Issue 001: Hype

Adrian Kosmaczewski

October 1st, 2018

Welcome to the first issue of *De Programmatica Ipsum*, dedicated to the subject of *Hype*. In this edition:

- Graham introduces the concept behind this magazine in De Programmatica Ipsum – On Programming Itself[1].
- Getting into the subject of this edition, Adrian argues that Mainstream Is The New Hype[2].
- Finally, in our first subscriber-only article, Graham unearthes the substance of hype in What Is Behind The Hype?[3].

Enjoy this first issue! Please let us know if you have any feedback[4] and get our free newsletter[5] to stay updated about new releases. If you want to support us, subscribe[6] for a month or a year, and let us know if you would like to write with us.[7]

Cover photo by Jonas Jacobsson[8] on Unsplash[9].

---

[1] https://deprogrammaticaipsum.com/de-programmatica-ipsum/
[2] https://deprogrammaticaipsum.com/mainstream-is-the-new-hype/
[3] https://deprogrammaticaipsum.com/what-is-behind-the-hype/
[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/photos/0FRJ2SCuY4k?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[9] https://unsplash.com/search/photos/magazine?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

De Programmatica Ipsum – On Programming Itself

Graham Lee

October 1st, 2018

Welcome to *De Programmatica Ipsum*! Established in 2018 by software industry veterans Adrian Kosmaczewski[1] and Graham Lee[2], this monthly magazine brings analysis and opinion on the art and technology of making software from the people who make it.

Here you will not find introductory articles on the latest web app framework or seed-funded database technology. Instead, you will find discussion on why there *are* so many web app frameworks and seed-funded databases, and how to survive the fatigue of being told it's time to migrate to the newest and shiniest – again. Indeed the first issue is themed "Hype"[3] and will cover exactly that. Your editors bring decades of experience from the fields of software development, testing, systems administration, architecture, team leadership and management. They are also both accomplished presenters, authors and trainers, skilled at entertaining and informing together.

De Programmatica Ipsum explores the real value in software: the people who use it, their reasons for using it, and the people who make it and their reasons too. The authors of the Agile Manifesto[4] described the higher value of the "things on the left": individuals and interactions, working software, customer collaboration, and responding to change. These ideas have frequently been viewed through the narrowest of lenses: the individuals and interactions *on the engineering team*, for example, or the customers and collaborations *for the current project*.

[1] https://deprogrammaticaipsum.com/user/akosmatr/
[2] https://deprogrammaticaipsum.com/user/graham/
[3] https://deprogrammaticaipsum.com/issue/issue-001-hype/
[4] http://agilemanifesto.org/

In science fiction, Isaac Asimov famously formulated the Three Laws of Robotics, and wrote a series of stories collected in *I, Robot* and *The Rest of the Robots* exploring their implications, as well as novels featuring robots that obeyed the laws. In one of these novels, *Robots and Empire*, a Zeroth Law is introduced. The first three laws were about individuals (a robot must not harm a human being, nor jeopardize its own existence) and interactions (a robot must follow instructions from a human); meanwhile the Zeroth Law forced them to consider society as a whole (a robot must not harm humanity, nor allow humanity to come to harm).

Now the laws of robotics are not an appropriate model for an ethical code, but moving from thinking about the people that *I* see in the office to *everybody* that my work impacts; considering how *society* is helped or harmed by my products, my work, my conduct; treating Agile software development as the first laws from which we should derive the Zeroth Law: *that* has value.

You will find more in *de Programmatica Ipsum* than the stories of its founder-editors. We believe in hearing a diversity of perspectives and insights, and want to feature guest authors in each issue to bring a fresh perspective on the month's theme. That could be you, and in return for your contribution, we would gift you a cash honorarium and a free years' subscription. Want to get involved?[5] You will be able to read a new article here on the first Monday of each month, or you can become a Premium Subscriber[6] to support our work, and get full access to all articles, including the archive of back issues. In the meantime, feel free to subscribe to our free monthly newsletter[7] to be notified of new releases.

Once again, welcome, and on with the first issue: **Hype**[8]. Future issues will cover the workplace, ethics, burnout, revisionism, security, quality, and more. We look forward to sharing the journey with you.

Ave!

Cover photo by Daniel Alvarez Sanchez Diaz[9] on Unsplash[10].

[5] https://deprogrammaticaipsum.com/write-with-us/

[6] https://deprogrammaticaipsum.com/subscribe/

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

[8] https://deprogrammaticaipsum.com/issue/issue-001-hype/

[9] https://unsplash.com/photos/_RgeLpaIp7k?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[10] https://unsplash.com/search/photos/old-computer?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Mainstream Is The New Hype

Adrian Kosmaczewski

October 1st, 2018

Early in September, while the first drafts of this article hit the administration console of WordPress, a friend of mine invited me to attend a projection of yet another Apple keynote, beer and pizza included; the usual hipstanerd package, in the office of some app development agency in the city of Bern. Surprisingly enough (or not,) Microsoft chose to host its online .NET Conference at exactly the same day and time.

