Restarting Scaled Agile Development at Austrian Post

MICHAEL NIESSL, Austrian Post
CARMEN GRUBER, Austrian Post
MARTIN EDER, Austrian Post

Like many large organizations Austrian Post implemented SAFe during its first wave of agile transformation. We describe issues we experienced three years after its initial implementation that we think could be relevant for others, and how we addressed some of those issues by "restarting" and finding our own way of scaled agile development.

1. BACKGROUND

Austrian Post has a long history stretching back to the early 19th century but has existed in its current form as a publicly traded company since 1999. Throughout its history, Post has operated primarily as a Transportation and Logistics company to which IT systems have been added over time.

Starting with 2019 – and especially throughout the pandemic – the internal work force of the IT department more than doubled in size since 2017. Today, most of Post's business is relying on internally developed software products.

Not all of IT focusses on software development, though - the IT department also includes teams providing hardware services, helpdesk, running standard software (like SAP) and other services. For this experience report we will focus on the teams responsible for developing proprietary software products.

2. ABOUT US

Carmen Gruber has a background in academia, but re-focused on working with software development organizations in the past decade. She works with single scrum teams as a scrum master as well as with scaled development organization where she supports and guides the release trains (or equivalents) as an agile coach. With a large set of tools for agile teamwork and an extensive theoretical foundation in agile software development practices and scaling frameworks she is well positioned for the initiative described in this experience report.

Martin Eder has been working with Scrum teams for twelve years, the last five years as a Scrum Master and for one year also as an agile coach of a release train. He has extensive experience working with agile teams and has been instrumental in helping to formulate and implement changes to the scaled agile approach at Post.

Michael Niessl has been working in software development organizations for the past 20 years. As an engineering manager he has had the chance to establish multiple scrum teams in multiple locations from scratch. Working with growing teams gave him the opportunity to experiment with scaling frameworks and – together with his teams – find approaches that work for his environments based on elements of the common existing frameworks.

All of us joined Post about 2.5 years ago and found an already established scaled agile environment. In this experience report we describe the history of how this environment emerged, which problems we found in our area of influence and how we used our previous experiences in similar environments to address some of our major concerns.

3. HISTORY OF AGILE DEVELOPMENT AT AUSTRIAN POST

The Austrian Post began its journey toward an Agile transformation in 2016, initially as a grass-roots movement in which some development teams of Group IT started working with Scrum. Relatively quickly more than 10 teams started to work with Scrum. Post set up an "agile transition team" in 2017 to support and guide the transition.

Michael Niessl; michael.niessl@post.at
Carmen Gruber; carmen.gruber@post.at
Martin Eder; martin.eder@post.at

Field Code Changed

Field Code Changed

By 2018 about 15 agile teams from Group IT worked on various software products in Post. The agile transition team evaluated scaling frameworks and settled on SAFe (Scaled Agile Framework). In mid-2019 the first agile release train had been established and the first big room PI planning had happened.

In fall of 2019 Post added a system team to the train and started planning a second release train. By mid-2020 a second agile release train started its work. The agile transition team transformed to a “lean agile center of excellence” and started planning another release train.

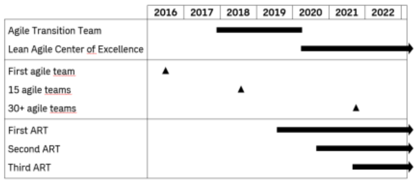

Figure 1~~2~~. **history of agile development at Post**

Today there are more than 30 Scrum teams working in a total of 4 agile release trains. Over time, though, the established processes – especially those related to “Scaled agile” – raised problems that kept adding up. The rest of this report describes those problems and how we addressed them by restarting our approach to scaled agile development.

4. PROBLEMS

Although the implementation of a standardized framework across teams was an important step for Post to create a more unified agile approach across a large development landscape, over time the limitations and costs of this approach began to outweigh the benefits.

4.1 Lack of focus

One of the main problems we identified with our implementation of SAFe was that processes began to be valued above all else. This was evident in meetings that involved many people and didn’t have any clear goals or outcomes yet continued to take place without reflection on what we actually wanted to achieve. These meetings were held simply because SAFe prescribed them, or perhaps because they had previously served a need; either way, processes became entrenched and accepted, even if their purpose wasn’t clear. Additionally, large numbers of people taking part in some of these meetings contributed to a lack of focus. Rather than thinking about the purpose of a meeting and inviting the necessary people, train meetings became bloated and were only marginally relevant to a large portion of the participants.

