Issue 045: Requirements

Adrian Kosmaczewski

June 6th, 2022

Welcome to the forty-fifth issue of *De Programmatica Ipsum*, dedicated to the subject of *Requirements*. In this edition:

- As an author, Graham explains requirements gathering[1] so that teams stop making assumptions.
- Adrian tries to reconcile agile practitioners with requirement gathering[2].
- In the Library section[3], Graham reviews a growing number of worker union-related titles[4].

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

Cover photo by Dylan Gillis[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/the-other-side-of-the-card/
[2] https://deprogrammaticaipsum.com/on-agile-requirements/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/workers-of-the-digital-world/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@dylandgillis?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/collaboration?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

The Other Side Of The Card

Graham Lee

June 6th, 2022

If you ask many developers what a "user story" is, you will be told that it is a software requirement expressed using a prescribed formula:

> As a *category of person*, I want to *do a thing* so that *some goal is attained*.

This is very close to the formula of a Hoare Triple[1] that is used to make assertions about the ways in which computer programs work: Given that I am this category of person, when the software lets me do this thing, then I will attain a certain goal. It is perhaps no surprise that software engineers use the same pattern for requirements and for tests.

But this is not the whole truth about the user story. Before we go into what is missing, let us take a moment to understand why this partial story is so popular. Many software engineers do not engage with the broad "software engineering literature" very much: through the act of reading this magazine you are placing yourself at the pinnacle of software engineering curiosity!

So most people who know what a user story is were told what a user story is. But we still need an on-ramp for our partial explanation: we need to know why people think it is that sentence above and no more. A great book on the topic is User Stories Applied[2] by Mike Cohn. If you are not sure about this whole user stories thing, you may not want to commit to buying the book, but that is OK, you can sign up and download a sample chapter from Mountain Goat Software.

That chapter happens to be the one that describes the one-sentence formulation above. So here is my theory. Some people heard that the Agile folks were using this "user story" thing

[1] https://deprogrammaticaipsum.com/how-to-reason-about-mutable-state/
[2] https://www.mountaingoatsoftware.com/books/user-stories-applied

that is way easier than use cases. Curious, they read the sample chapter from Cohn's book, which told them about this one-sentence formulation. They decided to try it out. "What is that?" "It is a user story. It is an easy way to capture requirements. I learned about it from the book, *User Stories Applied* by Mike Cohn."

On with the show: my point is that there is a lot more to capture about software requirements than the headline, if you want to make good software that satisfies your customers. This is the stuff that goes on the back of the index card: the stuff you find out when you discuss the title with anyone.

I say "index card" because that is how this whole agile thing was designed. It was at a time when a lot of processes, including those in software companies, had not been computerised: indeed it was in many ways a protest against the automation we saw in Computer-Aided Software Engineering (CASE) tools. Agile software development is done in one room, with lots of talking, and high-visibility, highly-mutable information captures: walls full of post-it notes and index cards.

It is important to acknowledge the dual nature of back of the index card information. It is of conversations, but it should not be confined to conversation. It is part of the formal documentation of the project, but will not be uncovered if restricted to formal requirements-gathering ceremony. It exists in the liminal space between conversation and meeting, between process and chaos, between the TPS report and the back of the cigarette packet.

I have previously written about the requirements trifecta[3]: companies do well when their products balance the company's visionary direction; market expectation; and technical reality. So the back of the index card tracks all of that information. Is this story linked to any others in ways that are important for the vision to make sense? Are there any particular customers or leads who would benefit from the value realised by this story, and what effect does knowing that have on priorities? Is there technical work that needs to be done to realise this story, and how does that affect delivery estimates?

And then there is all the additional detail that did not fit in the one-sentence description on the front of the card. For example, Atlassian give this example of a user story[4]:

> As Max, I want to invite my friends, so we can enjoy this service together.

I have questions! Some of them:

- Who is Max?
- Why do we care what Max wants?
- Does Max want to choose which friends they invite?
- How many friends does Max have?
- Over what timespan will these invitations get sent?
- Over what medium will these invitations get sent?
- What will Max's friends experience when they receive the invitation?
- How will Max's friends interact to accept or decline the invitation?
- What is the "together" nature of this service?
- What is this service?
- What operational costs can we accept in deploying this invitation capability?
- Are there any privacy, security, or regulatory implications to consider when building this invitation feature?

[3] https://www.sicpers.info/2022/05/the-requirements-trifecta/

[4] https://www.atlassian.com/agile/project-management/user-stories

- How will invitations fit into the existing code?
- Can Max retract an invitation?
- What happens in Max leaves the service before one of their invitations is accepted?
- Does Max represent a broader population of users?