A telling story of Apple vs. Microsoft, once again, clashing in my professional life, offering me another possibility to branch the future of my professional universe with a simple decision.

Let's start with the mandatory dictionary quote (after all, this is a license I can take, given that this is the first issue of this magazine.)

> **hype** | hʌɪp | informal
>
> noun *[mass noun]* extravagant or intensive publicity or promotion: his first album hit the stores amid a storm of hype.
>
> • [count noun] a deception carried out for the sake of publicity: is his comeback a hype?
>
> verb *[with object]* promote or publicize (a product or idea) intensively, often exaggerating its benefits: an industry quick to hype its products | they were hyping up a new anti-poverty idea.
>
> ORIGIN
>
> 1920s (originally US in the sense 'short-change, cheat', or 'person who cheats etc.'): of unknown origin.

In my own personal dictionary, I cannot avoid seeing Hype as anything else than a factor of stress. Not just another element of our developer culture, but an actual trigger of anxiety, burnout, Fear Of Missing Out (or "FOMO,") and other niceties that are plaguing the minds of my colleagues.

In many ways, Hype spreads like cancer. It often starts unnoticed, spreading and multiplying in otherwise sane hosts, and by the time it is discovered it is usually too late.

Hype increases job instability, it feeds anxieties, and sustains a rather shallow cohort of trainers, consultants, businesses, online courses and whatnot, venting platitudes and marketing dogma to whoever is worried enough to lose their jobs. It is a source of antagonism, decrepitude, and trolling, all of which our craft could genuinely avoid for the greater good. How many otherwise competent workers have been fired for not being "on the cutting edge" anymore? How many dollars were spent in training teams for technologies that simply disappeared a few years later? How many rewrites have happened to take an otherwise fine working system into a new paradigm, dismantling teams, introducing bugs, and killing revenue?

And all of that, for what?

I have been victim of Hype many times during my career. Sometimes, I have to admit, I actually was one of the trolls feeding the anxiety of my colleagues with the technology choices I made. I even was one of the trainers, one of the consultants, and Hype has also paid my rent to a certain extent.

I can totally understand Hype as a phenomenon, and I can relate to its side effects. Then, who am I to judge? I fed it. It grew up on me. After all, the web has Hype in 1997. Later .NET was Hype in 2002. Also the iPhone was Hype in 2009. Docker was Hype in 2015. The four major technologies that have shaped my career, all touted as the world-changing things that they actually were. And at the time of this writing, Serverless is Hype and the cycle goes on.

As I said, I can understand Hype. But as a wise friend of mine once said, "the fact that one understands something does not imply that one has to agree with it." And Hype, I do not agree with it anymore.

For the sake of memory and history, here go more Hype-compatible terms from the past 30 years: CASE Tools, CORBA, RUP, SOAP, MDA, Software Factories, Semantic Web, OLPC, and (of course!) Augmented Reality, in both of its moments of glory, 2009 and 2017.

What is "Hype"? How can one define it? Maybe I should define it through its opposite, through its counterpart: the "Mainstream".

If Hype is the future, the Mainstream is the past.

If Hype is unstability, the Mainstream is stability.

If Hype is fun, the Mainstream is boring.

If only it was so simple. For Hype is, maybe, just another acronym, like there are so many in our industry, one meaning *Hyperbolic Yet Passing Effusion*. Of course, we all know *People Can't Memorize Computer Industry Acronyms* or PCMCIA – yet another Hype technology back in the 90s.

(The astute reader will have noticed the absence of the word "Legacy" in this text. It is no coincidence, as once again, that word is more closely related with subjective visions of evolu-

tion than actual facts. Otherwise, go tell that poor soul working in a datacenter near Zurich Paradeplatz that their COBOL banking system, running flawlessly and profitably since the 70's, is "Legacy.")

Should we avoid Hype? This author says yes, for sanity, although it is very simple to realize that for good or worse this cannot be done; we simply cannot fight against it. We will not be able to make it disappear. Can we fight against Vogue, Versace, Karl Lagerfeld and Anna Wintour all saying unanimously that, let's say, blue garments are the new "it" this season?

Hype is very often related to fashion, a concept itself related to taste, a concept subjective by definition. It is actually funny to consider that Hype is highly subjective and irrational, given the tendency in geek circles to value objectivity as the guiding rule for all decisions in the world. Yet, more than once, a technical decision is simply guided by this angel of death called Hype, flying its wings once again in a meeting room somewhere.

