The Pragmatic Bookshelf

PragPub

Keeping It Light

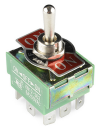
IN THIS ISSUE:

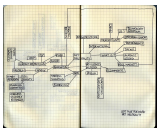
* Marcus Blankenship on onboarding
* Johanna Rothman on WIP limits
* Mike Beedle on a soapbox
* Darren Sim on Feature Toggles
* Scott Ambler on MVP
* Michael Swaine on Net Neutrality
* Plus new tech books, tech news, a sudoku/anagram puzzle, and
* John Shade on 2018

Issue #103
January 2018

Contents

FEATURES





COLUMNS

DEPARTMENTS

Published monthly in PDF, mobi, and epub formats by The Pragmatic Programmers, LLC, Dallas, TX, and Raleigh, NC. E-Mail webmaster@swaine.com. The editor is Michael Swaine (michael@pragprog.com). Visit us at http://pragprog.com for the lowdown on our books, screencasts, training, forums, and more.

ISSN: 1948-3562

On Tap

Net Neutrality

by Michael Swaine

What's in the issue? Who are these authors? Whence those images?

A co-author of the Agile Manifesto challenges the movement he helped start. A young developer embraces an old technique that is newly relevant. A manager detects a leadership smell. It's the January, 2018 *PragPub*.

Mike Beedle helped start the agile software development movement. In the kickoff essay in our new Soapbox series, he sounds off on something that's been bothering him. The Agile movement isn't agile. Shouldn't we be eating our own dog food? Shouldn't we iterate and improve agile itself? Mike thinks so, and he has some ideas about how to start.

For years, Scott Ambler has been working with organizations around the world to help them to improve their software processes and be more agile. One insight he has gleaned from this work in the trenches is that terminology matters. He demonstrates that in this issue with an article that sorts out several similar-sounding terms from software development, and works through a concrete product-development example to show how the terms differ and how using terms correctly can help keep your development efforts on track.

One of the challenges of Continuous Delivery (CD) arises when a feature spans multiple commits, and an urgent fix needs to be released before the development of that feature is complete. When a feature is completed and deployed to production, teams also need a way to experiment safely, beta testing with a small group of users and being able to roll back quickly if an undesirable experience is observed. Feature toggles to the rescue. Darren Sim explains this practice, which was common twenty years ago and is being rediscovered today.

Net neutrality was the big tech policy story last year. It might be the big tech policy story this year, too. The battle is not over, it is just moving to another venue. Your editor reviews what net neutrality is, what the arguments for and against are, what happened at the FCC last year, and what to expect this year.

What else? Cryptocurrencies, the history of online, a sudoku-slash-anagram puzzle, plus Johanna Rothman on how WIP limits can reduce backlog pressure, Marcus Blankenship on why your company may be doing onboarding wrong, Antonio Cangiano on all the new tech books, and John Shade on his predictions for 2018.

We hope you enjoy it!

Who Did What

writing, editing: Michael Swaine mike@swaine.com • editing, markup: Nancy Groth nancy@swaine.com • customer support, subscriptions: mike@swaine.com

• submissions: Michael Swaine mike@swaine.com • extreme curation: BoB Crew • reclusive curmudgeon: John Shade john.shade@swaine.com

Photo credits: Cover: "Work, Workaholic, Writer, Programmer, One, Laptop" [U1] by Comfreak is licensed under Creative Commons 2.0. Page 13: "Dare to Eat a Peach" [U2] by Chubby Soapbox is licensed under Creative Commons 2.0. Page 17: "Toggle Switch — 3PDT" [U3] by SparkFun Electronics is licensed under Creative Commons 2.0. Page 23: "MVP" [U4] by Craig Murphy is licensed under Creative Commons 2.0. Page 28: "Net Neutrality for Left Foot Forward" [U5] by Ged Carroll is licensed under Creative Commons 2.0.

You can download this issue at any time and as often as you like in any or all of our three formats: pdf [U6], mobi [U7], or epub [U8].

Swaine's World

Cryptocurrencies

by Michael Swaine

Cryptocurrencies, the history of online, and other stuff Mike found interesting this month. Also a puzzle.

THE HISTORY OF ONLINE. In support of this month's feature article on net neutrality, I thought I'd pass along some sources on the early days of online access.

In a Facebook group [U1] I belong to called The Real History of Online, Mike Banks shared news about his online history book:

"I'm working an all-new and expanded edition of *On Way to the Web*. It covers the genesis of the Internet, ALL the commercial dialup services (business and consumer), ARPANET, packet switching and PSNs, and everything that led up to all this fun stuff, from the 1940s on. I'll be posting the current book's table of contents, and then some illustrations that I did not use, and a bunch more. I encourage corrections, additional info, and discussion.

Here is a link to the current edition of Mike's book, plus other sources on the early days of online access:

- On the Way to the Web: The Secret History of the Internet and Its Founders 1st ed. [U2] — Michael Banks
- Where Wizards Stay Up Late: The Origins Of The Internet [U3] — Katie Hafner
- Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web [U4] — Tim Berners-Lee
- Inventing the Internet [U5] — Janet Abbate
- A History of the Internet and the Digital Future [U6] — Johnny Ryan
- The Global War for Internet Governance [U7] — Laura DeNardis
- The Internet: An Ethnographic Approach [U8] — Daniel Miller and Don Slater
- Tubes: A Journey to the Center of the Internet [U9] — Andrew Blum
- Networks and States: The Global Politics of Internet Governance (Information Revolution and Global Politics) [U10] — Milton L. Mueller
- How the Web was Born: The Story of the World Wide Web [U11] — James Gillies and Robert Cailliau

- *It is the period between Christmas and New year. No one knows what day it is. Time doesn't really exist. Can we start drinking at 10am? Why not. Existence is a confusion.* — @TechnicallyRon

PROGRESSIVE WEB APPS. In Antonio's tech books department in this issue, he talks about progressive web apps. A Progressive Web App is a web

app that uses modern web capabilities to deliver an app-like experience to users. They are (I didn't know, so I visited this site [U12]: progressive (work on any device and enhance progressively), discoverable (in search engines), linkable (able to retain or reload state), responsive (fit the device's form factor and screen size), app-like (look like a native app and be built on the same model), connectivity-independent (work offline), re-engageable (e.g., through push notifications), installable (on the device's home screen), fresh (in terms of content), and safe (hosted over HTTPS). Google really wants you to adopt them [U13]. Microsoft is on board [U14]. Also Apple [U15], apparently, though after some foot-dragging.

- *My parents got an amazon echo for christmas & all they do is shout at it & get disappointed by all it can't do. I've been replaced by alexa & it's great.* — @atragedyoflove

CRYPTOCURRENCIES: GOOD OR EVIL? The latest issue of *Scientific American* tries to sort out blockchain, cryptocurrencies, and the future of trust. Despite the fact that, according to one author, there is no use case for the blockchain technology [U16]. I think the authors of the three *Scientific American* articles would disagree. I recommend the issue, but here are a few points that stand out.

Cryptocurrencies are a truly disruptive technology. That disruption could be good or really bad, depending on implementation. So we should probably educate ourselves about the possibilities in self defense.

The Bitcoin model for cryptocurrency isn't going to scale to the needs of a global financial system. A blockchain-based cryptocurrency could bring increased transparency to the system and reduce the risks inherent in the current system, risks that lead to events like the 2008 financial crisis. But new models will be required. Other models that do already exist, and are being developed and tested and refined.

It's happening. The World Economic Forum predicts that ten percent of the world's GDP will be stored in blockchain-based technology by 2025. And hardly anybody knows what's going on. Three in five people in a 2017 HSBC survey have never heard of the technology and four in five of the rest don't know what it is. Two in five senior executives in major US companies admit that they don't know what it is. And there's widespread belief that whatever it is, it's probably illegal.

We may decide that cryptocurrencies is a poor name for blockchain-based technologies. Because identity and trust are what we should be thinking about as new applications of the technology begin to emerge. The concept of money may cease to have value.

- *If you cannot explain something in simple terms, you don't understand it. —Richard Feynman* — @ValaAfshar

TROY HUNT REVERSES. We published an article recently by Troy Hunt in which he defended Apple's Face ID on grounds that we found pretty convincing. He did admit, though, that he was defending the *concept* of Face ID. He hadn't yet tried it. Now he has, and he says Face ID stinks [U17]. He still stands by his article, which was about the security aspects of Face ID. "In all measurable ways," he says, "the security posture is as good as (or better than)

Touch ID." What he finds stinky is the fact that his phone doesn't always recognize him when he's wearing sunglasses.

WILL APPLE REVERSE? (This is not about the battery-life-slash-secret-slowdown kerfuffle.) Former Apple executive and always interesting gentleman Jean-Louis Gassée reminds us[U18] that Steve Jobs could famously tell us one week that some technology was braindead and that Apple would never embrace it, and the next week reveal that the same technology is actually the greatest thing ever. So we shouldn't be surprised if the current Apple continues this tradition.

So despite the fact that Tim Cook said, "You can converge a toaster and a refrigerator but you know, those things are probably not going to be pleasing to the user," we shouldn't assume we have any reason to believe that Apple won't come out with a toaster-fridge — or not that exactly, but a tablet-PC blend, which was what Cook was really talking about. But what would it look like? Gassée has some thoughts. Not a Mac with tablet features. An iPad with more and more computer features, yes. He's willing to predict that Apple will reverse itself on this hybridization thing, but it will happen on the tablet, not the Mac.

- *"No one in the brief history of computing has ever written a piece of perfect software. It's unlikely that you'll be the first." - Andy Hunt* — @CodeWisdom

The PragPub Puzzle

IT'S PUZZLE TIME! Here's this month's Sudoku/Anagram brain-teaser.

		S						I
	L							
E	T	N	A		G	L		
							N	
T					A		E	
	A			L	R			
A				R				
L				N	E	T		
	I					S		N

It's a Sudoku, yes, but with letters. Just use the nine different letters in the grid to complete it so that every row, column, and small square contains each of the nine letters.

Once you solve that, you can rearrange the nine letters to form a description of the most difficult problem, something encountered in calculus courses, and an obsession of Pythagoras. Solution further on in the issue; don't peek!

- *It's all harmless masonry until the final brick is laid and the Pink Floyd song turns into the Edgar Allen Poe story.* — @pragpub

FOLKS I FOLLOWED THIS MONTH: Vala Afshar, Michael Banks, Antonio Cangiano, Mike Elgan, Jean-Louis Gassée, Troy Hunt, Programming Wisdom, holy roses, *Scientific American*, and TechnicallyRon. You can follow me at www.twitter.com/pragpub [U19] or swaine.com [U20].

External resources referenced in this article:

[U1] https://www.facebook.com/groups/1081760928537189/
[U2] https://www.amazon.com/Way-Web-History-Internet-Founders/dp/1430208694
[U3] https://www.amazon.com/exec/obidos/ASIN/0684832674/cybergeographyre
[U4] https://www.amazon.com/Weaving-Web-Original-Ultimate-Destiny/dp/006251587X
[U5] https://www.amazon.com/Inventing-Internet-Inside-Technology-Abbate/dp/0262511150
[U6] https://www.amazon.com/History-Internet-Digital-Future/dp/1861897774
[U7] https://www.amazon.com/Global-War-Internet-Governance/dp/0300212526
[U8] https://www.amazon.com/Internet-Ethnographic-Approach-Daniel-Miller/dp/1859733891
[U9] https://www.amazon.com/Tubes-Journey-Internet-Andrew-Blum/dp/0061994952
[U10] https://www.amazon.com/Networks-States-Governance-Information-Revolution/dp/0262518570
[U11] https://www.amazon.com/How-Web-was-Born-Story/dp/0192862073
[U12] https://www.smashingmagazine.com/2016/08/a-beginners-guide-to-progressive-web-apps/
[U13] http://www.androidpolice.com/2017/12/05/google-wants-progressive-web-apps-replace-chrome-apps/
[U14] https://www.windowscentral.com/faq-progressive-web-apps-windows-10
[U15] https://cloudfour.com/thinks/apple-starts-work-on-progressive-web-apps/
[U16] https://hackernoon.com/ten-years-in-nobody-has-come-up-with-a-use-case-for-blockchain-ee98c180100
[U17] https://www.troyhunt.com/face-id-stinks/
[U18] https://mondaynote.com/apple-pc-tablets-twists-and-turns-1134e3a6d02e
[U19] http://www.twitter.com/pragpub
[U20] http://www.swaine.com

New Manager's Playbook

Onboard People, not Technology

by Marcus Blankenship

You're a developer, but you're at a point in your career where you find yourself managing others. Marcus shares tips on how to be as good at managing as you are at your "real" job. This month he talks about onboarding.