An underlying problem that contributed to large, unfocused meetings was the fact that Agile Release Trains were not clearly organized around value streams, nor were the domain boundaries of the trains clear. Over time, more teams and projects/products were added to the trains without sufficient consideration for the value that the train would collectively deliver. As such, one of the Agile Release Trains was ultimately made up of 9 teams working on products across core logistics, distribution, and even some customer-facing applications.

4.2 Planning Horizon and Pace

In addition to the fundamental problems discussed in section 4.1, the ten-week planning horizon as well as everything building towards the PI Planning presented us with challenges. Each of the teams within the Agile Release Train worked in two-week sprints, with each Program Increment (PI) comprising 5 sprints; PI Planning thus took place once every ten weeks, which is the standard SAFe planning cadence. This rather long planning horizon coupled with one big room planning event presented us with two main problems: intensive periods of refining and planning just before and during the PI Planning followed by a long lull, and lack of time to sufficiently refine features and user stories committed to the next PI.

The nature of the PI Planning as a big room planning that only takes place every 10 weeks resulted in a program backlog that saw a flurry of often ill-defined features added just before the PI Planning in the hopes that these features would be taken up by a team, followed by long down-times with little activity. Rather than using the 10-weeks of the PI to regularly refine and prioritize features, everything built toward one or two hectic weeks of meetings directly before the PI Planning. Therefore, we were never able to create and maintain a sustainable pace of development at the train level.

A further problem that evolved out of our lack of sustainable pace was that teams had a hard time planning for a full 10-weeks. The PI Plannings themselves were intense 2-day affairs, during which myriad features had to be discussed, refined, and planned. Many of these features had only recently, and sometimes hastily, been added to the program backlog and were completely new to the teams. They then had very little time to understand the features and clarify questions with stakeholders before committing the feature to the PI. Teams could of course reject features if they were too unclear, but they often felt pressured to commit highly prioritized features regardless of whether they were well-defined or not. Such poorly specified features meant that we were pre-destined for scope creep within the PI and presented immediate risks to the feasibility of the plan.

4.3 Bureaucracy and Roles

With the introduction of SAFe, we also had a proliferation of roles that were not well-defined, and in some cases even somewhat redundant. This appears to have been the result of taking on new “agile” roles defined in SAFe, while simultaneously keeping “old” roles. A role such as Team Lead is thusly lived very differently in different teams and sits awkwardly next to the Product Owner and Release Train Engineer roles. Additionally, the role of Product Owner was partially merged with the roles of Business Analysts and Requirements Engineers, such that POs are also often Requirements Engineers for other teams and products. However, we also have a team of Requirements Engineers who are not in or working with any specific agile team. In short, many roles are confusing, overlapping, and unclear. This in turn has a negative impact on accountability and ownership, since responsibilities are not well defined.

Confusion about roles and responsibilities creates waste and unnecessary overhead. It also results in more bureaucracy as teams either try to protect the territory they consider to be theirs or try to push away work they don’t feel responsible for.

5. SOLUTIONS

Because of the challenges described in section 4, we started addressing these issues with concrete measures in one of the existing agile release trains.

5.1 Post Rolling Refinement Model (PRIME)

To raise awareness for the problems we identified and to gain traction for our proposed solution we put together our own “framework”; a set of very simple rules and a bare minimum of processes supporting scaled agile work. We made sure that solutions to the major problems mentioned above are implicitly addressed by the framework but left room for “local” adaptions if necessary. As is usually done within Post we picked a name for “our” framework and first started promoting it and eventually rolled it out in our release train.

PRIME aims to remove a lot of the bureaucratic overhead introduced by the SAFe framework and tries to focus on *being agile* instead of *doing agile*. It strives to push as many decisions as possible to where the actual work is done (= the Scrum team), while keeping only necessary cross team coordination on train level. Our SAFe implementation had become quite bloated with meetings, as can be seen in the figure below. Most of these meetings also required large numbers of participants.

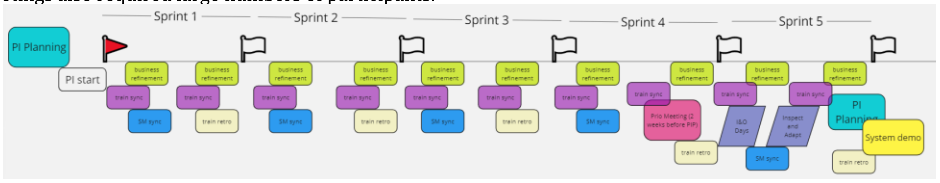

Figure 2~~3~~. **meetings before PRIME**

In PRIME, we aimed to reduce this to an absolute prescribed minimum to get started, and allowed the teams and trains the flexibility and freedom to add what they needed over time. The prescribed minimum set of PRIME meetings can be seen in Figure 4, each with a maximum timebox.