Any answers to those and similar questions that do not get written down somewhere the whole team can see and discuss will become assumptions. Did we all make the same assumptions? Do you feel lucky? Do ya?

Cover photo by Kate Trysh[5] on Unsplash[6].

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

[6]https://unsplash.com/s/photos/index-card?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

On Agile Requirements

Adrian Kosmaczewski

June 6th, 2022

We keep talking and writing blog posts, recording podcasts, and publishing an edition of this magazine dedicated to requirements because most software engineers and managers have a conflicting relationship with them. Engineers and managers will complain that writing requirements down (even in small cards) is against the ethos of Agile (spoiler alert: it is not), while the same engineers and managers will blame project failure to, guess what? Insufficient or incomplete requirements. Shocker.

Let us see what some authors have to say about requirements gathering.

Ian Sommerville's "Software Engineering[1]" textbook dedicates a section to requirements gathering, establishing a counterpoint between extremes. On one corner, we have IEEE/ANSI 830/1998[2], the heavyweight champion in the requirements category. We have Kent Beck's[3] agile cards and flyweight category champion on the other corner. Let the match begin!

Fred Brooks' is very clear on page 199 of his classic book "The Mythical Man-Month"[4]:

> The hardest single part of building software is deciding what to build.

So much for opening Visual Studio Code and having GitHub copilot write code for us right away, damn.

[1] https://www.iansommerville.com/software-engineering-book/
[2] http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf
[3] https://deprogrammaticaipsum.com/kent-beck/
[4] https://en.wikipedia.org/wiki/The_Mythical_Man-Month

Steve McConnell[5] not only agrees with Fred Brooks' assessment (page 113 of "Software Project Survival Guide") but also talks about requirements gathering on page 25 of "Code Complete"–that is, really, really, *really* at the beginning of the 900 pages volume. According to him, gathering requirements reduces project risk. The risk of what? The risk of failure, of course, which in software engineering has been a rampant and pervasive problem since, well, always.

Software quality[6] screams for requirements:

> Software quality is a complex and widely studied topic, but it is not hard to provide a simple definition: quality means that the software does the right things, and does them right. These "things" that a software system does are known as its requirements. Not surprisingly, requirements engineering is a core area of software engineering.

(Bruel, Jean-Michel, Sophie Ebersold, Florian Galinier, Alexandr Naumchev, Manuel Mazzara, and Bertrand Meyer. 2020. "The Role of Formalism in System Requirements (Full Version)." ArXiv:1911.02564 [Cs], April. http://arxiv.org/abs/1911.02564.)

Speaking about Monsieur Meyer[7], he explained in his book "Agile!" that the agile school rejects the idea of upfront requirements, which leads to downright professional malpractice (page 34) fueled by one critical component of ninja-intensive teams:

> The problem here is dogmatism.

Last but not least, let us not forget that "have a spec" is point 7 in the 22-year-old Joel Test[8].

Why are requirements so necessary? And why is it that so few software practitioners take the time to get them right? Why do teams not write[9] requirements down?

As with many other things in our industry, there are many options between the IEEE standard and Kent Beck. Let us mention two of them, both in the Pragmatic Bookshelf. "Domain Modeling Made Functional"[10] by Scott Wlaschin advocates using the F# programming language[11] to gather and document requirements and turn them into a solid foundation for the final implementation. If F# is not your cup of tea, maybe Ruby[12] is? In that case, for the past 15 years, the various editions of "Agile Web Development with Rails"[13] by Sam Ruby, Dave Thomas, and David Heinemeier Hansson have always included a walkthrough describing the interaction with a fictional customer, leading to the creation of a new software product. The idea behind both books is the realization that spending time thinking about the problem is not a sin committed against a holy Agile Manifesto[14].

Both Meyer & McConnell agree on a critical misconception of our industry: there is no conflict between "being agile" and gathering and documenting requirements. As pointed out by McConnell, the issue is not to have fewer requirements but to defer them to a later stage of the process when the cone of uncertainty of your project starts shrinking.

---

[5] https://deprogrammaticaipsum.com/steve-mcconnell/
[6] https://deprogrammaticaipsum.com/issue-2-quality/
[7] https://deprogrammaticaipsum.com/bertrand-meyer/
[8] https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/
[9] https://deprogrammaticaipsum.com/on-the-aversion-to-writing/
[10] https://pragprog.com/titles/swdddf/domain-modeling-made-functional/
[11] https://fsharp.org/
[12] https://www.ruby-lang.org/en/
[13] https://pragprog.com/titles/rails7/agile-web-development-with-rails-7/
[14] https://agilemanifesto.org/

Being an engineer requires the evaluation of all forces in a project to find the proper solution for the problem at hand. Requirements are a significant step towards adequate software engineering. Before going down the dogma[15] drain, we should consider if cards are better suited for your project than the IEEE standard. Requirements can take various forms, and your project deserves at least one of them at any given time.

Cover photo by Markus Winkler[16] on Unsplash[17].

[15] https://deprogrammaticaipsum.com/you-are-doing-it-wrong/

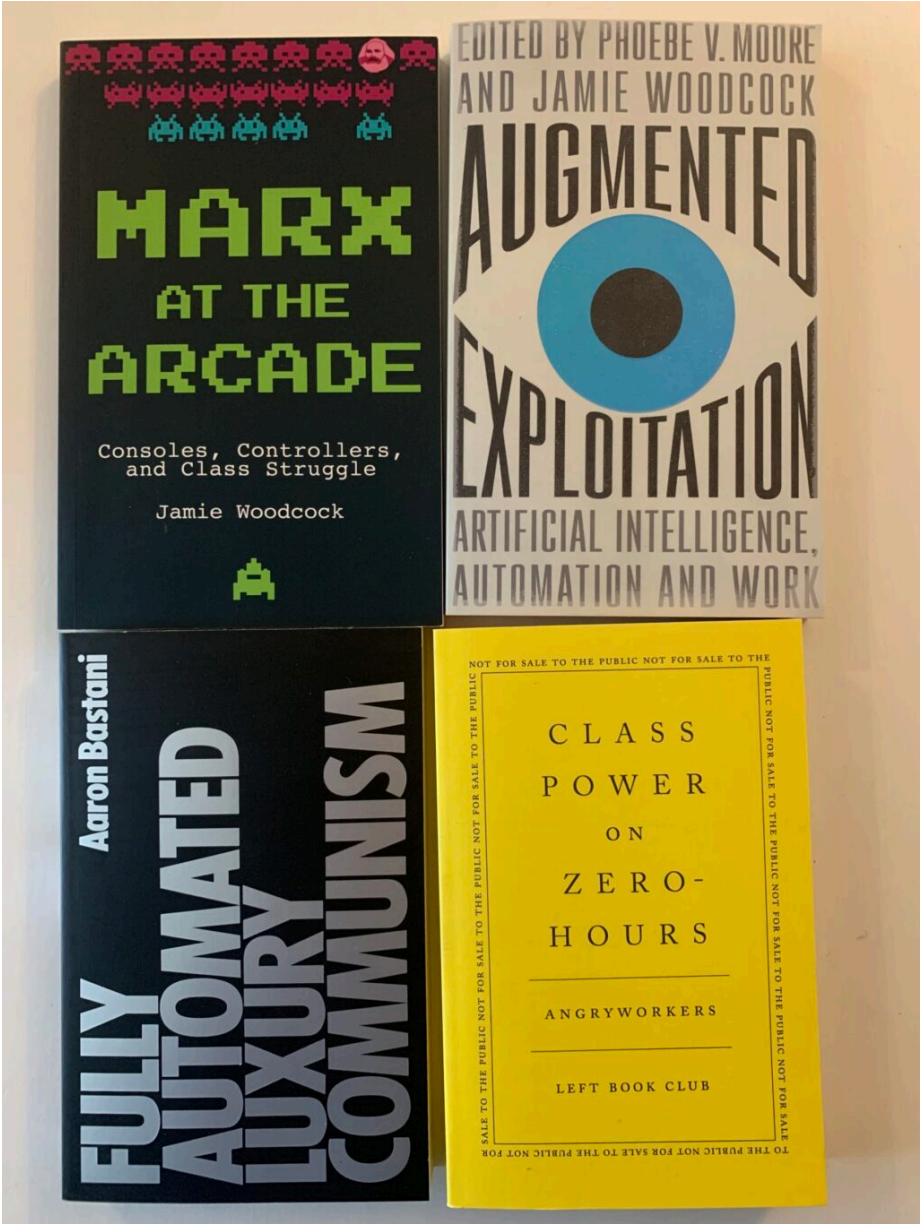
[16] https://unsplash.com/@markuswinkler?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

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

Workers Of The (Digital) World

Graham Lee

June 6th, 2022



They say that software is eating the world[1]. They also say, or rather they say that Jean-Jacques Rousseau said:

> Quand le peuple n'aura plus rien à manger, il mangera le riche.

[1] https://a16z.com/2011/08/20/why-software-is-eating-the-world/

A lot of software and other technological development is done in a kind of technocratic homeostasis: technologists believe that they are amoral actors simply advancing the boundaries of human knowledge, and that it is up to others to decide how their inventions and developments are applied. They channel Tom Lehrer's fictionalised version of a prominent Nazi party member and SS officer:

> "Once the rockets go up, who cares where they come down? That's not my department," says Wernher von Braun.

Of course, this is a complete nonsense. These supposedly consequence-free technologists and rational actors are taking money from those who profit from the very applications of their work that they claim to be indifferent to. Reality is exposed: it is not that they are contributing to humanitarian progress; they are contributing to their own access to foosball tables and at-desk massage perks.

We have previously covered why it is important that technology workers understand and participate in the trade union movement[2]. Now let us learn from those whose work is impacted by technology.

Discounting audiobook editions of printed books, this is the first audiophonic entry into the *De Programmatica Ipsum* library: the podcast Peak Salvation[3]. Former Facebook engineering director Philip Su takes a job at an Amazon fulfilment centre during the busy "peak" season between Thanksgiving and Christmas. If you have never worked a manual job—or even if like me you have but it was back when you could expect your manager to be a human being, not an app—take a listen to this short podcast and understand the human cost of getting you those oh-so-important same day deliveries on weird HDMI adapters you left at the gym.

Then find out how low-paid, precarious, gig economy workers can organise and fight back in Class Power on Zero Hours[4]. Gig economy workers have had some amazing successes in the UK, despite being managed remotely via app and not having the traditional shop floors where they can meet their union comrades and stewards.

Just Eat drivers in Sheffield and Thanet (actually working as "independent couriers" for Stuart, a company owned by the French postal service) strike for improved pay and conditions[5]. The new trade union IWGB has achieved significant benefits for Deliveroo drivers[6]. Meanwhile the AngryWorkers collective document successes among their members in West London.

Want to understand how precarious employment, workplace surveillance, and artificial intelligence are combined to increase exploitation in the workplace, reduce wages, and reduce the possibility of workers collaborating to improve their conditions: but also what workers can do about it? An excellent collection of essays on these subjects is Augmented Exploitation[7].

From questions like "who is the smart worker?" (if I am being tracked by a management AI, will I do good work or will I do whatever requires least effort and gets marked positively by the algorithm?) to "what can we do about it?" (Deliveroo riders worked collectively to

[2] https://deprogrammaticaipsum.com/issue-42-trade-unions/

[3] https://peaksalvation.com

[4] https://pmpress.org.uk/product/class-power-on-zero-hours/

[5] https://morningstaronline.co.uk/article/b/just-eat-delivery-workers-to-stage-rally-and-motorcade-against-pay-cuts-in-sheffield

[6] https://iwgb.org.uk/en/post/deliveroo-gmb

[7] https://www.plutobooks.com/9780745343495/augmented-exploitation/

first uncover, and then subvert, the algorithmic rules governing their work), Augmented Exploitation examines and lays bare the experience of working in a society constructed by consequence-ignorant Javascript coders.

Then, after so much focus on work and workers' conditions, how about relaxing in front of a nice video game? Jamie Woodcock has you covered, with Marx at the Arcade[8], a book detailing the part computer games play in modern capitalism.

From their reinforcement of modern ideas of militarism and imperialism in popular culture (how long do you think will be a tasteful wait before the Mariupol edition of *Counterstrike* comes out, and do you think they are already working on the DLC?), through the way the capitalist system rejects self-criticism in the name of "protecting users" (good luck getting a copy of Phone Story[9] for iPhone through official channels), to the way in which companies will stand up for their harassing—but profitable—users rather than their troublesome and harassed employees, even in a post-GamerGate world: this is the place to go.

Lest we get lost in the doldrums of the current system and its exploitation of the workers, Aaron Bastani provides a vision of the way out. In Fully Automated Luxury Communism: a Manifesto[10], he points out that not only is it possible to create a world in which technology works for us all in the way that Jeff Bezos imagines is only possible by zero-hours workers and for multi-billionaires, but that most of the technology needed to realise this post-scarcity future is already here.

Of course, successfully deploying this technology in a way that reverts the accelerating inequalities of the world requires massive political and systematic changes. Bastani is clear that these are achievable, but complex: he explains why facile solutions like Universal Basic Income are unlikely to fix anything and may even break things further.

Digital workers of the world, unite! You have nothing to lose but your blockchains.

[8] https://www.jamiewoodcock.net/marx-at-the-arcade/

[9] http://phonestory.org/banned.html

[10] https://www.theguardian.com/sustainable-business/2015/mar/18/fully-automated-luxury-communism-robots-employment