(The irony of the subjective/objective dichotomy does not stop there, as the heated discussions about "Programming as Art" tend to show. But this will be the subject of a future edition of this magazine.)

Maybe the problem is that Hype is devoid of substance; is it the case, though? Can one always discuss Hype in rational terms? If Hype is fashion, and thus subjective, and thus irrational, can it contain substance?

Clearly, history shows that yes, Hype can sometimes be full of substance, with more or less the same probability of being full of excrement (the editors of this magazine are keen in using the most appropriate language to avoid hurting any susceptibilities.)

What is then the magic substance that makes Hype stand the test of time? Is it *Market Share*? Is it the availability of *Conferences* and *Documentation*? Is it a healthy mix of FOSS (Free and Open Source) and Commercial projects around it? Is it a large number of answered questions in Stack Overflow with a score higher than 1400?

Can Hype become Mainstream? Of course it does. C++ was Hype in 1990. JavaScript, too, will become Mainstream one day – and the author of these lines thinks that this will not take long now.

Instead of fighting against windmills like the Quijote, here is a wish: can we stop feeding Hype and prevent it from spreading like a cancer in our production systems? Could we stop using Hype-compatible technology for our next application server, mobile app, healthcare product, and instead insist that proven, old, Mainstream, boring technologies be used instead? The author of this text hopes so, and fervently for that matter, that one day sanity will guide the CTOs of the world into the creation of the next *disruption* using a 30 year old language, and a framework recently updated to version 19.

(To be honest, this very blog encompasses that vision, using a decidedly Mainstream – slash boring slash stable – platform like WordPress, supported by the venerable quartet of PHP, MariaDB, Apache, and FreeBSD. But I hope that the mention of PHP in these lines will not alienate our readership.)

Hype *can, should and must* be kept away from the production process of critical systems. This will not always be the case. But one can always dream of a better world[1].

In the meantime, I chose to attend the .NET Conference – and I do not regret the choice. Being 18 years old, C# is mature enough now for being considered Mainstream. Even the

[1] https://akos.ma/blog/a-quest-for-a-better-world/

"classic" .NET Framework has been declared Mainstream[2] – or was it Legacy instead? Regarding my friends, well, I will have a beer with them tomorrow anyway, to celebrate the release of this very magazine.

Cover photo by Verena Yunita Yapi[3] on Unsplash[4].

[2] https://medium.com/@andy.watt83/the-net-framework-is-done-8aec3bbae12d

[3] https://unsplash.com/photos/NrtC3y108Ys?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

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

What Is Behind The Hype?

Graham Lee

October 1st, 2018

Argumentem ad novitatum[1] – appeal to novelty – is the fallacy of saying that some mindset, position, or artifact is good because it is *new*, and newer must mean better. Argumentum ad novitatum is itself named in Latin because of the equivalently fallacious form of argument, argumentum ad antiquitatem; the old ways are inherently better.

If someone on your team comes in effusing about a new UI framework that was just published by a Silicon Valley unicorn to Github or the brand-new Y Combinator funded NoSQL database that the winners of TechCrunch Disrupt used to build their hack, it can be tempting to dismiss the idea of adopting it because newer doesn't necessarily mean better. That we should choose boring technology[2]. But there's probably good intention behind the new tool, even if your colleague isn't getting that over to you.

While *you* may be hearing about this shiny thing because it is new and exciting, that is probably not the only reason for its existence. Try to look past the excitement and empathise with the people who created it. What problem did they have, and what was it about existing solutions that didn't work for them? Were they really making a new tool so that it would be newer than the existing tools, or did it do something the others did not? Now, is that something you can use?

This empathy is often absent, particularly when the community around an idea gets larger and enters the main stream. David Chapman identifies three categories of people[3] in the development of a subculture: geeks, MOPs, and sociopaths. The geeks are the people who

[1] https://en.wikipedia.org/wiki/Appeal_to_novelty
[2] http://mcfunley.com/choose-boring-technology
[3] https://meaningness.com/geeks-mops-sociopaths

love the thing, care deeply about it, understand it in great detail, and love exploring it, creating it, or talking about it.

As the geeks explore and share their interest, they attract a community of MOPs (short of Members Of Public). MOPs are less interested in the minutiae and the philosophy, but still want to be associated with the subculture and want to be *seen* to be associated with it. They adopt its imagery and lexicon, while putting less effort into creation and comprehension of the core parts of the scene. They also provide useful validation (financial or emotional) for the geeks, by legitimising the scene and boosting its numbers.

