A Scaled Agile Framework® (SAFe™) Case Study

Executive Summary

A leading Cable Television company faced long and challenging development cycles with quality problems. Guided by a small team of Agile Coaches*, they successfully implemented the Scrum framework with SAFe™ to scale up to 150 people delivering their set-top box/DVR software and server-side systems to support the DVRs. The following challenges drove the need for change:

- 12+ month release cycle; unable to respond to a rapidly changing marketplace
- Missed delivery dates; schedule slippage
- Quality problems due to late integration and 3 concurrent versions in development

Agile methods and SAFe™ reduced time-to-market for major releases from 12+ to only 4 months, the shortest practical timeframe given the cost of deploying firmware to over 11 million DVRs nationwide. Releases changed to fixed-date; scope was managed and prioritized to ensure that all business capabilities were delivered on-time, even though some low priority features ('bells and whistles') were cut to meet the delivery date. Quality improved significantly as a result of earlier and more frequent integration testing which is fundamental to the Agile approach. SAFe™ was tailored to the organization's unique needs after piloting elements of it on a smaller scale.

Key conclusions:
- The Agile Release Train model is effective for coordinating efforts of multiple, tightly integrated teams toward a short-term delivery.
- Many elements of SAFe™ are can be eliminated or scaled back when teams are working on decoupled or only loosely integrated products, features, or components; the Program level in particular may be excessive.
- SAFe™ is sometimes implemented in its entirety in a "big bang" change. This is possible but extremely challenging and risky. Our recommendation is to implement elements of SAFe™ in pilot mode to address known pain points, empirically determining which elements work and how, rather than pushing unproven changes to large swaths of the organization.

Scaled Agile Framework® overview

The ScaledAgileFramework website thoroughly describes the SAFe™ model. SAFe™ defines three levels for scaling an Agile organization:

1. Portfolio
2. Program
3. Team

At the portfolio level, Lean-Agile principles are applied to balance workload with delivery capacity and optimize value delivered, while aligning architectural efforts. At the Program level, product features are integrated and tested often by a System team. At the team level, multiple Agile teams build and test small features within a single business domain or system component, and deliver running, tested features (user stories) on a set cadence (usually every 2 weeks) to the System team. SAFe™ prescribes fixed released dates with variable scope using the *release train* metaphor; if a feature misses the train (the date), it has to wait for the next release train.
Other frameworks for scaling agile methods are also useful in many contexts, including Disciplined Agile Development, Large Scale Scrum from Larman/Vodde, and Scrum of Scrums.

Managing change: evolution or revolution?

At agile42, we recommend an incremental and empirical approach to introducing agile practices at scale, rather than prescribing one particular framework to implement. Scaling lean-agile practices is a complex problem and every organization's context is unique. Long-term success is more likely with an empirical and evolutionary approach, as described in agile42's Enterprise Transition Framework™.

(1) Assess challenges to identify specific needs for improvement
(2) Pilot changes in a low-risk way
(3) Empirically measure the results of the change
(4) If the pilot succeeds, expand the practice more broadly
(5) Repeat...

In the cable TV case study, agile practices were first piloted by 2 teams. We tried many of the SAFe™ practices thru the pilot efforts and used the lessons learned to guide the expansion of Agile and SAFe™.

Where the full SAFe™ framework was excessive

A different agile42 client, a financial institution, issued a corporate mandate to implement SAFe™. In this case, teams did their best to implement all of SAFe™ in a "big bang" rollout. It became clear after a few releases (4 months) that significant portions of SAFe™ were unnecessary, and even counter-productive in their context. Their five teams were each working on mostly independent applications, and there was no need for the overhead and coordination of a Program-level agile release train so they abandoned it, allowing teams to

operate more independently. An evolutionary approach could have helped this organization learn what parts of SAFe™ were applicable in their context in a less disruptive manner.

Reducing time to market

In our case study, the cable TV company changed their release cycles as shown in Figure 1.