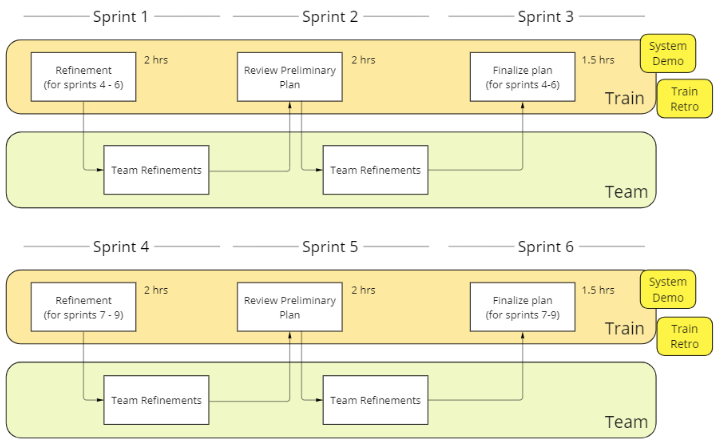

Figure 34. **core of PRIME**

PRIME (see Figure 4) introduces a shorter iteration of just three sprints (compared to the five used with SAFe), significantly reduces the number of meetings, and reduces the number of required participants per meeting; more on meetings in chapter 5.2. This is because we identified only the minimum set of meetings required and eliminated the rest. Teams/Train are of course free to add meetings as they see fit, but PRIME does not prescribe them. Breaking down the PI planning into smaller, more spread-out meetings also allows for representatives rather than whole teams to take part, thus making the meetings optional for more people, while simultaneously providing more time for information gathering and dissemination before finalizing the plan.

PRIME removes the need for a single big room PI planning session by using each iteration to continuously refine the plan for the next one. Team representatives meet at the train level for 1) initial refinement, 2) plan review and 3) plan finalization. The immediate results of those meetings will be discussed and refined on team level. The train backlog - as with team backlogs - is refined on a continuous basis while the top of the backlog is always being estimated and prioritized, removing the need for deadlines and prio meetings. A positive change we've noticed during PRIMEs is that now we have a higher quality of planned features.

While we clearly deviated from SAFe by introducing our own framework, we emphasized the evolutionary progress we were making. It was important to us to keep some of the things introduced with SAFe. We continue to call a group of teams working together a "train" (but we don't use the terms agile release train or ART). We continue to have an innovation day per PRIME iteration (but not a "planning and innovation" sprint). We also continue to have system demos.

Our Implementation of SAFe	PRIME
Decision pushed up, which reduced ownership in the team. All team members are mandatory in all meetings.	Decisions anchored locally, which enabled teams to take more ownership and reduced the size of meetings as only team representatives are needed.
Backlog not maintained continuously but deadline driven once every 2 months, teams not involved.	Backlog refined continuously (= rolling backlog), teams involved in the process, which led to more visibility and transparency
Standard iteration length which was not a good fit for our stage of transformation.	Reduced length of iteration which made it easier to have an overview of and understand the planning horizon.
Number of meetings grown out of hand, unfocused with large lists of participants, hard to facilitate, hard to see value.	Reduced number of participants, focused meetings, removed unnecessary meetings.

Table 1. **key differences**

PRIME does not solve all our challenges but addresses a wide array of problems:

5.2 Addressed Problem: Meetings

Instead of trying to handle every bit on “train level” leading to large, unfocused meetings as described in section 4.1, we tried to put the teams in the center again and only require representatives of teams (and roles) in every team meeting. Representatives are responsible to act for the team and take decisions back to the team as soon as possible.