At my first programming job, it took three weeks to get my dev environment fully set up. I was only the second developer to work at the company, ever, so nothing was documented. The first person quit, which is why I had the job.

At my second, it only took four days, because I was the eighth person in the programming department, so I had seven other people to help me.

Today, my clients tell me things like,"we use Docker, so it takes less than an hour to onboard a new developer."

I'm glad new devs don't have to face the same frustrations I did.

But setting up a productive dev environment isn't onboarding.

Onboarding is setting up a *person* to work productively on your team.

A Leadership Smell

We might fall into the trap of believing once the dev environment is set up, we've done our part to make them successful, and the rest is up to them. That all they need is a computer, chair, a dev environment, and a project to work on.

This is a dangerous leadership smell. Dangerous for the managers, but mostly for the new developer.

The danger is the unspoken idea that after we apply onboarding to a new developer, they have everything they need to be productive.

Of course, when you read it, it may seem ridiculous.

But I get a whiff of this from managers at many organizations I work with. It typically comes out in a conversation like this:

Me: *How is your new programmer working out?*

Manager: *He's doing …"okay." Not quite what I hoped, but he'll be fine.*

Me: *What makes you say that?*

Manager: *Well, we got everything set up quickly, and got him all his access and onboarding. But he's just a lot slower than I expected.*

Me: *Why did you expect he'd be faster?*

Manager: *First, he's was supposed to be an expert in Java development. Second, we've automated our dev environment setup at great cost and effort. So, there shouldn't be anything standing in his way.*

As you might guess, this situation is more dangerous for the programmer than the manager, because it's the manager's expectations that aren't being met.

It's the manager who's disappointed in the speed at which things are progressing.

And it's the manager who feels that their onboarding process, whatever it is, should yield a developer who's "up to speed and ready to run."

So, if the manager believes they've done their part, the problem must be with the developer's motivation, drive, or skill. This ignores the most important keys to productivity: good relationships and a healthy environment. These are the things your developer needs to be onboarded to, and it takes much longer than just a few hours.

Onboarding Is Leading

Onboarding is a key activity of technical managers and leaders. It's not the responsibility of HR, and it's certainly not simply setting up the dev environment.

In my experience, onboarding should result in the following:

- Positive relationships with the the developer's team.
- Introductions to the project stakeholders.
- Clarity about the team's current goals and key upcoming projects.
- Clarity about the team's values.
- Knowing who to go to for help, and how often it's okay to ask for help.
- First-hand experience with the team's development process and ceremonies (Agile, Scrum, stand-ups, retros, etc.).
- First-hand experience with the key communication channels (team meetings, 1:1s, etc.).
- Having received a piece of adjusting and affirming feedback from their manager.
- Having given a piece of adjusting and affirming feedback to their manager and teammates.
- A clear sense of the tech and infrastructure used by the team.
- Having a productive development environment

How long should this take?

It should take you, the manager, at least a few weeks to initially onboard a new developer. After that, expect it to take a few quarters (3–9 months) for them to "come up to speed."

Of course, this will wildly vary according to the kind of projects you are working on. I've led teams where I spent a year onboarding a dev to work on a large ERP system. You might have experienced an even longer time period.

Onboarding is important leadership work that you need to be actively involved with. Like other work, you don't have to do it all yourself, but you need to make sure it all gets done, and done well.

Onboarding sets the stage for many years of productive work. It's about intentionally getting the right relationships in place from Day 1, which are the *key* to a productive development team.

Don't shortchange your team by pretending it's just about tools, paperwork, or diversity videos.

About the Author

Nearly 20 years ago I made the leap from senior-level hacker to full-on tech lead. Practically overnight I went from writing code to being in charge of actual human beings.

Without training or guidance, I suddenly had to deliver entire products on time and under budget, hit huge company goals, and do it all with a smile on my face. I share what I've learned here in *PragPub* and here [U1].

Johanna on Managing Product Development

Agile Approaches Help Teams Create and Manage WIP Limits

by Johanna Rothman

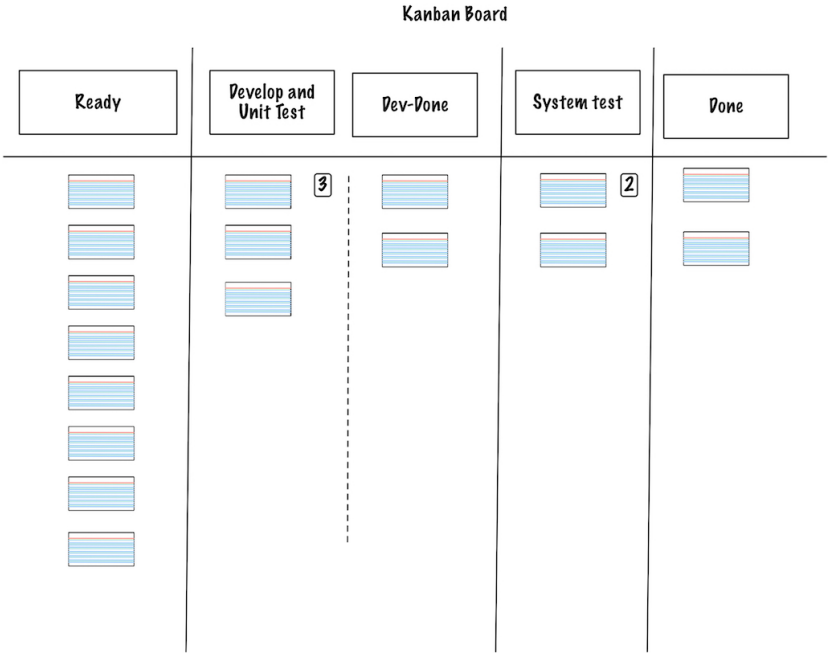
You'll feel less pressure from the backlogs, roadmaps, and project portfolio, and increase your throughput.

Even without knowing you, I am sure you have too much work to do. Your backlogs and roadmaps are crammed full of possibilities. Both the roadmaps and the project portfolios seem to grow daily. Your organization has grand ambitions for too few teams.

One way to manage all of this work is to create WIP (Work in Progress) Limits.

Iteration-based agile manages WIP by estimating what you can do in an iteration. You might count points. Or, you can use my preference, which is to count the (small) stories.

If you use flow-based approaches, you use kanban. You set WIP limits for the columns on your board.



In this image, there's a limit of eight items in the Ready column, three in Dev and Unit Test, and two in System Test. The interesting question is how did this team decide on these limits?

This is a large-ish team. They have eight people aside from the Product Owner: six developers and two testers. They decided to use a reasonable approximation for deciding on WIP limits:

- Take the number of people who work on a given column. That's the numerator. Here, for the Dev and Unit Test column, it's 6.
- Divide that number by 2. That gives you 3 as the WIP.

This team happens to have a policy of "No one works alone on the code," so their approximation works well. You might have a product that requires a front-end, middleware, and back-end Develop and Uniter for each feature. You would have a WIP limit of 2 on the Dev and Unit Test column because you need three people for each feature.

Now, there are only two testers on this team. How did they get to a WIP limit of 2?

The testers do not work together. They each work independently. That means they can each work on a story. They can't work on more than two stories at a time because they each take one story. This team agreed to work on stories until the story is done. There is no "Stuck" or "Waiting" column.

Every so often, the testers need help from the developers to complete a story. That's because the developers didn't realize something, or implemented something that is not quite right. In that case, the testers walk over to the developers and negotiate when the developer is available to help. Often, it's right away. Yes, the developers stop what they are doing, because finishing something they thought was Done is more important than starting or completing something new.

If you need a Stuck or Waiting column, you might add WIP limits to that column also. Why? Because you don't want that column to turn into a purgatory/limbo column, where partly finished stories go and never emerge. You might call it Urgent, although I tend to reserve Urgent for support issues.

If you use iteration-based agile, and you have unfinished work at the end of the iteration, consider using a kanban board so you can see where the work is piling up. You might have a problem with "Everyone takes their own story." (See Board Tyranny in Iterations and Flow [U1]).

Back in March, I wrote a column about Your Personal Project Portfolio, where I suggested you use a paper board to see all the work, because the size of the board literally helps to manage the WIP. Consider that for your team, too.

When people can see everything they have to do, and they can see where work builds up, they are more likely to stop working on some of the work so they can finish other work. It doesn't matter if you use iterations, using a paper board helps.

One team had this problem of too much work in their iterations. They dutifully estimated the work before they started an iteration. They crammed the iteration full of work, because they were always "behind" where the Product Owner wanted them to be. They had the problem of the organization wanting them to finish more work.

Partway through the iteration, they would receive urgent requests from Support — they had to address problems right away. That meant that the team had to stop working on features and start to work on the support issue. When they finally finished the support issue, they could return to the feature.

Sally, a senior developer was tired of feeling as if she was behind all the time. During the retrospective, she requested that the team — including the Product Owner — work on creating one-day or smaller stories. She explained her reasoning, "If we can finish something every day, we always know that even

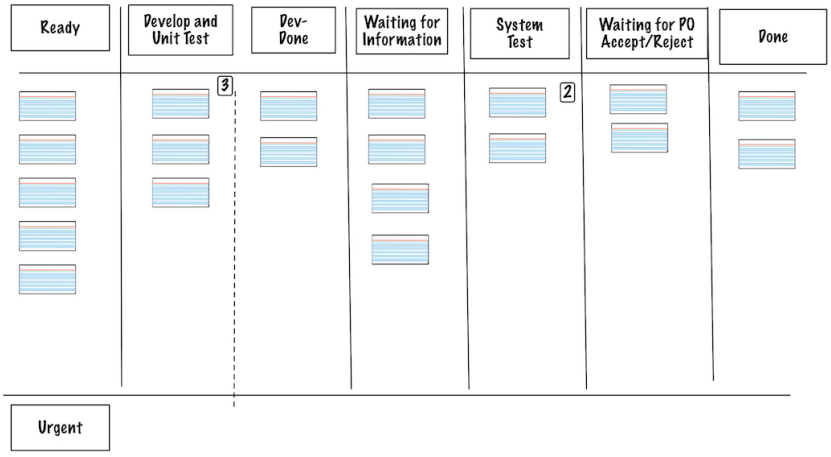
if we have to stop for support, we can return to and finish the feature the next day."

They had some success with smaller stories. That wasn't quite enough, because some of their support issues had delays. Sometimes, the team needed to ask other people for help. Sometimes, they needed information from another person to understand the problem. Sometimes, the support issue was working as they had designed it.

Sally had a bright idea: the team could change their board to show their wait states. She suggested this board to the team, as a paper board.

They worked through several iterations, updating their board as they encountered more wait states. Eventually, they had the board with the addition of Waiting Columns:

Team Board: Addition of Waiting Columns

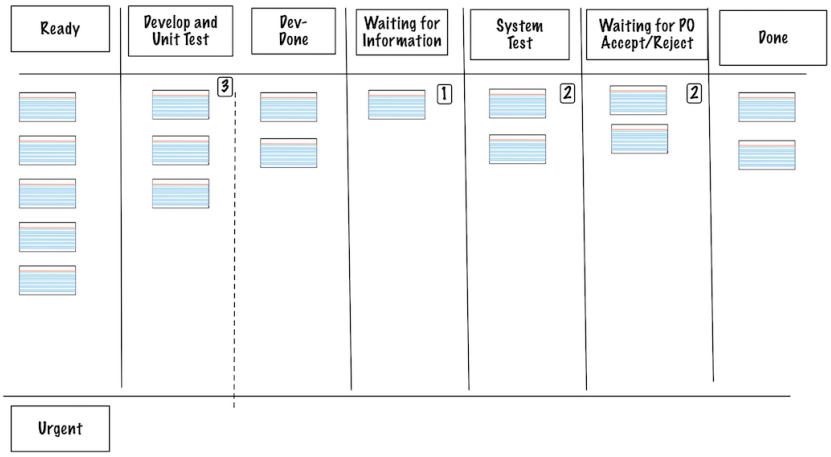


It took them several iterations to realize where all the wait states were. They evolved the board as they realized other states. And, they realized they also waited for the Product Owner to accept or reject the story before they could move it to Done.

Then, they decided to add WIP limits for the Waiting columns. The team discussed what their WIP limits should be. Sally advocated for a limit of 1 when they needed information. Most of the time, they needed information when they worked on a support problem.

The team also decided to create a WIP limit of 2 if they had to wait for the Product Owner to accept or reject a story. The team was determined to increase their throughput, which meant they had to keep the WIP limits lower.

Team Board: Addition of Waiting Columns



The team still used iterations, and they added their flow and WIP limits to make their work transparent. It took them several months, but they managed to release a story a day, and only rarely have anything in a wait column.

As they reduced their WIP, they felt less pressure from the backlogs, roadmaps, and project portfolio. WIP limits along with transparency and a determination to reduce story size increased their throughput.

About Johanna

Johanna Rothman, known as the "Pragmatic Manager," provides frank advice for your tough problems. She helps leaders and teams see problems and resolve risks and manage their product development. Johanna is the author of more than ten books, including *Create Your Successful Agile Project* [U2], *Predicting the Unpredictable: Pragmatic Approaches to Estimating Project Schedule or Cost* [U3], and *Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects, second edition* [U4]. She writes two blogs on her website, www.jrothman.com [U5], as well as a blog on www.createadaptablelife.com [U6].

External resources referenced in this article:

[U1] https://www.jrothman.com/mpd/agile/2016/08/board-tyranny-in-iterations-and-flow/

[U2] https://pragprog.com/book/jragm/create-your-successful-agile-project

[U3] https://pragprog.com/book/d-jrpredict/predicting-the-unpredictable

[U4] https://pragprog.com/book/jrport/manage-your-project-portfolio

[U5] https://www.jrothman.com

[U6] http://www.createadaptablelife.com

The Agile Movement ... is NOT very Agile!

We are the problem we warned ourselves about.

by Mike Beedle

In the first essay in our new Soapbox series, a co-author of the Agile Manifesto challenges the movement he helped start.

Let's say it out loud without restraints, political correctness, or sugarcoating, shall we?

"The Agile movement ... is NOT very Agile!"

This came up within another "little conversation" that I had with Jim McCarthy at the OSA (Open Space Agility) [U1] FB group, the other day. Yes, the the author of the Core Protocols [U2], and a guy who has the uncanny ability to make me more blunt, as if I needed any help.

But now this statement is really getting to me, burning me, really. How did this happen? Wouldn't you expect that WE, the people that tell the entire world that Agile is the "way to go," the "way of the future," the "better way," *be Agile in what we do?*

Let's break it down a bit, before it appears to be just another "complaining" rant:

1. Agile Manifesto

Let's start here, from the "original source" that defines what Agile is.

As much as I am proud of the merits and accomplishments of the Agile Manifesto [U3], it's really not a very good explanation of what Agile really is. Mind you, not that there is anything wrong with it, it is just not very well-explained. I think Alistair Cockburn has done a good job with his Heart of Agile [U4] with a better explanation, but I find myself at odds saying that the official source that defines Agile, needs improvement.

Shouldn't we be eating our own "dog food" ... and *iterate* and *IMPROVE*?

Jim McCarthy calls this "abandonment" irresponsible ... because the entire world depends on this; and I'm starting to side with him. It would appear to some, that we, the Agilistas, are "too proud" to take criticism, or afraid of "the mirror" ... to improve. Are we suffering from the very same syndrome that those that we so blatantly criticize for "not changing" fast enough?

I am currently working on my own version of a better explanation of Agile. I will publish an article on this soon, and I will seek community feedback, but again, the "official source" is static! That is not a good sign.

2. Agile Transformations

This needs very little explanation, because everybody knows that:

Agile Transformations ... done by arguably "Agile practitioners" and "Agile Coaches," supported by arguably "Agile leaders," ... are *not* Agile at ALL for the most part!

In fact, most Agile Transformations are managed through "traditional management" techniques. Some people even use Gantt charts! Most "transformation plans" look waterfall-ish: assess, envision, plan, train, coach, deploy, DONE! To make this worse, most of the "transformation plans" are done by external entities from those being transformed, which then FORCE their disrespectful transformation plan onto others, with little or NO concern for their well-being, desire, or ability to transform.

Is that the "Agile way"? Come on, we know better.

You've heard Daniel Joseph Mezick make this point, time and time again. And yes, some of us have offerings on "more AGILE Agile Transformations," for example OSA and ES — Agile Transformation, but we are not even close to making a dent in the overall picture.

3. Agile Frameworks

There are many things about the Agile Frameworks that are NOT very Agile:

1. *Development and Publishing:* The way they are published (as Jurgen Appelo recently pointed out in his article "Agile Methodologies Are Not Agile" [U5])
2. *Genericity:* The way they map to different domains — most Agile Frameworks are just for "software" or "product" development. Enterprise Scrum is the exception as it handles 50+ different activities.
3. *Scaling:* The way they scale. Most frameworks, have ONE option to scale, as if the scaling problem space was so simple or static. And most frameworks only scale within Software Development or Product Development. Again, Enterprise Scrum is the exception, as it has 1,024 combinations to scale.
4. *Configuration:* How configurable are Agile frameworks? Well, most Agile frameworks don't have fully explicit configuration options, or only make some parameters visible. For example, Scrum allows you to configure it with 1-, 2-, 3-, or 4-week Sprints, but there are very many other choices that are tacit, for example, the introduction of techniques such as User Stories, User Story Mapping, Moscow criteria, Architecture Scan, Release Planning. Enterprise Scrum is an exception here as well, as it provides very many explicit configuration options.
5. *Guidance:* For some, "Agile Framework," even in the guidance they provide, is not very Agile, e.g., SAFe; ok, it is more Agile than waterfall, but is it really Agile?

4. Agile Coaches, Agile Trainers, Agile Leaders, Agile Developers

Let's face it, as the interest in all things Agile has exploded, the percentage of truly qualified Agile people is decreasing literally by the minute. Sadly, our industry knowledge is very diluted. I worry about this.

Again, we are trying to change this by providing high-quality education, but let's accept it: we are not making any substantial progress in "reversing the trend."

My only hope is that even those "believing bad Agile" or "knowing bad Agile" can benefit from it, and that therefore, this "gross collective ineptitude" won't "kill the Agile movement" in its entirety.

5. Agile Implementations

Finally, whether at the team, program, portfolio, or Enterprise level, the vast majority of people doing Agile implementations, by whatever means, are not really following the Agile principles all that well:

- they don't collaborate all that well
- they don't deliver working software
- they don't talk to the customer all that much
- they don't respond to change very well etc.

Yes, we do training and coaching, but again, and especially with so many newcomers, the percentage of "good Agile implementations" is decreasing by the minute.

Is there a way to change this and improve?

Yes, it's an obvious answer ... let's make the Agile Movement more AGILE!

Well, until then, the above description is the ugly reality of the Agile movement right now.

And yet, I'm SUPER excited about Business Agility!

I dream of the benefit and positive change that working in a more Agile way across all domains may bring to people around the world. I dream of an enhanced ability to please customers, to innovate, to keep employees motivated, all while making profits for stakeholders.

What do you think? I'm just turning into an overly critical, straight-shooter, grumpy old man? Or ... wait, there's someone knocking at the door: I think they are representatives of the Agile Industrial Complex. They are now threatening me though a handheld loudspeaker:

"Shut up ... or else."

All jokes aside ... WE are part of this AIC — it's not us versus them, it's ALL of us involved with Agile one way or another — *WE are the problem*.

We, the Agilistas, ... are NOT that Agile!

Okay, here is the punch line, come on, you knew this was coming, I plead to You, the reader, that:

IF you understand the problems stated above, and since you've made it reading this far, that it is your moral obligation now that you understand these problems to help in whatever shape or form, and in whatever role you are involved with, to IMPROVE the Agile Movement as much as you can!!

I'll take that as a "YES." Thank you in advance. I plead to to the same!

About the Author

Mike Beedle is the creator of Enterprise Scrum, co-author of the Agile Manifesto, and CEO of Enterprise Scrum Inc. [U6] Mike is the second Scrum adopter and the first CEO to manage an entire company in an Agile way using Enterprise Scrum, a generic Scrum-like framework with scaling that he created and that that can agilize almost anything. He and his companies have introduced Scrum, Enterprise Scrum, and Business Agility to tens of thousands of people and thousands of companies, providing training, consulting, mentoring, and coaching. Mike teaches many courses from the Scrum Alliance, is an in-demand conference speaker, and has written several books. You can reach him at mikebeedle@enterprisescrum.com [U7].

External resources referenced in this article:

[U1] http://openspaceagility.com

[U2] http://www.mccarthyshow.com/online/

[U3] http://agilemanifesto.org

[U4] http://heartofagile.com

[U5] https://blog.agilityscales.com/agile-methodologies-are-not-agile-d6411126d80

[U6] http://www.enterprisescrum.com

[U7] mailto:mikebeedle@enterprisescrum.com

Feature Toggles

Reducing Complexity and Shortening Feedback Loops

by Darren Sim

Feature Toggles were a common technique twenty years ago. It's time to bring them back.

As software professionals, our success is measured by our ability to improve our users' lives.

Unlike other engineering fields which have been around for hundreds of years, the software engineering discipline has existed for barely a hundred years. As an industry, software development is still largely part-art, part-science.

With the ever-evolving business and technological landscape, many of our past experiences serve merely as references. Adding to the unknowns, users seldom know what features they want, but they know what they don't want. More often than not, it takes several attempts of trial-and-error before software teams become successful in delivering features that stick.

The ability to continuously deliver working software into the hands of customers has served as a competitive advantage for leading tech companies like Amazon and Facebook. Not only did this capability allow engineering teams to *deploy code into production* in small chunks, *quickly* and *safely*, it also *shortened the feedback loop* — allowing teams to learn fast and respond to changes quickly.

In the practice of Continuous Delivery (CD), it's important that the application is continuously integrated (CI), and deployable at all times. This can be challenging, especially in situations where a feature spans multiple commits, and an urgent fix needs to be released before the development of that feature is complete. This highlights the need for deployment and release to be decoupled.

When a feature is completed and deployed to production, teams also need a way to experiment safely, beta testing with a small group of users and being able to roll back quickly if an undesirable experience is observed.

Feature Toggles offers options to overcome to these challenges.

In this article, we will explore what Feature Toggles are and how we can use them, before discussing best practices.

What Are Feature Toggles?

Feature Toggling is not a new concept and was a common practice more than 20 years ago, in the absence of reliable and easy-to-use concurrent versioning systems (CVS). In recent years, Feature Toggles experienced a resurgence with the popularization of DevOps practices and mindsets within software development teams.

Also known as "Feature Flags," "Config Flags," "Flippers," "Feature Bits," and "Conditional Feature," Feature Toggling is a set of patterns that help teams

deliver new functionality to users rapidly but safely, separating deployment from release, allowing teams to modify system behavior without changing code. Implemented by simply wrapping an if-statement around a code block that we are trying to isolate, the condition of this if-statement can be a simple Boolean or complex decision trees with multiple paths.

The basic idea here is that we have a configuration system that allows the development team to toggle features on or off. When the application is executed, it will use these toggles to decide what features to display. The actual implementation will vary — ranging from a hard-coded configuration value within the application where changing the toggle configurations require a redeployment of the application, to reading toggle states from a text file or a web service, allowing teams to modify toggle configurations at runtime.

Some of the benefits that Feature Toggles bring to the table include helping teams eliminate nasty merge conflicts, reduce deployment risks, and achieving a shorter feedback loop. However, there're always two sides to a coin. Apart from introducing technical debt the moment we use Feature Toggles in our code, Feature Toggles also results in code being more fragile, harder to test, harder to maintain, and harder to support. Hence, Feature Toggles should always be short-lived and used on a need-be basis rather than it being the silver-bullet to software development.

One of the most common misunderstandings of Feature Toggles is that it can double as a licensing and authorization service. This results in tight coupling of responsibility and should be avoided.

Types of Feature Toggles

In his thought-leading article "Feature Toggles (aka Feature Flags)," Pete Hodgson extended Martin Fowler's original post "FeatureToggle," nicely categorizing Feature Toggles into FOUR categories:

	Release Toggle	Experimental Toggle
Dynamism	Largely static, changing release toggle decision usually done in code; requiring redeployment.	Highly dynamic; changes with each request
Longevity	Transitionary by nature; should not stick around longer than 1 weeks.	Long enough to generate statically significant results, usually kept around for hours to weeks.
Benefits	Allows incomplete and untested code-paths to be deployed to production.	Provides development team with means to make data-driven decisions.
Disadvantages	Results in dead code that will not add value to anyone.	Highly dynamic in nature, requiring a third-party services; a conscious technical debt.
Implementation	By hardcoding the condition in the if-statement to false.	Usually by means of an external feature toggle web service.
When to Use	When developing code that is not ready for consumption and has a high risk of breaking the application.	When performing multivariate testing (e.g. AB Testing)

Short-Lived Toggles

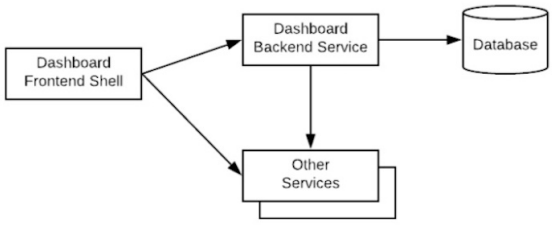
	Operation Toggle	Permission Toggle
Dynamism	Relatively static; changes be done with runtime reconfiguration.	Highly dynamic in nature; changes may happen between requests.
Longevity	Usually from weeks to years.	Usually spanning months to years.
Benefits	Allows system operators to disable / degrade a low performing feature quickly without having to perform a redeployment of the code.	Perform targeted experiments and learnings.
Disadvantages	Tends to stick around forever; leading to increased system complexity.	Tends to stick around forever; leading to increased system complexity.
Implementation	By means of a config file on a CDN or other means of data cache services.	Usually by means of an external feature toggle web service.
When to Use	When rolling out a new feature with unclear performance impact.	Used in cases of dog-fooding, alpha and beta programs.

Long-Lived Toggles

While this categorization is by no means all-encompassing, it serves as a good reference when making decisions on designing Feature Toggle implementations.

My Team's Experience with Feature Toggles

My team has been working with Feature Toggles for the last 18 months and have gone from using an internally developed, maintained and hosted Feature Toggling service, to a commercially operated SaaS-based solution.



Our bread and butter as a team is to build and operate a management dashboard for accountants and bookkeepers. Through this dashboard, our users will be able to get a bird's-eye view of all their client's accounting books (also known as "ledgers") and categorize transactions (also known as "transaction coding") where required. Data is fed to this management dashboard through a plethora of web services that are maintained by various teams within the organization.

The management dashboard comprises a front-end shell that deals with everything that runs on the browser, as well as a backend service that acts as a data aggregation and caching service, used exclusively by the front-end.

In our team, we have adopted practices such as Continuous Integration (CI), Continuous Deployment (CD), Trunk-based Development, and Test-Driven Development to help us deliver working software quickly and safely.

Given that we practice elephant carpaccio during story slicing and work mainly in a mob or as pairs, we found Release Toggles to be less useful. Instead, we adopted the Feature Branching approach when developing new features. With Feature Branching, we are able to make multiple commits while working on the feature, without generating excessive noise in GIT. As we squash our commits with links back to the original feature branch, GitHub allows us the

ability to rid the feature branch, while still being able to look back at those micro-commits.

The two toggles that we used were *Operations Toggles* and *Permission Toggles*.

As a downstream service that depends on many other services for data, the ball is seldom in our court. Each of these services that we depend on for data is usually built and operated by different teams, and has varying performance thresholds.

Operation Toggles has been especially effective; offering us the capability to gracefully degrade or disable calls to these dependent services in the event that the business dashboard or any of its downstream services experience a traffic surge, resulting in undesirable system performances and/or behaviors.

Given that this service doesn't have to be highly dynamic, we developed a simple setup that required little to no maintenance. Backed by an AWS S3 bucket to store the static configuration files and a CloudFront Distribution to deliver the configuration payload, this infrastructure cost us close to nothing to operate.

In order to better understand if features were developed and delivered in a way that actually increases the productivity and effectiveness of our users, we have also introduced beta programs to get early feedback before a feature is released to the entire user base.

Given that this service needs to be able to form toggle configurations dynamically based on a user's profile, locality, and other factors, a highly dynamic service is required. Leading vendors that offers such services include FeatureAvailability.IO, Launch Darkly, and Split.IO. Given that these services usually offer very similar capabilities and pricing, decisions usually boil down to product roadmaps, and quality of support. Flagr and Petri are two popular open source options that can be downloaded and self-hosted.

Commercially operated solutions typically charge based on a per-request model; hence implementing Feature Toggles on stateless functions (e.g., AWS Lambda) can be costly given that caching of this state is not possible. Feature Toggle states are usually cached for up to a minute within an application to reduce the chattiness between the application and the Feature Toggle service.

Lessons Learned

Feature Toggling is a useful technique that helps teams avoid painful merge conflicts and experiment and learn fast; however, it is not without risks. Here are some of our lessons learned and best practices:

Business Metric and Traffic Monitoring Dashboard

A key benefit of Feature Toggling is the ability to perform experiments and learn from feedback, quickly and safely. Hence, having a business metric dashboard is of high importance, allowing us to understand the business impact of a new feature. Consider an e-commerce system: If the introduction of a new feature results in massive decrease in sales, that is something that is probably useful feedback that is worth looking into in greater detail.

Avoid Using Release Toggles

As Martin Fowler has put it, while release toggles is a useful technique that is used by many teams, they should be used as a last option when putting new features into production. Instead, we should be adopting the practice of breaking features down more so that they can be safely introduced into the product. The benefit of doing so is similar to that of any strategy advocating small and frequent releases. Risks can be reduced and development teams can get early feedback that can be used to improve the feature.

Avoid Coupling Decision Point and Toggle Router

Decision point is defined as the location in the code that calls the toggle router, while the Toggle router contains the logic to check if a certain decision point is active by reading toggling configurations to detect features, or simply detecting a Toggle Context.

```
if(User == “john@doe.com” && Weather == “Sunny”){
      DoSomething();
}
if(User.Segment == “Beta”){
      DoSomething();
}
```

The two examples above exemplify patterns we want to avoid, as they tightly couple toggles' decisions into the application logic.

```
if(FeatureSwitchSvc.IsPsychicEnabled){
      DoSomething();
}
```

What we want instead, is to de-couple decision points from decision logic by introducing a decision object that implements that logic. In this way, the strategy-pattern can be applied, encapsulating routing conditions to the routing layer. Thus, a change in feature grouping will not break the code.

Avoid Nesting Feature Toggles

If nesting if-statements increases the cyclomatic complexity, nesting Feature Toggles increases that by many folds.

Dashboard with Feature Toggles Longevity

Like branches, long-lived Feature Toggles should be avoided as they add significant complexity to the application and cause the code-base to be brittle over time. While it is easy to visualize how many stale branches you have in GitHub, long-lived Feature Toggles are less obvious and run the risks of snuggling in forever. Hence, a physical or digital dashboard in the team area that keeps track of the longevity of each Feature Toggle is always a good practice.

Always Check for Security

One of the common misconceptions is that if you hide a feature behind a feature flag, it is not accessible and hence, we do not have to worry about

security. This is untrue and a dangerous mindset to have in software development, a phenomenon known as "the security sandwich." Consider a Feature Toggle implemented on the front-end. What is stopping a malicious user from manually setting that flag to true in the console? Security as an afterthought is always a recipe for trouble.

Automated Tests Should Exercise Both Toggle On and Off States

If TDD is practiced in the team, it is always a good habit to practice red-green-refactor. When implementing Feature Toggles, it is always important that automated tests exercise flows with the switch turned on, and off, to avoid regressions. While this is not a silver bullet, it helps radiate obvious regressions. The added complexity in testing introduced by Feature Toggles warrants reviewing existing test strategies; constant and effective communication between between members of the development team is also important.

About the Author

Darren is an experienced software programmer, and conference speaker who is passionate about improving the lives of others through the use of technology. Having been involved in the software industry in various roles since 2003, Darren is currently working in a Software Engineering Function at MYOB. Darren currently takes a keen interest in Modern Agile, Software Craftsmanship, Evolutionary Software Architecture and Security. Reach out to Darren via darren[at]darrensim.io or on Twitter @darrensimio.

Defining MVP, MMF, MMP, and MMR

Words Matter

by Scott Ambler

These four terms are often misused. Keeping the terms straight will help keep your development efforts on track.

The term *minimal viable product* (MVP) has achieved buzzword status in recent times and I'm now hearing people throwing around the term MVP almost on a daily basis. Sometimes they're using it correctly, but many times they aren't. Frankly it's driving me nuts.

The issue is that it's common for people to say MVP when they are actually talking about a *minimal marketable feature* (MMF), a *minimal marketable product* (MMP), or even a *minimal marketable release* (MMR). As you can see, these terms are very similar to one another so we shouldn't be surprised that there's a bit of confusion around them. So let's try to clear things up.

First, Some Definitions

Figure 1 below overviews how these following terms relate to one another:

- Minimal Viable Product (MVP) [U1]. An MVP is a version of a new product that is created with the least effort possible to be used for validated learning about customers. MVPs are used to run experiments to explore a hypothesis about what your customers really want. They are much closer to prototypes than they are to the "real" running version of your end product. A development team typically deploys an MVP to a subset of your (potential) customers to test a new idea, to collect data about it, and thereby learn from it. MVPs are created to help you to find the features that customers are actually interested in.
- Minimal Marketable Feature (MMF) [U2]. An MMF is the smallest piece of functionality that can be delivered that has value to both the organization delivering it and the people using it. An MMF is a part of an MMR or MMP.
- Minimal Marketable Release (MMR) [U3]. Successful products are deployed incrementally into the marketplace over time, each "major" deployment being referred to as a release. An MMR is the release of a product that has the smallest possible feature set that addresses the current needs of your customers. MMRs are used to reduce the time-to-market between releases by reducing the coherent feature set of each release to the smallest increment that offers new value to customers/end users.
- Minimal Marketable Product (MMP) [U4]. An MMP is the first deployment of a Minimal Marketable Release (MMR). Having said that, the terms MMP and MMR are often used interchangeably. An MMP is aimed at your initial users, typically innovators and early adopters. The key is to develop an MMP for the few, not the many, and thereby focus on the key features that will delight this core group of people. An MMP is a tool

to reduce the initial time to market because it can be developed faster than a feature-rich product.