When the sociopaths turn up, they are like human cuckoos, displacing the geeks from the centre of the scene in order to reap the rewards that come from selling to the MOPs. They are people who are not fanatics about the basis of the scene, and simply see the opportunity to be had in exploiting its existence. They dilute the core principles on which the geeks founded the scene, even to the point where the geeks leave and the subculture is hollowed out.

While Chapman's article was about scenes and subcultures of the late 20th century, like punk music or the fruitarian diet, we can see parallels in some of the big movements of the software industry.

Richard Stallman and others initiated the Free Software[4] movement in the 1980s based on a political or ethical notion that the people buying and operating computers should have certain freedoms to use them for any purpose, and to study, share, and improve the software employed on those computers. Two motivating events, according to Stallman[5], were the acquisition by his department at MIT of a Digital Equipment Corporation computer and a Xerox printer. Both ran proprietary software, and both had room for improvement. But here they were, in the Artificial Intelligence lab at a prestigious university, and neither *could* be improved by the local skilled and motivated programmers because the vendors believed that they owned the software and that the people trying to use their products did not. Stallman came to realise that he disagreed, and that "ownership" of software was anathema.

Later, the term "Open Source" was created by Christine Peterson in a group deliberately seeking to take the activities involved in Free Software and make them palatable to businesses, by removing the ideological connotations attached to the Free Software term. They instead associated the idea of sharing source code with Open Systems, an economic movement promoting interoperability between decidedly proprietary software products like OpenStep and OpenVMS. So it was that Netscape Navigator (the browser that was the predecessor to Firefox) became the world's first Open Source project, after Eric S. Raymond convinced the development team to use that term.

Now, plenty of pundits will tell you that the Open Source ecosystem has never been healthier[6], yet Richard Stallman's problems remain unaddressed. I am writing this article surrounded by computers – the one I'm working at, the one in the audio system it's connected to, the ones in my phone and my wristwatch; and, yes, just like Stallman, the one controlling my printer. Despite the apparent success of open source, *not one* of those systems is fully open for me to use, study, share and improve.

To me, Free Software has the hallmarks of a late-stage subculture: the scene has been popularised, the surface ideas taken and turned into mainstream businesses. All that is left is the hype: we are open! Buy our open thing!

[4] https://fsf.org

[5] https://www.gnu.org/events/rms-nyu-2001-transcript.html

[6] https://www.techrepublic.com/article/open-source-has-never-been-bigger-yet-still-is-misunderstood/

Another example: Scrum[7] is currently such a popular anti-hero for showing how the Agile movement "went wrong" that Agile manifesto signatory Ron Jeffries complains about "Dark Scrum"[8] and is even motivated to use scare quotes on the word "Agile"[9]. Undoubtedly, the people at that Snowbird meeting two decades ago were creators and fanatics, keen to "[uncover] better ways of developing software by doing it and helping others do it." But what of the legion of today's Certified Scrum Masters and Agile consultants? Do they all have the same deep understanding? Does such an understanding even *help* now, or is the context of today too far removed from the context of Snowbird in 2001 for the same principles and practices to be relevant? Are they geeks, MOPs, or sociopaths?

Regardless of the answers to these questions, the world's companies are not done being told that they are all software companies; that software is eating the world; that they are long overdue an "agile transformation" and that if they are failing it is because they are not agiling hard enough.

You pick up the allegorical, digital equivalent of a programming industry trade journal and read that your monolithic backend server is yesterday's news. It's 20th-century technology. The new hotness is microservices – breaking each component in your server out into a separate deployable service with its own interface. This, they tell you, is Service-Oriented Architecture done *right*. Is that relevant? Are the benefits of separating the components worth the increased complexity of deployment, or the performance impact of so many API calls per external request? Only you and your team can answer those questions, but that does not mean that the authors of the article on microservices know nothing that is relevant to your situation.

Whether it's last decade's big idea or this morning's new Javascript framework, it was created by someone motivated to solve a problem they saw. Dismissing these things out of hand – the old because it is yesterday's news, the new because it is untried and immature – means missing out on fresh perspectives from which to view your work. As with many situations, application of empathy will go a long way. Empathising with the creators of the ideas or projects will help you to see what the world looks like from their position, why they solved things the way they did, and whether that is relevant to your own context. Maybe you agree that the problem they saw is real, but it isn't one that you have. If so, you have not found a tool that you can use – but you *have* learned more about your situation and somebody else's. And that understanding is where design happens.

Cover photo by Dmitry Ratushny[10] on Unsplash[11].

[7] https://www.scrum.org/

[8] https://ronjeffries.com/categories/dark-scrum/

[9] https://ronjeffries.com/articles/018-01ff/agile-manager/

[10] https://unsplash.com/photos/wpi3sDUrSEk?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[11] https://unsplash.com/search/photos/empathy?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText