Table of Contents

"The Rollout: A Novel about Leadership and Building a Lean-Agile Enterprise with SAFe" by Alex Yakyma

If you ever wanted to understand why people talk about scaling in a different way that to traditional agile thinking, this is the book for you. This is a business novel format book that brings home the basics of SAFe to people.

Starting with the premise that a lot of implementations of agile work well for teams, but are perceived as limited (or even a failure) when considered in a larger organization, the book takes you on a journey (via novel and characters) from an organization that had multiple agile teams to an approach where there is an enterprise and portfolio view of how we implement work through a team of teams.

What I liked in particular about this book is the way it brings some SAFe concepts down to earth. No longer are we talking about an "Agile Release Train" but rather we are understanding that the reason an ART works is because there is they "plan together, execute synchronously, and review the results together". This understanding helps people understand how they would apply the concept in non-traditional environments.

I also liked the explanation of themes, epics, and capabilities. It sometimes seems like these concepts have been blown up to be intentionally confusing. But the explanation in this book (themes as filters for epics, epics being a longer term investment necessitating a business case to allow you to make (appropriate) centralized decisions in a largely de-centralized environment, if your value stream is too big to line up with an ART, split it by capabilities) helps keep the conversation grounded.

Recommended for any business leader who has done "agile" and is wondering "what happened" and "whats now".

Some interesting observations:

- On the fractal nature of an agile implementation and why we should do PI planning face-to-face: "So why is it that we acknowledge face-to-face communication as the best way of conveying information at the team level, but at the program level this simple, yet absolutely fundamental tenet of Agile is demonstrably ignored"
- On why agile at team level is not enough: "The new process deployed on top of a traditional management mindset is like an old ship: patched everywhere you can imagine. She seems watertight in calm seas, but as soon as the slightest turn in the weather hits the sails, she will wreck on the reef of organizational impediments. And when all patches suddenly come off—one after another—nothing can save her. She goes down."
- On why we do PI Planning Event: "These two days provide critical alignment in terms of what and how the teams should be building in the PI. You're right: PI planning is an investment, but so is the PI itself. We have significant problems building new functionality, as we all know perfectly well. That's because the teams don't have the ability to consistently manage dependencies and risks."
- On observable milestones "Inability to demonstrate PI outcomes is a significant dysfunction for an Agile Release Train, and must be addressed immediately and thoroughly without any concessions

or tradeoffs." Also "Nothing strengthens trust between teams and business owners better than the PI System Demo." This leads to the catch phrase "No demo, no numbers". If you can't demonstrate a fully integrated system, don't fool yourself with metrics. It's better to accept the defeat and focus on solving the underlying problem.

- On the role of the change agent: "A successful change agent must ensure that there's no unproductive tension between themselves and the stakeholders. Any antagonism, even latent, may inhibit the transformation and must be promptly resolved. It's up to you to make the first step."
- On why the agile approach works better than the traditional approach: "The traditional mindset progressively builds blindspots and inhibits learning. Surrounded by all kinds of numbers, we have no ability to really tell where we are in the process. Being separated into silos, we fail to learn and introspect as a whole. We need to put all the pieces of the puzzle together."
- On the Kanban board for a transformation team: "We built the board based on a simple observation: pretty much everything the Transformation Team does requires other people outside the team. We realized that we needed to reflect that aspect of the workflow in the structure of our board. Therefore, we split our 'WIP' step into three subservient steps: 'Engage' – for engaging those other stakeholders, subject matter experts, and so on; 'Execute' – for having them actually do what was required by a particular backlog item; and 'Verify' – which was for us to make sure that the result was what we wanted."
- Why normalize points: "Let me ask you this: you all use story points to express your team's velocity, but how about the velocity of the entire program?"
- Why not use some kind of "uber points" for features: "Someone has to own the 'point' and think in that point as their main 'currency'. But that someone can't be a team, because they already have their own 'system of measure'"
- On the role of an RTE: "Being selective in terms of where to apply one's major effort is vital to an RTE."
- On why teams over-commit all the time (and need management help): "The teams themselves can have flawed instincts and regularly take on more work than they can actually accomplish."
- On the goal of PI Planning: "The goal of the PI planning is not to go into depth on each story, defining acceptance criteria, etc. The goal is to understand the bigger picture view, and for that, focusing on dependencies, program risks and objectives is key."
- On why an ART works: "Plan together. Execute synchronously. Review the results together"
- On why architecture is not an isolated capability delivered from on-high: ""If you want people to inherently and naturally follow certain rules, simply stating those rules or announcing them won't get you there. People won't be able to respond the way you need them to because that information will remain superficial in their mind. What you want instead is to have them deeply care about those constraints, architectural requirements and nonfunctional considerations. It has to reside deep in their minds, not just live on paper. When I'm coding, I'm thinking, right? So if I don't think in terms of higher-level architectural implications, I will just keep contributing to the overall system decay." "And…?" said Ed, encouraging Josh to go on. "So, in order to encourage developers to care about higher-level architecture on a day-to-day basis, you can't make decisions for them. Instead, you have to make them part of the decision-making process. If they have contributed to those architectural assumptions, they will be sure to follow them and keep them in mind at all times. And I can totally understand how some of those may come solely from you Anand, but people need to be involved in the process in order to validate, to reason about it, to offer certain adjustments and so on.""
- On types of value streams: "There are two types of value streams: development and operational. Development value streams deliver software to the customer, while operational value streams offer

a certain type of service. Every operational value stream has a development value stream associated with it: the teams that develop the software systems in support of the operational workflow." SAFe's focus in on operation value streams.

- On use of capabilities: "50-125 people is an optimum ART size. If a value stream is bigger than that, it should be split into multiple trains. Trains can be organized around subsystems or capabilities. Organizing around subsystems can lead to too many hand-offs, though. Capabilities may offer a better organizational paradigm, but may lead to deterioration of the system design over time. Architectural coordination may be needed in such cases."
- On the mindset change: ""If you dig deep enough," I said, repeating almost word-for-word what Rachel had told me a year ago as she handed me a cup of 'Sleepytime', "you will eventually find that at its core, every mindset is a belief system. That's why it is so horrifically difficult to change. People don't like their core beliefs to be shaken up and gravely endangered. The harder you push, the harder they push back. What we learned was that mindset cannot be changed just by teaching people a new way to operate. They may honestly accept the new ideas, but as soon as the new ways of working cross one of their core beliefs – everything suddenly grinds to a halt.""

Want to Know More?

- ["The Rollout: A Novel about Leadership and Building a Lean-Agile Enterprise with SAFe"](#)

[Book](#), [BusinessNovel](#), [Basics](#), [SAFe](#), [Scaling](#)