Editor: **Christof Ebert**
Vector Consulting Services
christof.ebert@vector.com

Scaling Agile

Christof Ebert and Maria Paasivaara

From the Editor
Agile software development has become mainstream. Industry-scale agility for distributed teams, large projects, or critical systems requires scaling agile practices, which agile scaling frameworks attempt to provide. Here, Maria Paasivaara and I explore frameworks such as the Scaled Agile Framework (SAFe) and show best practices from two industry case studies. I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —*Christof Ebert*

AGILE PRACTICES HAVE evolved steadily. During the early '90s, Microsoft invented most of what was later called agile.[1] Driven by the fast-growing complexity of their Windows and Office product suites, Microsoft early on advanced concepts such as continuous build, feature-driven teams, and the close connection of business needs with requirements and architecture flexibility. A key milestone was Internet Explorer, which Microsoft fully redeveloped in the late '90s to allow for flexible, scalable evolution. These practices found their way to the early agile frameworks. The initial Agile Manifesto, based on the experiences of Microsoft, IBM, and others, primarily collected principles and practices. The label "agile" was a wonderful marketing stimulus and immediately triggered hype, specifically regarding small teams and low-risk products.

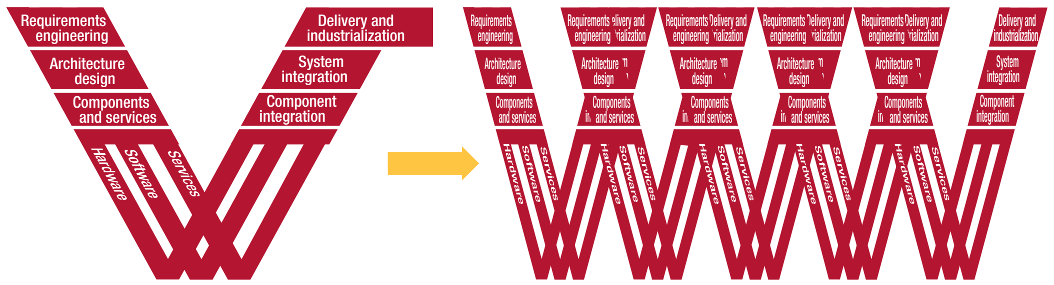
Industry soon realized that critical systems need more than an agile manifesto. Industry-scale software development typically doesn't fit with the heavy constraints of early agile practices, such as Extreme Programming. Projects easily take several years and span teams worldwide. Global software engineering demands that agile practices be scalable.[2–4] Safety-critical systems need thorough documentation, once neglected by agile proponents.

In addition, business models in software-driven systems have evolved to flexible ecosystems. The classic functional split demanded by legacy-driven architectures has been replaced by a more service-oriented architecture and delivery model. Recent technology trends, such as three-tier cloud architectures, adaptive component frameworks, and connectivity for the Internet of Things and Internet of Services, facilitate new business models and scalable reuse across companies and industries.

Again, these industry developments require agile practices to be scalable. However, scaling isn't easy; large projects often are globally distributed and have many teams that need to collaborate and coordinate. So far, proven models haven't been available. Empirical research is missing on how to do a large-scale agile transformation and what the results should look like. Here, we provide an overview of agile scaling frameworks.

A Look at Some Frameworks

Agility in its early years was more often a dogma than a coherent yet flexible framework. At times, some of the "gurus" preferred to fight processes for the sake of revolution, rather than meeting industry needs. Even today, developers and



Classic V model for systems engineering:

- Goal: Collaborative development of safety-critical functional components
- Organically grown synchronization points between hardware, software, and emerging services
- Governance-oriented heavy processes with insufficient flexibility and agility

Agile scaling frameworks:

- Goal: Flexible collaboration across the lifecycle with adaptive architectures and assured quality
- Full vertical abstraction layers with clear interfaces and synchronization points
- Flexibility and agility while pertaining to quality demands

FIGURE 1. Agile scaling frameworks claim to provide a recipe for adopting agile at the enterprise scale. The classic V development model is evolving to a W model of continuous delivery.

managers are puzzled by traditional agile themes such as "customer on board" or "software before documentation." Critical systems need more meat in order to work flexibly while considering product liability and governance needs.

The classic V development model is evolving to a W-shaped model of incremental development and continuous delivery, thus incorporating typical agile practices (see Figure 1). The W model starts with agile planning, which involves such things as the frequency of deliveries, increments, and the train of releases. Then it proceeds to incremental design, verification, and integration. Service-oriented architectures allow adjusting to services and microservices and evolving toward DevOps practices, such as upgrade paradigms during the operation of a system. A typical example is the over-the-air upgrades of modern automotive onboard infrastructures.

Current approaches for scaling agile blend agile and lean practices to address real industry needs. Organizations can use such frameworks to help achieve these transitions:

- *Business model*—from static building and selling to dynamic composing and continuous delivery.
- *Governance*—from simple IT systems and throwaway apps to interwoven quality assurance with auditable product liability, functional safety, cybersecurity, privacy, and availability.
- *Process*—from the V model with heavy release cycles to scalable agile processes.
- *Competences*—from silos of functional know-how to IT as a core competence of all engineers.
- *Development*—from components and functions to services.
- *Architecture*—from localized features to service-oriented patterns with the convergence of embedded electronics and open IT systems.

The experience reports on these frameworks' home pages present fabulous success stories. However, few independent empirical studies exist on how the frameworks work in practice, what circumstances each framework suits best, and what the challenges are and how to overcome them.

Table 1 compares five agile frameworks, which we selected on the basis of recent surveys and what we see in industry usage. For example, in a recent survey, almost 30 percent of the respondents said they used the Scaled Agile Framework (SAFe) to provide scaled agility.[5] Managers find it comfortable because it has plenty of role definitions, which hasn't been explicit in the classic agile toolkits used in the past two decades. On the other hand, many users perceive such a highly prescriptive role-and-process scheme as overhead and not any more agile.

Many practitioners consider SAFe too heavy and complex.[6] Like CMMI in the past, it tries to include all best practices but doesn't provide guidance on how to scale down. Some even say that SAFe adds

Table 1. A comparison of five agile frameworks.

Criteria	Framework				
	Scrum of Scrums (SoS)	Scaled Agile Framework (SAFe)	Large-Scale Scrum (LeSS)	Disciplined Agile Delivery (DAD)	Lean Scalable Agility for Engineering (LeanSAFE)
Scope	Software, hardware, and systems; flexible	Software	Software	Software	Software, hardware, and systems
Differentiator	Enables scrum for all situations and scales	Is complex, with many artifacts, roles, and guidelines	Provides flexibility by offering only suggestions	Is complex, with coverage of many models	Works with critical systems
Underlying technology	Scrum	Scrum and other agile principles, lean	Scrum	Scrum, lean	Scrum, lean
Adoption	Usage in many companies	Usage in several companies	Usage in several companies	Usage has started	Usage has started
Scaling	Is flexible and good to adapt in different settings	Targets large companies but is perceived as heavy	Can be adapted to different settings	Can be adapted to different settings	Can be adapted to different settings
Complexity	Low	High	Medium	Medium	Medium
Cost	Low	High	Medium	Medium	Low
Globally distributed teams	Feasible	Feasible	Feasible	Difficult	Feasible

to bureaucracy, evolving to "the new waterfall."

Other agile frameworks, such as Large-Scale Scrum (LeSS) and Lean Scalable Agility for Engineering (LeanSAFE), have addressed this high complexity. They define much less and give more freedom to tailoring. So, companies must incrementally compose their own framework and empirically measure which practices have the best fit and value.

Using an Agile Framework

The deployment of SAFe at two globally distributed companies illustrates the use of scalable agile technologies. Comptel, a telecommunications company that was recently acquired by Nokia, processes mobile-device usage data. It serves more than 300 customers, mostly communications service providers, worldwide. NAPA provides software for designing and operating ships.

SAFe is complex and provides a huge set of templates and process elements.[6] It includes the team, program, and portfolio levels as well as the optional value stream level. At the team level, it adopts Scrum practices, but using Kanban is also possible. At the program level, it embarks on incremental deliveries with different scales, such as the concept of an agile release train. This corresponds to sprints at the team level, but with a longer time frame. As a process framework, SAFe also determines specific roles—for example, system team, product manager, system architect, release management team, or deployment team. At the portfolio level, planning is often based on *epics* that define large development initiatives. The value stream level supports the development of large, complex solutions, which require multiple, synchronized releases.

Before the SAFe adoption, both Comptel and NAPA had been using agile development at the team level for years. However, they both lacked agility in other parts of the organization, and they especially lacked support structure in the organizational layers above the team level. In addition, both companies hoped to improve their capability to deliver more frequent releases, add portfolio level prioritization, manage work from ideas to implementation,

ADOPTING SAFE AT COMPTEL

We compared the adoption of the Scaled Agile Framework (SAFe) in two business lines of Comptel, a globally distributed organization that was recently acquired by Nokia. The business lines started their adoption at different times, within six months. On the basis of our interviews, we determined that the business line that started later had a more successful transformation, especially because that business line learned from the other business line's experience.

We identified seven success factors, which we briefly elaborate here.

First, train personnel well in advance. Comptel learned that training both the managers and team members on the framework and its adaptation is essential. The earlier adoption didn't properly perform this training, which led to challenges.

Second, inform and engage people. Ensure from the start that everybody understands the reasons for the change and why it's important. In the first adoption, the lack of knowledge and communication about the change increased the resistance to change, whereas in the second adoption, the transfer of information was a success factor.

Third, involve change agents. At Comptel, the change agents were people who visibly pushed the change forward, gave training, and contributed to the customization and continuous improvement. These persons included managers, external coaches, and a full-time release train engineer (RTE).

Fourth, hire an experienced external consultant to support tailoring agile scaling and putting it into practice. Comptel hired external consultants both at the beginning to give training and during the transformation to coach the key personnel and to help with the first program increment (PI) planning event. The company saw this external support as a main success factor.

Fifth, prepare well for the first PI planning event. In the second adoption, the RTE prepared extremely well, with an external coach's help, by creating agendas and instructions for the participants. The whole organization participated in the event, which positively affected the participants' attitude toward the change.

Sixth, have a full-time RTE. The second adoption employed a full-time RTE, which was considered a success factor. Besides preparing and leading the PI planning events, the RTE managed the coordination by arranging and leading, for example, the Scrum-of-Scrums (SoS) meetings and taking care of the continuous-improvement items. (For more on SoS, see Table 1 in the main article.)

Finally, take recognized improvement items seriously by assigning responsibilities and monitoring their implementation. In the second adoption, as soon as improvement items were raised, the RTE concentrated on improving the ways of working by creating action plans, assigning responsible persons, and following the implementation. Our interviewees were happy because even though the organization faced problems, they knew that improvements were ongoing.

and systematically manage dependencies. Although agile development worked relatively well at the team level, the team members had difficulty seeing how their daily work linked to and affected other parts of the globally distributed organization.

At Napa the adoption was slow and gradual, whereas at Comptel both business lines planned and transformed the organization in a couple of months. Both companies customized the framework to suit their organization and found that to be one of the success factors. Other success factors were

- investing in the first program increment planning event to make it a success,
- employing external consultants to train and coach,
- having active internal change agents,
- having strong leadership support, and
- quickly reacting to feedback to continuously improve and remove the adoption's pain points.

The most serious challenges included change resistance, lack of communication of the adoption, lack of continuous improvement, and lack of training and support during the adoption.

Both organizations considered their transformations successful. They reached their goals:

- more frequent and more predictable releases with better quality,
- improved visibility and communication across the globally distributed organization,

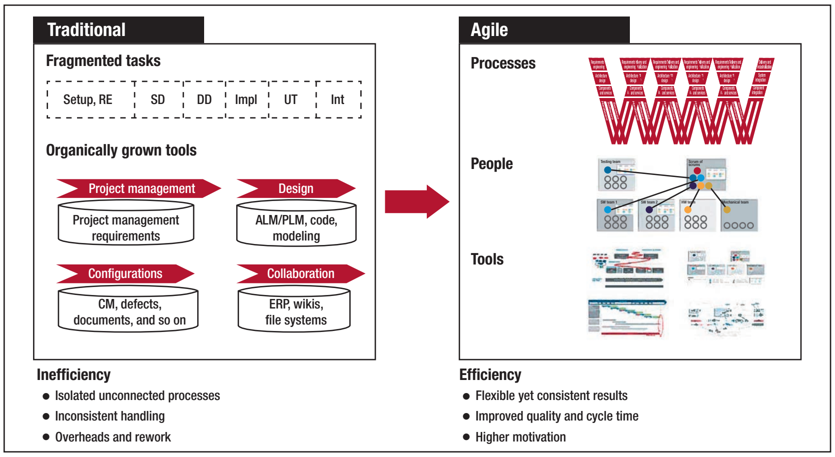


FIGURE 2. Agile frameworks enhance efficiency by integrating different perspectives of processes, roles, and tools. RE = requirements engineering, SD = system design, DD = detailed design, Impl = implementation, UT = unit testing, Int = integration, CM = configuration management, ALM/PLM = application lifecycle management/ product lifecycle management, and ERP = enterprise resource planning.

- better synchronization and collaboration across the teams,
- improved dependency discovery and management, and
- better alignment across the organization.

However, the work isn't over; both organizations find it extremely important to continuously improve their practices after the adoption. The sidebar "Adopting SAFe at Comptel" shows some observations from practical use.

Lessons Learned from Using Agile Frameworks

Here are seven practical recommendations for developers.

- Move from classic functional responsibilities to small, cross-functional teams. Grow methodologies and the underlying technologies to agile engineering.
- Enhance the lifecycle for continuous development. Using frequent synchronization points, change the V model to the W model. Focus on integrity by establishing flexible synchronization points between hardware, software, and service development along the lifecycle.
- Empower distributed teams, both internal as well as suppliers and ecosystem partners, to drive design decisions at their respective level. Ensure consistent documentation with collaborative workflow tools.
- Grow continuous integration of systems and services. Introduce integrated processes and a systematic methodology based on an application lifecycle management and product lifecycle management (ALM/PLM) tool chain.
- Ensure robust system-level design. Approach novel technologies at the system level: system-on-chip, microservices, and cloud solutions for innovative products and services.
- Master relevant quality requirements for critical systems. Cybersecurity, functional safety, service orientation, and usability must be designed and achieved on the systems-engineering level.
- Enhance reuse across platforms, products, and markets. Evolve to self-x-type architectures and technologies such as self-aware adaptive systems to cope with fast-changing components and environments.

In the companies in which we implemented agile scaling frameworks, we found two main transformational patterns. First, it takes effort to tailor the frameworks to concrete environments, organizations, cultures, and so on. There is no "out of the box," and you shouldn't assume that using only some parts will provide early payback. This transformation must start in peoples' minds so that it reaches the organizational culture.

Second, agile frameworks enforce a more comprehensive view of the transformation. Rather than simply focusing on a process or workflow, as in the early agile methods, the frameworks require organizations to adapt processes, roles, and tools in parallel. In addition, it's not only the team level that needs to transform but also the whole organization. Figure 2 shows this evolution, together with the more obvious benefits.

The rising demands for high quality, quick delivery, reduced costs, and flexibility have pushed agility to practically all industries. Users expect the same adaptive behaviors and continuous-delivery models they're used to with their mobile devices.

Agility has arrived in real-world development, beyond mere software applications. Development of even critical systems will evolve to a continuous process that fully decouples the rather stable hardware from delivered services driven by continuous software upgrades. Agile service delivery models combining DevOps, microservices, and cloud solutions will allow functional changes far beyond the traditional V model. Hierarchic modeling of business processes, functionality, and architecture from a systems perspective will allow early simulation while ensuring robustness and security. Development processes across the lifecycle from vision to concept to operations and service will follow this trend to fluid delivery models.

Because constraints vary across application domains, such as liability and governance, agile practices must be tailored to needs and risks. To address such tailoring, companies can use agile scaling frameworks, which can be applied across the organization. From working with many companies worldwide on implementing these frameworks, we've found that introducing agile development means changing the culture and mind-set. It requires long-term commitment, big investments, and customization to a company's specific situation.

Barriers to using agile technologies no longer exist. Developments in globally distributed teams, large projects, safety-critical systems, and hardware and systems engineering have showed that agile technologies are adoptable and adaptable. Scaled agile isn't about tailoring a model; it's about ensuring that the agile culture reaches the engineers and their management. The biggest, most difficult change is that of the mind-set. Changing the practices won't make a company agile if the underlying culture and thinking don't change.

We look forward to seeing other studies on the use of agile scaling frameworks. We're interested in not only success stories but also reports on customizations, challenges, and the solutions to those challenges.

ABOUT THE AUTHORS

CHRISTOF EBERT is the managing director of Vector Consulting Services. He is on the *IEEE Software* editorial board and teaches at the Universities of Stuttgart and Paris. Contact him at christof.ebert@vector.com.

MARIA PAASIVAARA is an associate professor of computer science at the IT University of Copenhagen. She's the general chair of the 2018 International Conference on Global Software Engineering (ICGSE). (For more on this conference, see the related sidebar.) Contact her at maria.paasivaara@gmail.com.

ICGSE

The annual IEEE International Conference on Global Software Engineering (ICGSE) brings together worldwide industry and research leaders in distributed software development. ICGSE 2017 had participants from over 20 countries, with one-third of the papers from industry. ICGSE 2018 will be in Gothenburg, Sweden, colocated with the International Conference on Software Engineering (ICSE), from 27 to 29 May. Join the conference and learn how to succeed with distributed software projects. For more information, visit www.icgse.org.

References

1. M.A. Cusumano and R.W. Selby, *Microsoft Secrets*, Free Press, 1998.
2. C. Ebert, *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*, John Wiley & Sons, 2012.
3. M. Paasivaara et al., "Integrating Global Sites into the Agile and Lean Transformation at Ericsson," *Proc. IEEE 8th Int'l Conf. Global Software Eng.* (ICGSE 13), 2013, pp. 134–143.
4. M. Paasivaara, "Adopting SAFe to Scale Agile in a Globally Distributed Organization," *Proc. IEEE 12th Int'l Conf. Global Software Eng.* (ICGSE 17), 2017, pp. 36–40.
5. *11th Annual State of Agile Survey*, VersionOne, Apr. 2017; explore .versionone.com/state-of-agile /versionone-11th-annual-state-of -agile-report-2.
6. M. Kalenda, "Scaling Agile Software Development in Large Organizations," master's thesis, Faculty of Informatics, Masaryk Univ., 2017; is.muni.cz/th/410499/fi_m/masters _thesis.pdf.