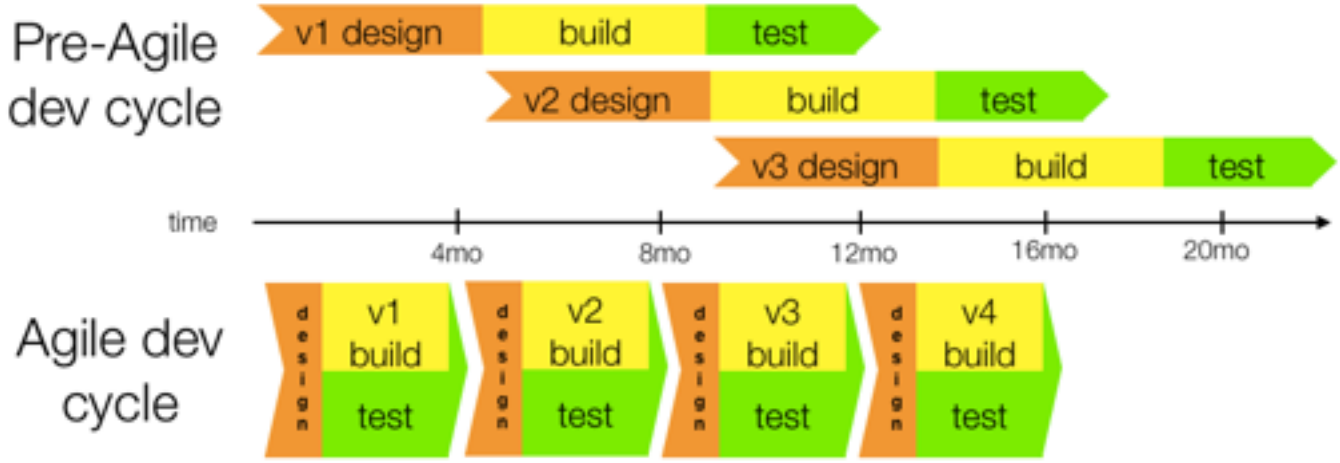

Figure 1 - Product dev cycle before and after

The agile development cycle uses the *release train* concept from SAFe™. Releases have a fixed date, and scope is selected -- and adjusted if necessary -- in order to meet the deadline. *If a feature misses the train, it has to wait for the next train.* By aggressively prioritizing scope throughout development, and frequently integrating and testing, this model ensures that a viable product with the most important features will be ready on the planned date.

Portfolio Planning

The R&D organization started with a list of over 150 requests for features (projects) from the business. Senior leadership formed a Product Council consisting of 10 Product Owners (product managers), each of whom was aligned with a particular business area, plus R&D Directors. The Product Owners each made a 'sales pitch' for the highest value projects/ features in their own domain, and the Product Council stack-ranked all the requests that might fit into a 4-month development cycle based on ballpark estimates. Ranking was accomplished by first scoring each request on a number of criteria: importance to business stakeholders, alignment with strategic initiatives, and cost of delay (urgency). This objective scoring cut the number of 'contenders' down to a more manageable number, from which point the Council members used a multi-voting technique to arrive at a final ranking.

Agile team structure

See Figure 2 for a description of team structure before and after. Before Agile was introduced, most of the people worked in large teams organized around technology components: the DVR

(client) component and several back-end server components. Most of the business features however, required both client *and* server. As a result, there was no clear ownership of the end-to-end business value. In the agile model, most of the people were organized into smaller *feature teams* (purple in Figure 2 below), each one owning features across client and server for one area of the business. One *component team* on the server side remained focused on building a major new architectural service. To maintain design integrity across feature teams, virtual platform teams coordinated designs across all the feature teams, as shown by the dotted line boxes in Figure 2.

At first, the management team thought it wouldn't be possible to form small cross-functional feature teams because each one would require too many people across too many specialties. So they put the names of every person on a separate card and began moving them around, trying to form feature teams of 10 people maximum. The managers were surprised to find that could form feature teams with only few gaps in skill sets and a handful of specialists (such as DBAs) who would need to serve multiple teams. Some organizations have accomplished the same structure through self-organization: allowing all the team members to collectively choose teams, rather than having a few managers do it. This organization wasn't quite ready to embrace that idea.