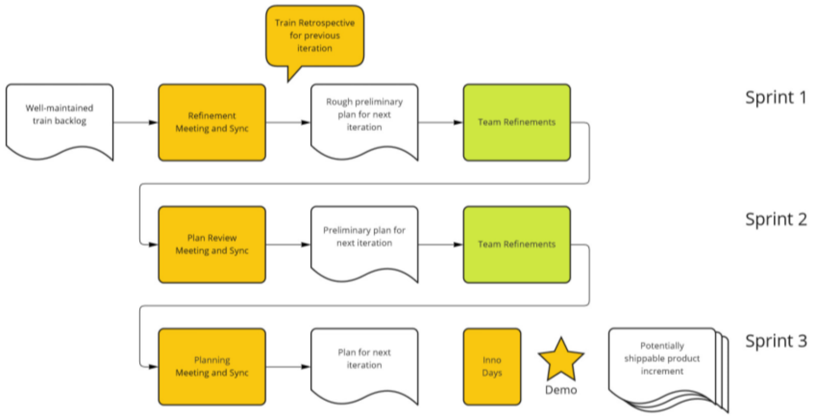
Train Retrospective for previous iteration
Well-maintained train backlog
Refinement Meeting and Sync
Rough preliminary plan for next iteration
Team Refinements
Sprint 1
Plan Review Meeting and Sync
Preliminary plan for next iteration
Team Refinements
Sprint 2
Planning Meeting and Sync
Plan for next iteration
Inno Days
Demo
Potentially shippable product increment
Sprint 3

Figure ~~4~~5. **train meetings**

PRIME defines 6 types of meetings on train level:

- Initial plan refinement, plan review and final plan meeting: for the continuous refinement of the backlog and the continuous planning process (that replaced the PI planning)
- Train Retro: equivalent to the one described in SAFe
- System Demo: equivalent to the one described in SAFe
- Innovation Day: equivalent to the one described in SAFe

Meetings now happen less frequently, have a clear purpose and are generally more focused. The improved focus of the participants is achieved by reduced meeting complexity. They foster an environment where work happens within Scrum teams and the train backlog gets continuously refined and is well-maintained.

5.3 Addressed Problem: Planning Horizon

The rolling refinement described above was meant to address some of the problems with the planning horizon as described in section 4.2. Instead of having one big planning every 10 weeks, with an actual “call for feature deadline” the rolling nature of our refinements made the process more fluent and less focused on this *one* deadline.

Additionally, to contain the fear of missing out for another long 10 weeks, we reduced the length of our iteration from 5 sprints to just 3. This makes re-planning less likely and makes it mentally easier to “miss” one iteration and wait for the next.

5.4 Addressed Problem: Roles

As described in section 4.3, we had (and in parts still have) many roles associated with software product development in some sense. However, for the train we clearly defined just 4 major roles in PRIME:

- Scrum Teams: those are the teams where the work is done. Usually they comprise Product Owner, Scrum Master, Test Engineers / Engineers / Architects (the “developers”).
- Train Coach: like a release train engineer, a train coach facilitates train meetings, inspects train processes and in general strives to improve the train.
- Head of Engineering: responsible for the train’s dev team and the technical delivery (the “how”)
- Head of Product: responsible for the products of the train, owner of the train backlog (the “what”)

Additionally, there might be supporting roles depending on the train and situation. These roles include the train architect, operations/DevOps, system team, project manager, lead architect, test manager and others, but PRIME doesn't prescribe any of them.

5.5 Addressed Problem: Value Stream Mapping

As mentioned in section 4.1, it was hard to focus within our agile release train as it comprised 9 teams with completely different areas of expertise. Looking at the core value stream of Post and how the business is oriented along that stream, we concluded that we should be aligned just as well. That's why we cut the train in two, aligning each half to one part of the stream as shown in Figure 5~~Figure 6~~.

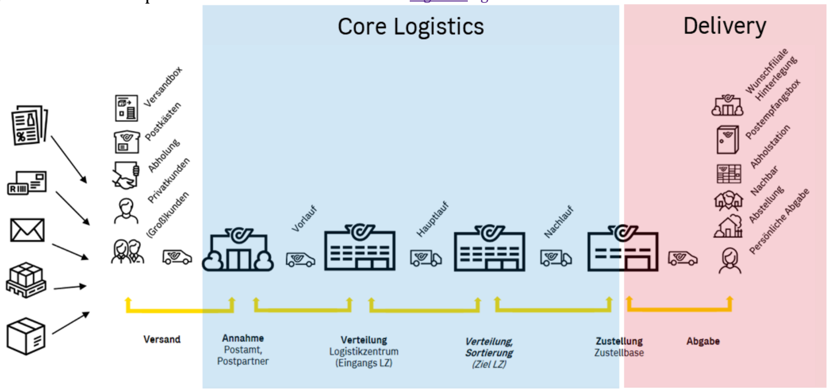

Figure 5~~6~~. **core value stream of Post with two trains aligned to it (Core Logistics and Delivery)**

We decided on such a slice for two main reasons:

- It is aligned with how the rest of the organization works which is functionally organized. It allows to align with business stakeholders and collaborate.
- It is in line with our analysis of inter-team dependencies, answering the question: are teams of "the other half" in fact necessary for end-to-end delivery? – they are not for the most part as shown in Figure 8 Figure 6~~Figure 7~~where we analyzed 4 consecutive PIs (a total of 20 sprints).

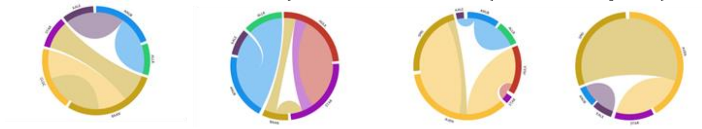

Figure 6~~7~~. **dependency graphs show two separate trains**

6. WHAT WE LEARNED

During our journey we had the opportunity to learn a couple of lessons that we would like to summarize in the following section.

6.1 Changes are hard…

Looking at the success rate of the newly established trains, we can see an immediate drop after the implementation of PRIME (dropping from the 70% average we had with SAFe to a meager 55% in our first iteration). We attribute this decrease to the reduced iteration length and our problem adjusting to it. Shown in Figure 7~~Figure 8~~ are our first 4 iterations after introducing PRIME, proving that we were able to quickly bounce back and achieve success rates never seen before in our environment.

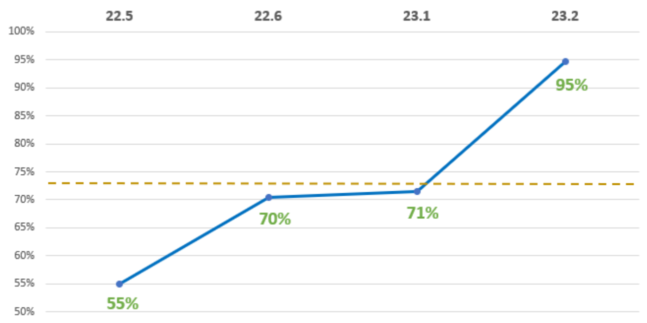

Figure 7~~8~~. **success rate per PRIME**

6.2 … but changes are possible

Even in well-established organizations like Austrian Post, it is possible to initiate significant bottom-up change. Additionally, it does not take as much time as you think it would. Our take-away is that you should not be discouraged or feel constrained by a large organization: change is indeed possible.

6.3 Estimation Meetings

Originally, we planned to use our refinement meetings to get to a shared understanding of the size of a feature. So far this has not worked. We saw major pushback by the senior engineers and architects who wanted to get more time to understand technical details and do some research. We had to bring back separate (small) feature estimation meetings, with the disadvantages of adding meetings to our process, excluding other team members from this task, and reducing transparency.

6.4 Involvement and ownership increase buy-in

We have seen fast acceptance for the model we implemented and large buy-in from most people involved in our trains. We believe that's because PRIME allows more freedom than (our implementation of) SAFe, puts more responsibility in the hands of the teams and in general is perceived as being more transparent.

7. WHAT'S NEXT

Although PRIME has helped us to streamline our train processes, some problems remain. One of these is a continued strong focus on projects rather than products. Projects can become bureaucratic and include stage gate processes that are anathema to agile software development. One of our most important current initiatives, the building of an event-streaming platform, is being developed largely outside of project structures, but a strong project-orientation continues to be a problem that cannot simply be solved with a new scaling framework.

We also continue to have some roles that are not properly defined, most notably the double PO role in which the same individual is a PO in one team, and concomitantly Requirements Engineer for projects or products in other teams.

Another challenge we continuously face is the tension between planned and unplanned work. We of course want to be able to react to change, but we often have many small change requests that individually seem small but can amount to the death of our plan by a thousand cuts. This also means that we recognize the risk to the plan rather late, and therefore also react very late. Indeed, it also forces us to be reactive instead of proactive. Teams currently leave a buffer in their plan to deal with unexpected or unpredictable changes, and we have started tracking these metrics more closely to first make the amount of unplanned work more visible, and secondly to inspect and adapt as we go forward.

8. ACKNOWLEDGEMENTS

First and foremost, we want to thank the *teams and individuals* forming our trains for their willingness to try something different – without their commitment to continuous improvement, the change described in this report would not have been possible. We want to acknowledge *Agile Alliance* for trusting that what we can share is interesting for a larger audience. Finally, we want to express our gratitude for the guidance of our shepherd *Henrik Sternberg*.