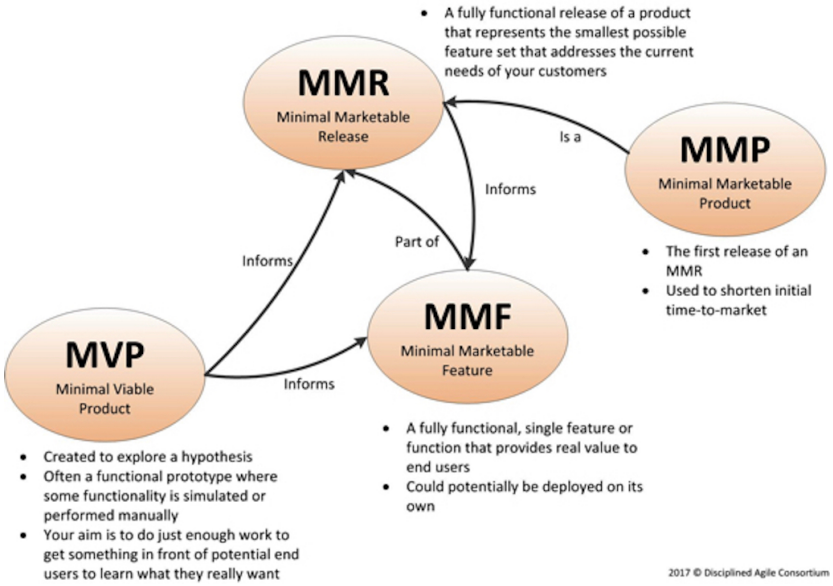


Figure 1: *The relationship between MVP, MMF, MMR, and MMP*

Is it Minimum or Minimal?

Given that I'm being picky about terminology, I realized that there isn't agreement as to whether we should use the term MINIMUM viable product or MINIMAL viable product (and similarly for MMR, MMP, and MMF). Once again, the words are very close:

- Minimum. This refers to the least quantity or lowest possible amount.
- Minimal. This refers to barely adequate or sufficient (similar to the Agile concept of *just barely good enough* (JBGE) [U5]). Minimal is an adjective derived from the word minimum.

As you can see, very nuanced. For our purposes, the term minimal is more appropriate than minimum because it brings in the idea that it must be sufficient to fulfill the needs of our product's customers. Or more precisely, what we believe to be the current pressing needs of our stakeholders.

Example: Developing a New Product

Now let's work through an example of the development of a fictional product. One day while shopping in the local mall my phone ran out of power. This proved to be a problem for me because I had a conference call that I had to be on, forcing me to cut my shopping trip short to go home and take the call there. This experience made me realize that there's a potentially untapped market need as I would have been very willing to pay to charge my phone while at the mall. Note: I am fully aware that products such as Safecharge [U6] and Brightbox [U7] exist, but let's pretend they don't for the sake of this example.

Just because I'm willing to pay for this doesn't mean that others will. To determine whether this could be a profitable endeavor I decide to follow Disciplined Agile's Exploratory Lifecycle [U8] (see Figure 2), which is based on

Lean Startup's hypothesis-driven approach. My plan is to iteratively build a series of MVPs to explore this product idea.

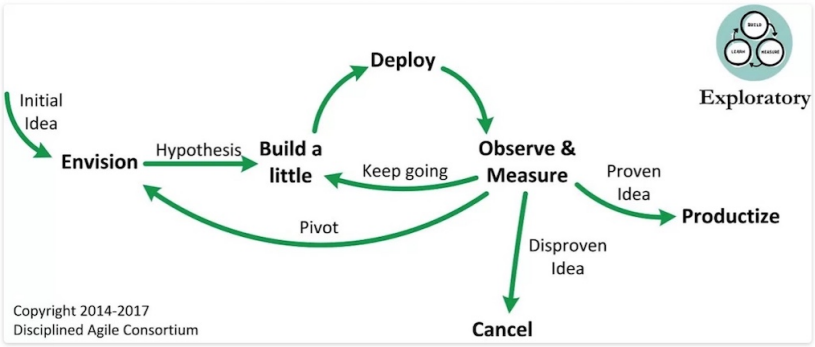


Figure 2: *The Exploratory Lifecycle*

Over a several week period I work through a series of minimal viable products (MVPs):

1. MVP #1: A power bar on a table. I start with a very simple approach: I talk the mall manager into allowing me to put a table against a wall for a one-week period to run an experiment. I plug a power bar into a nearby outlet and put it on the table. On the wall I have a sign that indicates this is a phone charging station. Throughout the week, I stand by the table telling people about the service and tell them I'll keep an eye on their phone if they want to go shopping while it charges (I quickly discovered that nobody is willing to actually do that, or at least they're not willing to trust me, hmmm …). For anyone willing, I have them take a short survey asking them what they think about the service.

2. MVP #2: I add several common power cords. On the first day several people indicated that they would use the service but unfortunately didn't have their charging cable with them. So at the end of the first day, I bought several power adapters from an electronics store in the mall. Sure enough, over the next few days I had more people willing to charge their phones at my table. By the end of the week, I had gathered a fair bit of data that showed there was general interest in the idea but that a major problem was the inability to safely leave a device to charge while they go off to shop.

3. MVP #3: I move to a cafe. The following week, I run a similar experiment in a cafe a few blocks away from where I live. Interestingly, I have several people ask to borrow a power cable from me so that they could power their phone while sitting at their own table. The cafe already has power sockets for people to charge devices, and it's fairly common for people to camp out in the cafe for several hours with a laptop or table plugged into the wall. After several days, it becomes clear to me that a cafe isn't a good option for a charging station.

4. MVP #4: I add lockable cubby holes. Over the next week, I decide to build out a more sophisticated solution, a wood cabinet that has 16 cubby holes for charging devices. Each cubbyhole has a specific type of charge cable, so if you want to charge a phone you need to use a cubby with the right type of cable. Each cubby has a door with a physical key lock. I go back to the mall, in the same location as I'd been in previously, and

instead of a survey, I interview people to discover what they they think, how they would make it better, and what they'd be willing to pay for such a service.

This series of experiments led me to identify a collection of minimal marketable features (MMFs) that this product should offer:

1. Lockable cubby holes. People will only leave their phones and other devices if they're safe. Each cubby hole needs to be locked in such a way that only the person who left their phone in the cubby can get access to it. This could be an electronic lock where people can type in a private code or a physical key-based system.
2. Common phone power cords. We need to be able to support charging a range of devices. Each cubby should have several common power cord/cables as well as a normal power plug.
3. Easily accessible location that doesn't offer charging alternatives. Malls and restaurants are good options, but public areas that already support device charging (like cafes) are not.
4. Payment processing. We want to support credit card and possibly blue-tooth payment strategies such as Apple Pay. Payment options need to be investigated still.

Over the next two months, we built a minimal marketable product (MMP). The MMP was five large boxes, each of which had 16 cubby holes for small devices such as phones. We wanted five boxes so that we could place three boxes in the mall where we had run our initial experiments and two boxes in another smaller mall on the other side of the city. We made each box from folded sheet metal with clear, thick plastic doors so that people can see their devices. For security and payment processing we built a device that used a small touch screen (it was actually a large smart phone) as an input device attached to a card swipe for capturing both credit and debit card payments.

Over time, we continued to evolve the product via a series of minimal marketable releases (MMRs). We ran some experiments in a public library where we discovered that library patrons wanted to charge large devices such as tablets and laptops as well as smaller devices. We developed a "Library Charging Station" that had eight small device cubbies and six large device cubbies. We also hired a designer to develop a sleeker looking box when one mall manager told us that they loved the concept but wouldn't allow our boxes into their more upscale locations until our boxes were more attractive.

Why The Confusion?

There are several reasons why there is significant confusion in the marketplace:

- These are closely related concepts with very similar names.
- Various authors over the years have used these terms in different ways, thereby muddying the waters.
- Few people go back to the original source of a concept and instead choose to read derivative work (such as this article). In effect, they suffer from the whisper game — you heard the term MVP from one person, who heard it from someone else, who heard it from someone else, and so on.

My hope is that this article, and the supporting poster that is now available on the Disciplined Agile Consortium site, has helped to clear up some of this confusion.

About the Author

Scott is a Senior Consulting Partner of Scott Ambler + Associates, working with organizations around the world to help them to improve their software processes. He provides training, coaching, and mentoring in disciplined agile and lean strategies at both the project and organizational level. Scott is the (co)-creator of the Agile Modeling (AM) and Agile Data (AD) methodologies and the Disciplined Agile (DA) framework. He is the (co-)author of several books, including *An Executive's Guide to the Disciplined Agile Framework*, *Disciplined Agile Delivery*, *Refactoring Databases*, *Agile Modeling*, *Agile Database Techniques*, and *The Object Primer 3rd Edition*. Scott blogs regularly at DisciplinedAgileDelivery.com [U9] and his company's home page is ScottAmbler.com [U10].

External resources referenced in this article:

[U1] https://en.wikipedia.org/wiki/Minimum_viable_product

[U2] http://www.netobjectives.com/minimum-marketable-features-mmfs-explained

[U3] https://chronologist.com/blog/2012-10-23/virtues-of-minimum-marketable-releases/

[U4] https://www.romanpichler.com/blog/minimum-viable-product-and-minimal-marketable-product/

[U5] http://agilemodeling.com/essays/barelyGoodEnough.html

[U6] https://www.thesafecharge.com

[U7] https://brightboxcharge.com

[U8] http://www.disciplinedagiledelivery.com/lifecycle/exploratory-lifecycle/

[U9] http://www.disciplinedagiledelivery.com/

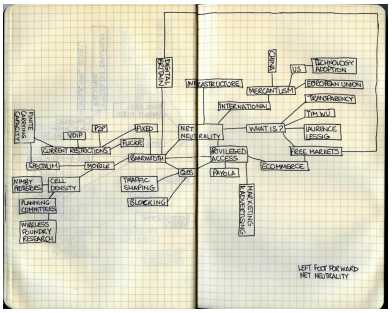
[U10] http://scottambler.com/

Net Neutrality

A Look at the Debate, Its History, and Its Likely Eventual Outcome

by Michael Swaine

Your editor thought he understood the net neutrality debate pretty well. Then he did some research.



"If you wish to converse with me, define your terms." Voltaire

I wish to converse with you about net neutrality, so let's define our terms. Or term. What, precisely, is net neutrality?

Net neutrality, short for network neutrality, has an official definition. It is a network design principle articulated and named by Tim Wu [U1], currently the Isidor and Seville Sulzbacher Professor of Law at Columbia Law School, in a proposal [U2] he wrote back in 2002. Wu has some cred in this area. He clerked for Justice Stephen Breyer and Judge Richard Posner, worked at the White House National Economic Council, at the Federal Trade Commission, for the New York Attorney General, and in the Silicon Valley telecommunications industry, and wrote the books *Who Controls the Internet* [U3], *The Master Switch* [U4], and *The Attention Merchants* [U5].

Wu defines network neutrality as *non-discrimination*: "absent evidence of harm to the local network or the interests of other users, broadband carriers should not be allowed to discriminate in how they treat traffic on their broadband network on the basis of internetwork criteria." In other words, they should act like the electric grid. "The electric grid does not care if you plug in a toaster, an iron, or a computer. Consequently it has survived and supported giant waves of innovation in the appliance market." That's network neutrality. (For the nitty-gritty definition and discussion, check his FAQ [U6].)