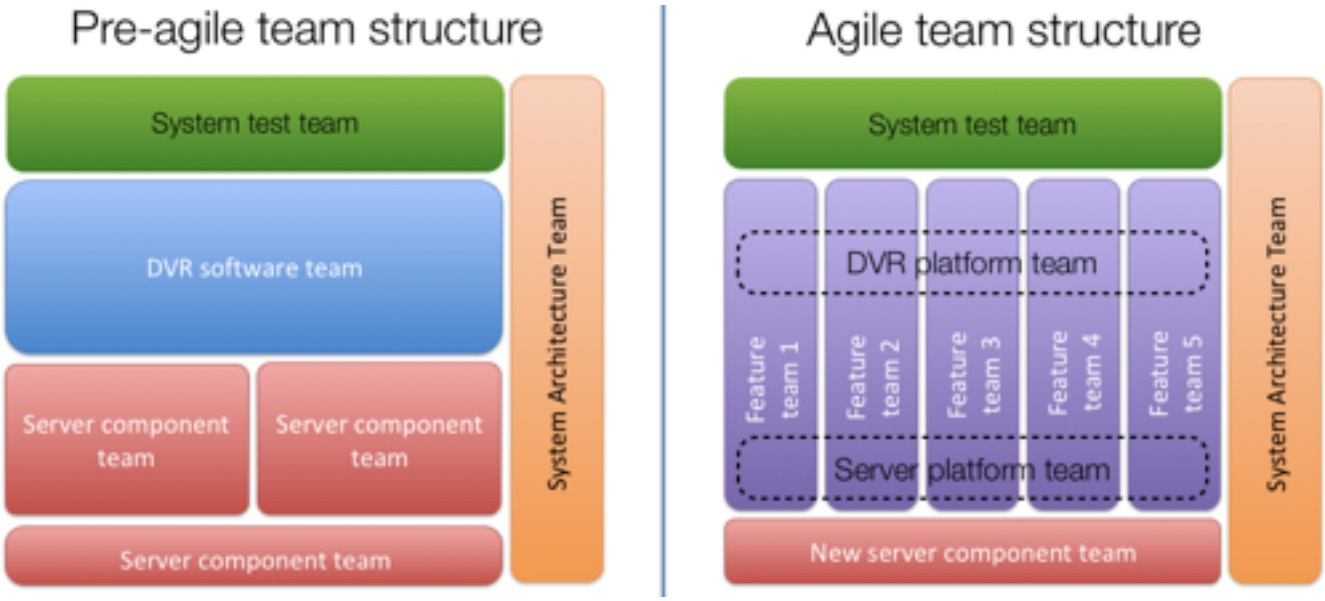

Figure 2: Team structure before and after

Release train (4 month) planning

Figure 3 below gives an overview of the release train timeline.

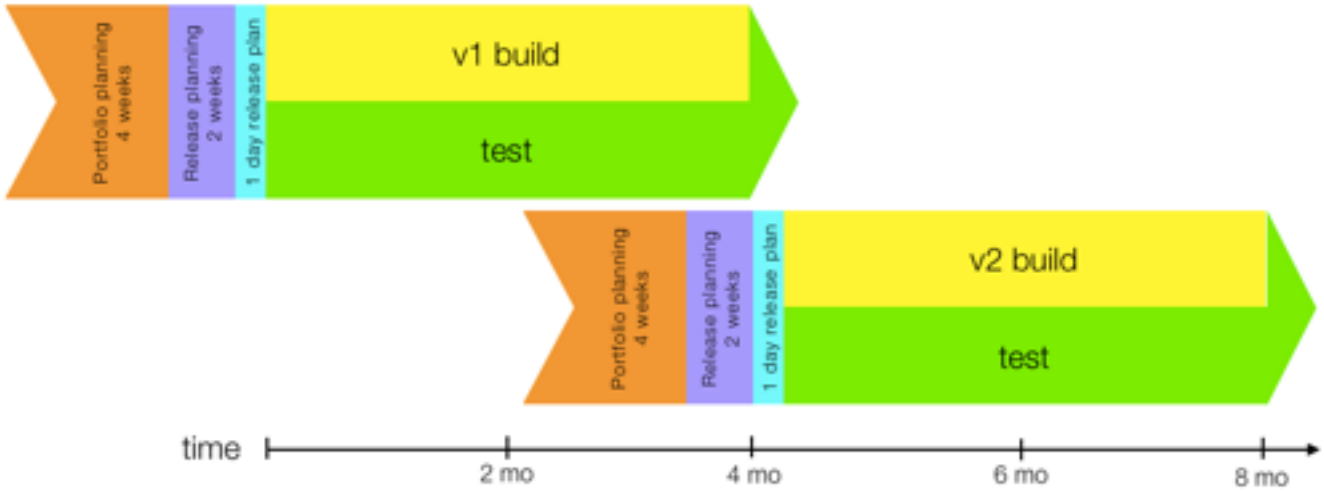

Figure 3: Overview of the release trains from portfolio planning to delivery

- 4 weeks of portfolio planning
- 2 weeks for each team's independent release train planning
- 1 day for all teams to build a combined plan for the release
- 4 months for building and testing - using 2-week sprints/iterations

