-to-tech-people-2c759025970f
[13] https://www.typotalks.com/news/2012/04/07/mike-monteiro-what-clients-dont-know-and-why-its-your-fault-3/
[14] https://www.invisionapp.com/inside-design/ruined-by-design/

Barry Boehm

Adrian Kosmaczewski

December 5th, 2022



We have often talked about software economics in this magazine. For example, when we enumerated Eric Sink's[1] perspectives on the software business, discussed platforms[2] as a paradigm for economic analysis, or talked about how Brad Cox[3] advocated for an object-oriented economy. But there is a more extraordinary author about the subject, one we mentioned a few times in this magazine and who sadly passed away last August: Barry Boehm[4].

Barry Boehm started his career in software engineering in 1955, and his experience led him to analyze the economic forces shaping software engineering in business, government, and society.

The 800-page book "Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research"[5] by Richard W. Selby, published by Wiley in 2007 together with the Computer Society, contains a compilation of 43 influential papers published between 1973 and 2006.

Barry Boehm wrote about software quality, economics, tooling, risk management, and project management in those papers. He was a pragmatist; he wanted to find valuable solutions to common problems and provide software project managers with tools to implement those solutions.

[1] https://deprogrammaticaipsum.com/eric-sink/
[2] https://deprogrammaticaipsum.com/geoffrey-g-parker-marshall-w-van-alstyne-sangeet-paul-choudary/
[3] https://deprogrammaticaipsum.com/brad-cox/
[4] https://en.wikipedia.org/wiki/Barry_Boehm
[5] https://ieeexplore.ieee.org/book/5989528

Among the various inventions of Barry Boehm, we will find the COCOMO software project estimation models he developed in the 1980s and 1990s. I mentioned them in a previous article[6]:

> In his 1981 paper titled "Software Engineering Economics"[7] he went as far as to propose a formula to evaluate the cost of software projects. You heard right; this is an algorithmic method for software cost estimation (you can rub your eyes, I will wait for you); that is, a mathematical formula that could have been implemented in Excel 35 years ago. And, lo and behold, SLOCs are at the basis of the whole equation.
>
> This rather revolutionary contraption was called the COCOMO, or COnstructive COst MOdel. An unfortunate name, distorted 5 years later by the Beach Boys[8] as the soundtrack of a Hollywood blockbuster. But quite an incredible precedent, mostly forgotten by younger developers, and followed by the release of COCOMO 2.0[9] (by the same Barry Boehm) in 1995.

Some of the readers of this magazine might remember a social media website called "Ohloh" created by the same team[10] that brought us SourceForge and that offered services to open-source software developers. One of those services was, indeed, a cost calculation widget available on every project page.

Ohloh became Open Hub[11] in the meantime. However, thanks to the Internet Archive's Wayback Machine, we can see what Ohloh looked like in 2010. Their "Codebase Cost" page[12] explained in detail that they were using COCOMO for such evaluations. For example, the PEAR framework[13], providing reusable PHP components, was estimated to have cost 17'555'092 USD in June 2010[14]. Today, the Open Hub tells us that Mozilla Firefox[15] costs half a billion dollars[16].

In an age where we are debating the "unpopular opinion"[17] of whether we should pay people to make open-source software, Barry Boehm gave us the formula to figure out how expensive software actually was, and never paid attention to it. The work done by Boehm in this area is only comparable to that of Steve McConnell[18] and his book about software estimation[19].

Barry Boehm's did not stop there. In the 2000s, he proposed Value-Based Software Engineering (VBSE) as a paradigm to drive software team efforts and cost calculations.

As explained by Kevin Sullivan in the foreword of chapter 9,

> VBSE rests on the simple but transformative idea that any expenditure of scarce

[6] https://deprogrammaticaipsum.com/the-impossible-dialogue/
[7] https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE412/Boehm1981%20COCOMO.pdf
[8] https://www.youtube.com/watch?v=fJWmbLS2_ec
[9] https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE412/Boehm1995%20COCOMO%202%20.pdf
[10] https://www.linux.com/training-tutorials/ohloh-where-users-meets-developers/
[11] https://www.openhub.net/
[12] https://web.archive.org/web/20100626002648/http://www.ohloh.net/wiki/project_codebase_cost
[13] https://pear.php.net/
[14] https://web.archive.org/web/20100621050651/http://www.ohloh.net/p/pear
[15] https://www.mozilla.org/en-US/firefox/
[16] https://www.openhub.net/p/firefox/estimated_cost
[17] https://www.youtube.com/watch?v=8mRVM3BUFPc
[18] https://deprogrammaticaipsum.com/steve-mcconnell/
[19] https://www.microsoftpressstore.com/store/software-estimation-demystifying-the-black-art-9780735635135

> and valuable resources on software or a software-intensive system or initiative should be seen and managed as an investment in a risky asset: a project leading to a product or service of uncertain value.

Sounds about right: consider software a risky asset and apply rules inspired by financial management to understand costs, outcomes, and resources.

His obituary[20] says it all. A "legend" and a "giant" are the words used to describe his work and persona.

Barry Boehm was part of a generation of academics who oversaw the rise of computing in all areas of society. He watched them evolve from large rooms onto our desks and later into our pockets, transforming every aspect of our lives. But sadly, instead of a podcast or a blog, he poured his brain into PDF papers, which most full-stack engineers have never read.

Not many remain from his generation, and whether we like it or not, we are their legacy. We should understand the work of people like Barry Boehm better.

Cover photo by the author.

[20] https://viterbischool.usc.edu/news/2022/09/barry-boehm-a-living-legend-in-systems-and-software-engineering-dies-at-87/