Wu's proposal lays the groundwork for all subsequent discussion of net neutrality. It follows a clear plan. He documents a problem, giving concrete examples of troubling restrictions on user behavior by broadband providers. He also provides examples of clearly desirable or necessary restrictions. The challenge, as he presents it, is to define policy that discourages the bad restrictions and encourages the good ones. Network neutrality is his proposal for threading that needle.

"The non-discrimination principle," he says, "works by recognizing a distinction between local network restrictions, generally allowable, and inter-network restrictions, viewed as suspect." The non-discrimination principle *defaults* to not discriminating in broadband usage, but also embodies "an effort to develop forbidden and permissible grounds for discrimination." That effort hinges on distinguishing between local network and inter-network traffic, and Wu provides examples of what he means by restrictions in these two areas:

Local Network Restrictions

- Limits on Bandwidth
- Quality of Service within the service providers' network

- Reaching local devices
- Local broadcast, multicast
- Policing of Ethernet/Frame Relay equipment

Inter-Network Restrictions

- Blocking of specified Internet addresses
- Limits on acting as a server
- Bans on Internet VPN services
- Bans on certain applications by TCP port number
- Bans or limits on applications (based on cookie information)

"In practice what this means," the *BBC* recently summarized, "is that ISPs ... cannot block content, speed up, or slow down data from particular websites because they have been paid to do so. And they can't give preferential treatment to their own content at the expense of their competitors." [*BBC*, December 14, 2017]

The Economist chimed in: "Put simply it is the principle that all internet traffic, whether from Netflix, Tinder, or a news website, is treated equally by the "pipe" companies carrying that traffic, like AT&T or Verizon." [*The Economist*, December 15, 2017]

The Law of the Land

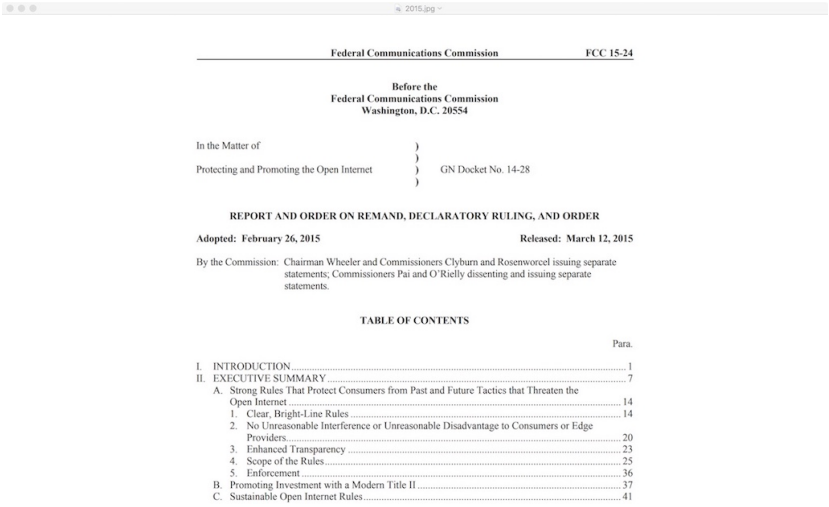
Wu considers possible objections to his proposal:

- Does it overly interfere with broadband carriers' ability to earn a return on their infrastructure investment?
- Can local restrictions be used to achieve the same result as internetwork control?
- Does net neutrality interfere with administration of internet addressing?

He addresses all these objections in his proposal [U7]. He concludes that net neutrality is a desirable design principle and would be good policy.

In terms of policy, it comes down to how broadband internet should be viewed and how it should be regulated: as an information service under the jurisdiction of the Federal Trade Commission (FTC), or as telecommunications under the jurisdiction of the Federal Communications Commission (FCC). Wu's view is that broadband internet is telecommunications.

This view was embraced by the Obama administration and the FCC, and in 2015, the FCC acted, and Net Neutrality became the law of the land [U8].



Federal Communications Commission FCC 15-24

Before the
Federal Communications Commission
Washington, D.C. 20554

In the Matter of)
)
Protecting and Promoting the Open Internet) GN Docket No. 14-28
)

REPORT AND ORDER ON REMAND, DECLARATORY RULING, AND ORDER

Adopted: February 26, 2015 Released: March 12, 2015

By the Commission: Chairman Wheeler and Commissioners Clyburn and Rosenworcel issuing separate statements; Commissioners Pai and O'Rielly dissenting and issuing separate statements.

TABLE OF CONTENTS

Para.

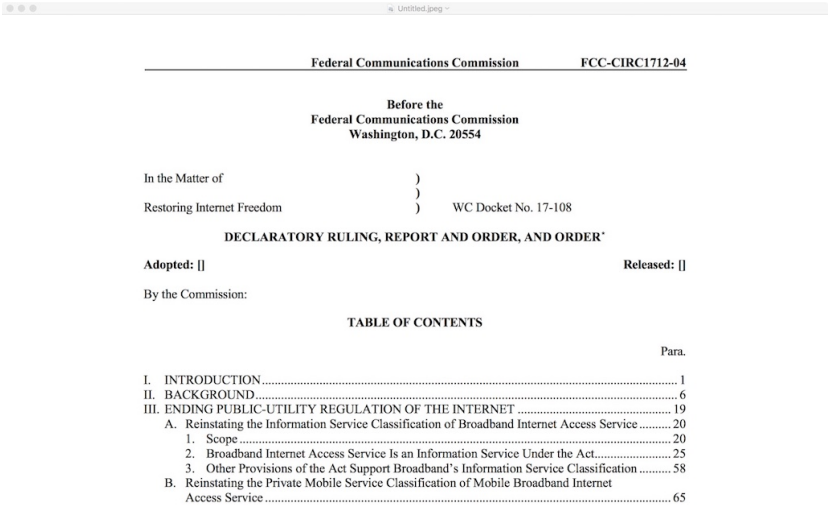
I. INTRODUCTION 1
II. EXECUTIVE SUMMARY 7
A. Strong Rules That Protect Consumers from Past and Future Tactics that Threaten the Open Internet 14
1. Clear, Bright-Line Rules 14
2. No Unreasonable Interference or Unreasonable Disadvantage to Consumers or Edge Providers 20
3. Enhanced Transparency 23
4. Scope of the Rules 25
5. Enforcement 36
B. Promoting Investment with a Modern Title II 37
C. Sustainable Open Internet Rules 41

"With the rise of Netflix and its ilk in streaming media," *The Economist* explained, "broadband companies began to suggest that they may have to charge more for some types of traffic, or slow down some services ('throttling' them). Net-neutrality activists argued that if providers could discriminate between different types of traffic, they would have far too much power over the internet. They could privilege their own services over competitors', or they could even throttle or block some services they did not like. The Obama-era rules were designed to prevent that." [*The Economist*, December 15, 2017]

Reversal

And so it stood until last year. In 2017, under new chair Ajit Pai, the FCC reversed this decision.

"The vote — three Republican commissioners in favour, two Democrat commissioners against — enacts the Restoring Internet Freedom initiative, which is widely seen as giving ISPs greater power to limit internet access while favouring certain data streams." [Creed Newton, *Al Jazeera* [U9], December 2017]



Federal Communications Commission FCC-CIRC1712-04

Before the
Federal Communications Commission
Washington, D.C. 20554

In the Matter of)
)
Restoring Internet Freedom) WC Docket No. 17-108

DECLARATORY RULING, REPORT AND ORDER, AND ORDER*

Adopted: [] Released: []

By the Commission:

TABLE OF CONTENTS

Para.

I. INTRODUCTION 1
II. BACKGROUND 6
III. ENDING PUBLIC-UTILITY REGULATION OF THE INTERNET 19
A. Reinstating the Information Service Classification of Broadband Internet Access Service 20
1. Scope 20
2. Broadband Internet Access Service Is an Information Service Under the Act 25
3. Other Provisions of the Act Support Broadband's Information Service Classification 58
B. Reinstating the Private Mobile Service Classification of Mobile Broadband Internet Access Service 65

"Technically," the *BBC* explained, "the vote was to reclassify broadband internet as an information service rather than telecommunications. The consequence of this is that the FCC will no longer directly regulate ISPs. Instead jurisdiction will pass to another regulator, the [FTC]."

Except that the FTC won't really regulate ISPs either. "Its key responsibility will be to check that the companies disclose if they block data, throttle it or offer to prioritise traffic, rather than stopping such behaviour." [*BBC*, December 14, 2017]

Comcast, as one of the biggest ISPs, says it has no intention of doing this.

"We have repeatedly stated, and reiterate today, that we do not and will not block, throttle, or discriminate against lawful content. These fundamental tenets of net neutrality are also key components of our core network and business practices — they govern how we run our Internet business."

And that sounds nice, but of course a company's business practices can change at any time and do not offer any actual protection of consumer interests. Comcast's policies that "govern how we run our Internet business" can no more be relied on than Terms of Use statements.

Pai's Defense

Pai's defense for the reversal drew on the first of Wu's three possible objections to net neutrality.

"Pai has said he fears that internet service providers are not investing in critical infrastructure such as connections to low income or rural households because the net neutrality rules prevent them from making money from their investments," the *BBC* reported. [*BBC*, December 14, 2017]

Pai argued "that the 2015 net neutrality rules, which essentially redefined Internet service providers as public utility companies ('common carriers') and regulated them as such, imposed unnecessary and burdensome restrictions on the industry. [He] likened the regulations to 'micromanaging' the Internet, said they stifled competition and discouraged investment and innovation." [David Emery, *Snopes*, December 15, 2017]

The new rules allow broadband providers to give preferential treatment to some services' data and to charge consumers more to access certain content. All the providers have to do is publicly disclose these practices. If they don't, someone can complain to the FTC, which will have the power to demand that they disclose, but no power to prevent these actions.

But what recourse do citizens have if they don't like the behavior the provider is engaging in and disclosing?

Commissioner O'Rielly suggested [U10] that Congress could step in to address particularly troubling restrictions:

"O'Rielly said that Congress would be a more appropriate body for creating future rules governing ISPs, and suggested that paid prioritization of Internet traffic, enabled by this overturn, would benefit emerging applications such as telemedicine and self-driving cars." [*Library Journal*, December 14, 2017]

That was from a publication of the American Library Association, particularly relevant since the internet is the library of today. ALA expressed its disagreement [U11] clearly:

"The majority of the FCC has just dealt a blow to equitable access to online information and services which puts libraries, our patrons, and America's

communities at risk. ... By rolling back essential and enforceable net neutrality protections, the FCC has enabled commercial interests at the expense of the public who depends on the internet as their primary means of information gathering, learning, and communication. We will continue to fight the FCC's decision and advocate for strong, enforceable net neutrality protections."

The Dissent

And two of the five FCC commissioners also disagreed in the strongest possible terms [U12].

Opponents of this rollback have been characterizing it as an attack on the open internet, and "a dismantling of not only the safeguards put in place by the Obama administration but ones that have been embedded in the World Wide Web since its invention in 1989." [Eric Allen Been, *VOX*, December 14, 2017]

Let's unpack that assertion. What were the "safeguards put in place by the Obama administration?"

Obama's FCC's wrote [U13]:

"[T]he court observed that nearly 15 years ago, the Commission constrained its ability to protect against threats to the open Internet by a regulatory classification of broadband that precluded use of statutory protections that historically ensured the openness of telephone networks. The Order finds that the nature of broadband Internet access service has not only changed since that initial classification decision, but that broadband providers have even more incentives to interfere with Internet openness today. To respond to this changed landscape, the new Open Internet Order restores the FCC's legal authority to fully address threats to openness on today's networks by following a template for sustainability laid out in the D.C. Circuit Opinion itself, including reclassification of broadband Internet access as a telecommunications service under Title II of the Communications Act."