With the portfolio priorities clear, and team structure decided, each new team spent about 2 weeks doing high-level *release train* planning. Each release train was a 4-month period culminating in an integrated delivery from all the teams. Each Product Owner decomposed high-level business requests (features or projects) into smaller pieces (called stories), and prioritized the stories. The newly formed teams independently estimated the scope they could deliver in 4 months and identified dependencies on other teams.

The entire R&D organization (about 120 people) gathered in one room for the 1-day release planning event, except for one team that joined remotely by video conference.

1-day release planning agenda:

- VP of R&D shared the vision and goals for the upcoming 4-month release train
- Marketplace of collaboration: Each of the 10 teams had a large, visible timeline of features they planned to deliver in 4 months. People circulated between teams to better understand synergies and negotiate dependencies. (See Figure 4)
- Each team adjusted their plan to reflect newly discovered dependencies and adjusted scope
- All teams combined their release plans into a single visible timeline covering the 4-month period. (See Figure 5)
- A retrospective on the one-day event: lessons learned to improve the next one.

Figure 4: One team's release plan on the wall; collaboration with other team members

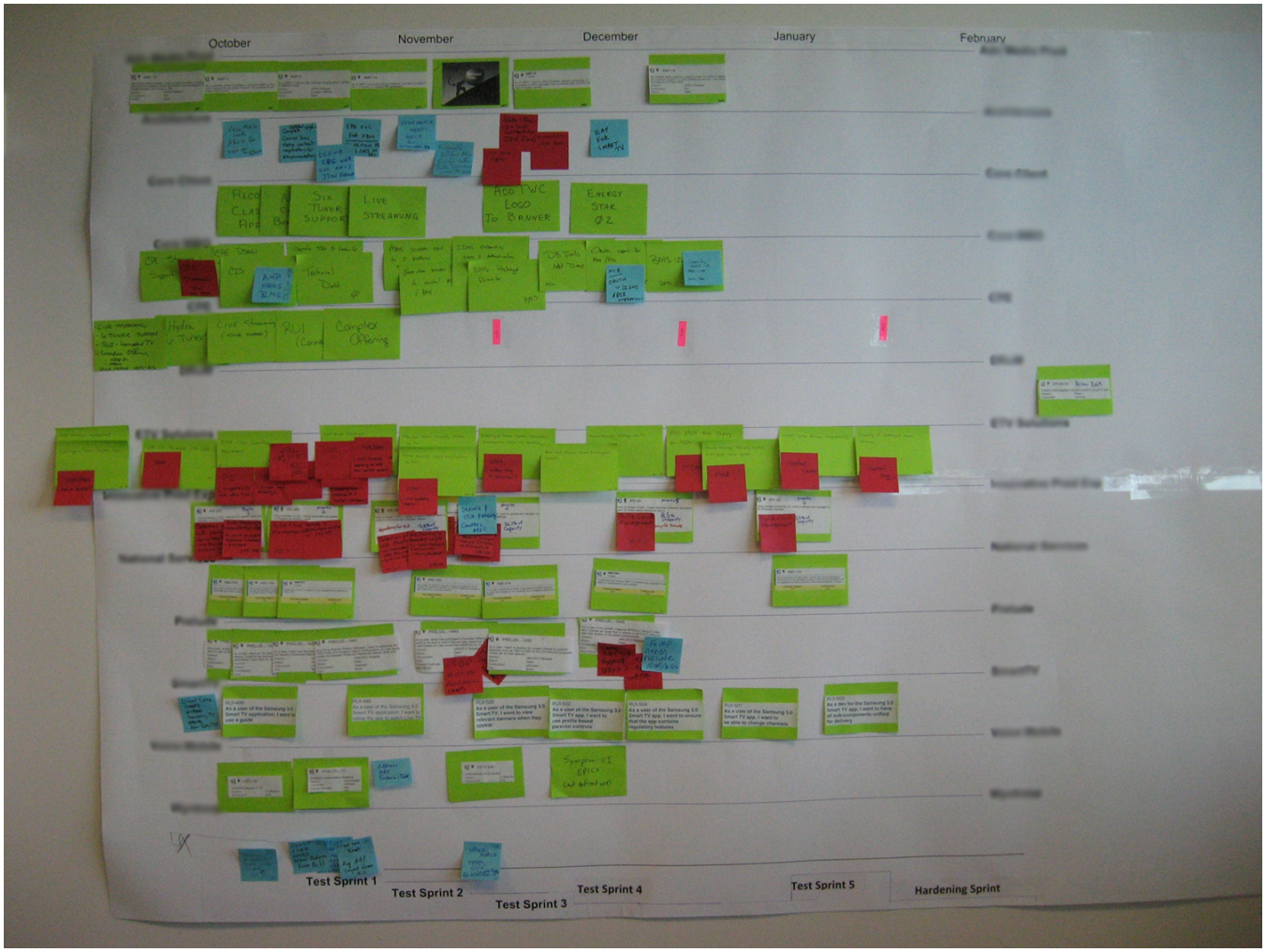

Figure 5: Combined release train plan for all 10 teams

Delivery Sprints

Each team worked in 2-week sprints (development iterations) throughout the four-month release train. The system test team integrated the work of all teams every sprint to test new features and run a regression test on the entire system. Some tests were automated but many required manual validation of video. The Product Owners from each meet met biweekly (once per sprint) to coordinate their work; additional team members participated when necessary. The final 2-week sprint was a 'hardening sprint' with all hands on deck to perform final regression testing.

Results

- The release was delivered on time with 100% of planned business capabilities delivered and 95% of planned low-level features included.

- Quality was higher than previous long-cycle releases: fewer total defects total and fewer severe defects were discovered post-release.
- The one-day release planning event was an overwhelming success. People really appreciated the opportunity to understand the big picture and quickly reach a common understanding of the goal and scope of the release.

Challenges:

- Reaching consensus on portfolio priorities was difficult and time-consuming. The Business Value Game or Buy a Feature would make it more effective.

- Forming feature-oriented teams was initially viewed as impractical due to the large number of specialists required to build the perfect team. Through many rounds of name-swapping, we arrived at a set of teams that each was focused on a single business value stream and consisted mostly of full-time dedicated people. A small number of specialists spread their time between multiple teams to fill specific gaps.

- Regression testing every two weeks was possible only because the organization had invested in test automation. Still, some testing was manual and incremental testing was a significant shift for the system testing team.

- One of the feature teams struggled to integrated the client-side and server-side developers into a truly integrated team. They reporting structure and culture separated those two disciplines, and in practical terms they worked as 2 separate teams.

Conclusions

SAFe™ was an appropriate model for the cable TV company because multiple teams are all building a single integrated and complex product. Prior to adopting SAFe™, the organization had already piloted Scrum on two teams with the help of experienced coaches, and learned how to make Scrum work in their context. This evolutionary approach to adopting Agile and SAFe™ was a critical factor in learning how to succeed in delivering on-schedule with high quality.

The experience of the financial institution, on the other hand, where SAFe™ was mandated, demonstrates the risk of wholesale adoption of a prescriptive framework without first piloting changes on a smaller scale and measuring the results. The financial institution learned that much of SAFe™ was overkill in their context.

Key conclusions summarized

•The release train model is effective for coordinating efforts of multiple, tightly integrated teams toward a short-term delivery.

• Many elements of SAFe™ are can be eliminated or scaled back when teams are working on decoupled or only loosely integrated products, features, or components; the Program level in particular may be excessive.

• SAFe™ is sometimes implemented in its entirety in a “big bang” change. This is possible but extremely challenging and risky. Our recommendation is to implement elements of SAFe™ in pilot mode, evolving as you learn which elements work and how, rather than pushing unproven changes to large swaths of the organization. The agile42 Enterprise Transition Framework™ takes the evolutionary approach.

*The team of Agile Coaches included Brad Swanson, Manny Segarra, Deanna Evans, and Ken McCorkell.