But *VOX*'s article also claims that the Pai ruling dismantles safeguards "that have been embedded in the World Wide Web since its invention in 1989." What are these?

Here's Tim Berners-Lee, writing prior to the reversal:

TBL

"When I invented the World Wide Web in 1989, I didn't have to pay a fee, or ask anyone for permission to make it available over the internet. All I had to do was write a new app and plug my computer into the net. If US net neutrality rules are repealed, future innovators will have to first negotiate with each ISP to get their new product onto an internet package. That means no more permissionless space for innovation. ISPs will have the power to decide which websites you can access and at what speed each will load. In other words, they'll be able to decide which companies succeed online, which voices are heard — and which are silenced.

"Net neutrality separates the connectivity market from the content market. As separate markets, both have flourished. But if the US allows the internet to become like the old cable TV model — with the same firms controlling the

cables and the content — competition in both markets will suffer. As other countries maintain separate and fiercely competitive markets, America will decline as the world's chief digital innovator.

"In the early years of the internet, ISPs didn't have the technical capacity to discriminate traffic online. Their computers were not fast enough and so net neutrality was a fact of life. Over time, as technology developed and the value of content flowing through the network increased, ISPs developed the ability and the incentives to discriminate internet traffic to get a cut of the spoils. We need rules to keep ISPs focused on what they do best: making access cheaper and faster.

"Historically, under both Republican and Democratic leadership, the FCC has worked to ensure net neutrality principles were respected, sending a message to ISPs that they could not engage in blocking content, throttling site speed, or charging for content to be prioritised. In 2015, these net neutrality principles were formalised via strong new rules to ensure the internet remained free and open.

"But now the new FCC leadership is trying to demolish these protections.

"The FCC's proposal, if voted through on December 14, would open the door for ISPs to act on short-term incentives and upend the internet as we know it."

And of course the FCC did indeed vote the proposal through.

The Netherlands

But opposition to the Pai ruling is all speculative, right? We don't yet know what the consequences will be of undoing net neutrality, do we? Actually, we do.

"In the Netherlands, the telecommunication company KPN was losing a lot of money because so many people were using online text messaging apps like WhatsApp. People weren't paying for the traditional, expensive text messages over a cellular network.

"KPN said, 'We are losing all this money. That's not sustainable. We will switch to plans that do not include the right to use online text messaging.' And if someone wanted to use online text messaging, they had to buy an online text messaging option. That got such a huge outcry in the Netherlands that they became the first country to adopt a net neutrality law that banned blocking and slowing down and speeding up websites." [Eric Allen Been, *VOX*, December 14, 2017]

But surely this made KPN unprofitable, right? Which would support the argument Pai put forward. But KPN seems to be doing just fine [U14].

Unpopular

So is this a partisan issue? Like so much in the US today?

It's certainly a *political* issue, with political officeholders lining up on one side or the other depending on their party. But is net neutrality really a partisan issue?

"Net neutrality isn't a partisan issue. Polls consistently show that Americans, whether Republican or Democrat, support the current net neutrality protections. A poll that was published in July shows that 77 percent of Americans support the current protections at the FCC — and that 73 percent of Republicans, 80 percent of Democrats, and 76 percent of independents want to keep the current protections." [Eric Allen Been, *VOX*, December 14, 2017]

Arguments can be advanced in opposition to net neutrality as defined by Wu and implemented by the Obama FCC. Wu himself articulated what are probably the most cogent objections. One need not agree that he fully answered them. But the defense of Pai and his concurring FCC commissioners for reversing net neutrality is not terribly convincing.

"Is there any evidence that net neutrality regulations actually hobbled investment and innovation?" *Snopes* asks, and answers, "[t]he chairman of the FCC says yes, but the financial statements of some of the United States' largest ISPs indicate otherwise; nor is it a given that repealing the regulations will stimulate investment and innovation as claimed." *Snopes* cites a *Wired* article [U15] that says "the FCC cites industry-funded studies concluding that investment in internet infrastructure declined 3 percent in 2015 and another 2 percent in 2016. ... But the nation's largest internet provider actually increased its spending during this period, as did several other companies. Others cut spending, but said the drops stemmed from completion of longer-term plans." [Clint Finley, "The FCC Says Net Neutrality Cripples Investment. That's Not True," *Wired*, December 8, 2017]

Ultimately, there is reason to believe that the reversal will not survive judicial scrutiny.

What Will Happen?

Tim Wu thinks the courts will ultimately save net neutrality.

"Wu is right that the FCC's draft order is based on a shaky legal foundation and is likely to be struck down in court." [Eric Allen Been, *VOX*, December 14, 2017]

Legal precedent supports Wu's view.

"[G]overnment agencies are not free to abruptly reverse longstanding rules on which many have relied without a good reason, such as a change in factual circumstances. A mere change in FCC ideology isn't enough." [*NYT, November, 2017* [U16]]

Lawsuits are being filed. The issue will be adjudicated. And if precedent prevails, the undoing of net neutrality will itself be undone.

About the Author
Michael Swaine is the editor of *PragPub*.

External resources referenced in this article:

[U1] http://www.law.columbia.edu/news/2017/11/net-neutrality-Tim-Wu-FCC

[U2] http://www.timwu.org/OriginalNNProposal.pdf

[U3] https://www.amazon.com/Who-Controls-Internet-Illusions-Borderless/dp/0195340647

[U4] https://www.amazon.com/Master-Switch-Rise-Information-Empires/dp/0307390993

[U5] https://www.amazon.com/Attention-Merchants-Scramble-Inside-Heads

[U6] http://www.timwu.org/network_neutrality.html

[U7] http://www.timwu.org/OriginalNNProposal.pdf

[U8] https://apps.fcc.gov/edocs_public/attachmatch/FCC-15-24A1.pdf

[U9] http://www.aljazeera.com/news/2017/12/fcc-votes-net-neutrality-171214164419039.html

[U10] http://lj.libraryjournal.com/2017/12/advocacy/net-neutrality-fight-likely-move-courts-congress/

[U11] http://lj.libraryjournal.com/2017/12/advocacy/net-neutrality-fight-likely-move-courts-congress/

[U12] https://gizmodo.com/read-the-dissenting-opinion-of-the-fcc-commissioner-try-1821290547

[U13] http://transition.fcc.gov/Daily_Releases/Daily_Business/2015/db0226/DOC-332260A1.pdf

[U14] http://ir.kpn.com/

[U15] https://www.wired.com/story/the-fcc-says-net-neutrality-cripples-investment-thats-not-true/

[U16] http://www.law.columbia.edu/news/2017/11/net-neutrality-Tim-Wu-FCC

Antonio on Books

New Tech Books

by Antonio Cangiano

Antonio Cangiano is the author of the excellent Technical Blogging [U1] and you can subscribe to his reports on new books in technology and other fields here [U2].

The web continues to evolve rapidly, with new browser features enabling developers to devise novel ways to deliver applications to mobile devices.

Progressive applications try to bridge the gap between web applications and native mobile applications, attempting to maintain the benefits of both: the ease of development and deployment of the former, and the offline capabilities of the latter.

Google is a major proponent of progressive web applications, so much so that they are deprecating Chrome Apps in favor of PWA even on the desktop.

It's definitely a topic worth exploring if you are a web developer. This month's pick, *Progressive Web Applications* by Dean Hume, is a great resource to get you started.

Our Staff Pick: *Progressive Web Apps* [U3] • By *Dean Alan Hume* • ISBN: *1617294586* • Publisher: *Manning Publications* • Publication date: *December 21, 2017* • Binding: *Paperback* • Estimated price: *$29.96*

Deep Learning with Python [U4] • By *Francois Chollet* • ISBN: *1617294438* • Publisher: *Manning Publications* • Publication date: *December 22, 2017* • Binding: *Paperback* • Estimated price: *$46.00*

The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams [U5] • By *Kurt Bittner, Patricia Kong, Dave West* • ISBN: *0134682661* • Publisher: *Addison-Wesley Professional* • Publication date: *December 27, 2017* • Binding: *Paperback* • Estimated price: *$24.10*

Designing Connected Content: Plan and Model Digital Products for Today and Tomorrow [U6] • By *Carrie Hane, Mike Atherton* • ISBN: *0134763386* • Publisher: *New Riders* • Publication date: *December 25, 2017* • Binding: *Paperback* • Estimated price: *$25.56*

Computer Architecture, Sixth Edition: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design) [U7] • By *John L. Hennessy, David A. Patterson* • ISBN: *0128119055* • Publisher: *Morgan Kaufmann* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$109.95*

Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python [U8] • By *Abhishek Nandy, Manisha Biswas* • ISBN: *1484232844* • Publisher: *Apress* • Publication date: *December 8, 2017* • Binding: *Paperback* • Estimated price: *$39.83*

Parsing with Perl 6 Regexes and Grammars: A Recursive Descent into Parsing [U9] • By *Moritz Lenz* • ISBN: *1484232275* • Publisher: *Apress* • Publication date: *December 11, 2017* • Binding: *Paperback* • Estimated price: *$16.50*

Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced Artificial Intelligence in Python [U10] • By *Santanu Pattanayak* • ISBN: *1484230957* • Publisher: *Apress* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$38.80*

Build Better Chatbots: A Complete Guide to Getting Started with Chatbots [U11] • By *Rashid Khan, Anik Das* • ISBN: *1484231104* • Publisher: *Apress* • Publication date: *December 15, 2017* • Binding: *Paperback* • Estimated price: *$26.73*

Windows 10 Development with XAML and C# 7 [U12] • By *Jesse Liberty, Jon Galloway, Philip Japikse, Jonathan Hartwell* • ISBN: *1484229339* • Publisher: *Apress* • Publication date: *December 11, 2017* • Binding: *Paperback* • Estimated price: *$23.62*

iOS Development with Swift [U13] • By *Craig Grummit* • ISBN: *1617294071* • Publisher: *Manning Publications* • Publication date: *December 1, 2017* • Binding: *Paperback* • Estimated price: *$43.49*

Practical Programming: An Introduction to Computer Science Using Python 3.6 [U14] • By *Paul Gries, Jennifer Campbell, Jason Montojo* • ISBN: *1680502689* • Publisher: *Pragmatic Bookshelf* • Publication date: *December 16, 2017* • Binding: *Paperback* • Estimated price: *$45.61*

Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber [U15] • By *Moises Macero* • ISBN: *1484231643* • Publisher: *Apress* • Publication date: *December 10, 2017* • Binding: *Paperback* • Estimated price: *$20.35*

Physically Based Shader Development for Unity 2017: Develop Custom Lighting Systems [U16] • By *Claudia Doppioslash* • ISBN: *1484233085* • Publisher: *Apress* • Publication date: *December 6, 2017* • Binding: *Paperback* • Estimated price: *$49.99*

Hello Scratch!: Learn to Program by Making Arcade Games [U17] • By *Gabriel Ford, Melissa Ford, Sadie Ford* • ISBN: *161729425X* • Publisher: *Manning Publications* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$20.08*

Java EE Web Application Primer: Building Bullhorn: A Messaging App with JSP, Servlets, JavaScript, Bootstrap and Oracle [U18] • By *Dave Wolf, A.J. Henley* • ISBN: *1484231945* • Publisher: *Apress* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$16.46*

wxPython Recipes: A Problem - Solution Approach [U19] • By *Mike Driscoll* • ISBN: *1484232364* • Publisher: *Apress* • Publication date: *December 13, 2017* • Binding: *Paperback* • Estimated price: *$35.29*

Developing Bots with Microsoft Bots Framework: Create Intelligent Bots using MS Bot Framework and Azure Cognitive Services [U20] • By *Srikanth Machiraju, Ritesh Modi* • ISBN: *1484233115* • Publisher: *Apress* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$33.95*

Pro Power BI Desktop [U21] • By *Adam Aspin* • ISBN: *1484232097* • Publisher: *Apress* • Publication date: *December 9, 2017* • Binding: *Paperback* • Estimated price: *$45.52*

Jumpstarting the Raspberry Pi Zero W: Control the World Around You with a $10 Computer [U22] • By *Akkana Peck* • ISBN: *1680454560* • Publisher: *Maker Media, Inc* • Publication date: *December 28, 2017* • Binding: *Paperback* • Estimated price: *$8.99*

Go in 24 Hours, Sams Teach Yourself: Next Generation Systems Programming with Golang [U23] • By *George Ornbo* • ISBN: *0672338033* • Publisher: *Sams Publishing* • Publication date: *December 18, 2017* • Binding: *Paperback* • Estimated price: *$25.09*

Introduction to Python for Engineers and Scientists: Open Source Solutions for Numerical Computation [U24] • By *Sandeep Nagar* • ISBN: *1484232038* • Publisher: *Apress* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$17.64*

The Tao of Microservices [U25] • By *Richard Rodger* • ISBN: *1617293148* • Publisher: *Manning Publications* • Publication date: *December 31, 2017* • Binding: *Paperback* • Estimated price: *$31.99*

C# Programming for Absolute Beginners [U26] • By *Radek Vystav* • ISBN: *1484233174* • Publisher: *Apress* • Publication date: *December 4, 2017* • Binding: *Paperback* • Estimated price: *$21.48*

XML and JSON Recipes for SQL Server: A Problem-Solution Approach [U27] • By *Alex Grinberg* • ISBN: *1484231163* • Publisher: *Apress* • Publication date: *December 19, 2017* • Binding: *Paperback* • Estimated price: *$34.99*

Jumpstarting the Arduino 101: Interacting with a Computer That Learns [U28] • By *Yining Shi, Sagar Mohite* • ISBN: *1680454552* • Publisher: *Maker Media, Inc* • Publication date: *December 28, 2017* • Binding: *Paperback* • Estimated price: *$9.47*

Interactive C#: Fundamentals, Core Concepts and Patterns [U29] • By *Vaskaran Sarcar* • ISBN: *1484233387* • Publisher: *Apress* • Publication date: *December 13, 2017* • Binding: *Paperback* • Estimated price: *$30.49*

PySpark Recipes: A Problem-Solution Approach with PySpark2 [U30] • By *Raju Kumar Mishra* • ISBN: *1484231406* • Publisher: *Apress* • Publication date: *December 10, 2017* • Binding: *Paperback* • Estimated price: *$34.81*

Using MVVM Light with your Xamarin Apps [U31] • By *Paul Johnson* • ISBN: *1484224744* • Publisher: *Apress* • Publication date: *December 9, 2017* • Binding: *Paperback* • Estimated price: *$32.61*

Introducing JavaScript Game Development: Build a 2D Game from the Ground Up [U32] • By *Graeme Stuart* • ISBN: *1484232518* • Publisher: *Apress* • Publication date: *December 7, 2017* • Binding: *Paperback* • Estimated price: *$17.15*

The BoB Pages

The Latest from The Pragmatic Bookshelf and The Prose Garden

by BoB Crew

What's new at the Pragmatic Bookshelf and The Prose Garden. Top-selling books. Author events. Back issues. Solution to puzzle.

BoB Crew here, with all the Back of the Book goodness. See, *PragPub* is brought to you by The Prose Garden in cooperation with the The Pragmatic Bookshelf, and this BoB department (short for Back of Book) right here contains a rundown on what's happening on the Shelf and in the Garden. A reader service, courtesy of yours truly, BoB. Enjoy!

Who's Where When

Here's what the Pragmatic Bookshelf authors are up to in the near future:

2018-01-02 **Learn to build a daily writing habit so you can write articles, blog posts, whatever you need to enhance your business and reputation**
Johanna Rothman (author of Behind Closed Doors [U1] , Manage It! [U2] , Hiring Geeks That Fit [U3] , Manage Your Job Search [U4] , Predicting the Unpredictable [U5] , Manage Your Project Portfolio, Second Edition [U6] , Agile and Lean Program Management [U7] , and Create Your Successful Agile Project [U8])
Non-Fiction Writing Workshop to Enhance Your Business (online workshop) [U9]

2018-01-03 **Take your writing to the next level. Learn how to have several projects in progress, finish them and publish them.**
Johanna Rothman
Secrets of Successful Non-Fiction Writers (Workshop 2) [U10]

2018-02-05 **Talk 1: What I learned from an Olympic sports career (so you don't have to). Talk 2: Creating Great Teams – How Self-Selection Lets People Excel**
Sandy Mamoli (author of Creating Great Teams [U11])
OOP: 2 talks [U12]

2018-03-21 **Build Web Apps with Elm Workshop**
Jeremy Fairbank (author of Programming Elm [U13])
JazzCon, New Orleans, LA [U14]

Our Back Pages

At The Prose Garden, we plant, water, fertilize, prune, and arrange your prose (and ours) into what we hope are beautiful bouquets of insight and knowledge. If you have a budding interest in sharing some technical insight of your own, please contact us at mike@swaine.com [U15] or nancy@swaine.com [U16]. We are always on the lookout for game-changing, enlightening, pragmatic articles and our audience is, too. In addition to publishing our monthly magazine, we also edit books, blogs, and articles.

What's Hot

Here are the best-selling Pragmatic Bookshelf titles right now:

1	NEW	Programming Elm
2	NEW	Java by Comparison
3	3	Domain Modeling Made Functional

4	1	Modern Vim
5	8	Agile Web Development with Rails 5.1
6	4	Release It! Second Edition
7	2	Adopting Elixir
8	5	Design It!
9	NEW	Node.js 8 the Right Way
10	7	Craft GraphQL APIs in Elixir with Absinthe
11	NEW	3D Game Programming for Kids, Second Edition

Solution to Sudoku

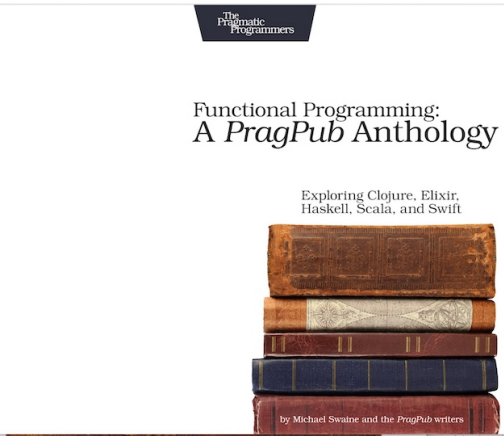
Here's the solution to the Sudoku that appeared earlier in this issue:

G	R	S	L	E	N	A	T	I
I	L	A	R	T	S	N	G	E
E	T	N	A	I	G	L	R	S
S	E	L	T	G	I	R	N	A
T	G	R	N	S	A	I	E	L
N	A	I	E	L	R	G	S	T
A	N	T	S	R	L	E	I	G
L	S	G	I	N	E	T	A	R
R	I	E	G	A	T	S	L	N

We said you could rearrange the nine letters to form a description of the most difficult problem, something encountered in calculus courses, and an obsession of Pythagoras. And you can: the most difficult problem is the gnarliest, integrals are encountered in calculus courses, and Pythagoras had a thing for triangles. So: GNARLIEST, INTEGRALS, and TRIANGLES.

To really be in the know about all things Pragmatic, you need to subscribe to the newsletter. It'll keep you in the loop, it's a fun read, and it's free. All you need to do is create an account on pragprog.com [U17] (email address and password is all it takes) and select the checkbox to receive the weekly newsletter.

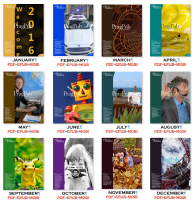
Functional Programming: A PragPub Anthology



If you like *PragPub*, I bet you'll like *Functional Programming: A PragPub Anthology* [U18]. We scrutinized all the articles on functional programming from the past eight years of *PragPub* and put together a book that looks at functional programming from the perspective of five languages: Clojure, Elixir, Haskell, Scala, and Swift. Yyou'll explore functional thinking and functional style and idioms across languages. Led by expert guides, you'll discover the distinct strengths and approaches of each language and learn which best suits your needs.

"This book is an amazing buffet of programming delicacies, all arranged around the vital and compelling theme of functional programming. These authors are great teachers. With so many expert voices presenting different aspects of the topic, this book is like *Beautiful Code* for the functional-curious." — Ian Dees

Order it from The Pragmatic Bookshelf here [U19].



Back Issues

If you like this issue of *PragPub*, there are more where it came from.

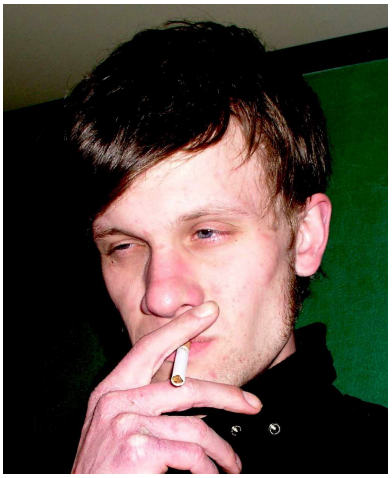
Get all 12 issues of a year of *PragPub* in all three formats — pdf, epub, and mobi — plus a downloadable index to the articles, in the Back Issue bundles. Twelve bucks for a full year of PragPub (2013, 2014, 2016, or 2016). And of course you can purchase individual back issues and you can subscribe to be sure of getting future issues. All at The Prose Garden [U20].

Shady Illuminations

Predictions for 2018

by John Shade

John fearlessly predicts the major tech developments of 2018.



I'm looking for the Winklevoss twins to make their big move this year. After turning a fraction of a sixty-five million dollar payoff from their old college friend Mark Zuckerberg into a billion dollars last year by buying bitcoins, the duo of entrepreneurs and competitive rowers look poised to blow their bundle on something incredibly chic.

The cloud will continue its rise, but look for category-cracking change. 2018 will usher in the era of Cloud 3.0. Cloud 1.0, of course, was about computation, Cloud 2.0 was about data, and Cloud 3.0 will be about the other thing.

A new primary color will be discovered and patented. Knock-off generic version will emerge later in the year, but you'll have to fly to Canada to get them.

In a breakthrough in medical science, Richard Branson and Elon Musk will merge into one powerful trans-human entity.

Flashmobs will transition from performance art to a replacement for temp services.

Soon noone will speak of buying a "phone" or a "camera" any more than you would speak of buying a device called a "clock" or a "calculator." Or a "computer." We'll buy an amorphous device with a chip for everything inside and we'll pay to have various features turned on. Whatever As A Service.

Facebook will change its user interface.

Augmented reality will see a big boom in popularity as regular reality become increasingly bizarre. New words will be invented to describe the audio and video analogs of photoshopping. New text-analysis algorithms will be developed to tell which of your deeply-held opinions were implanted by Russian Facebook bots. Someone will steal your identity and your phone will refuse to recognize you. You will seek comfort in robotic arms.

Everything will be 50% off all year long.

The release of the JFK files will finally resolve all questions about the assassination. At long last we will know the answers to the troubling questions regarding the Magic Bullet, the Grassy Knoll, and Ted Cruz's father.

2018 will be The Year of Unix.

About the Author

John Shade was born in Montreux, Switzerland, on a cloudy day in 1962. Subsequent internment in a series of obscure institutions of ostensibly higher learning did nothing to brighten his outlook. He predicts that 2018 will be almost as much fun as 2017.