

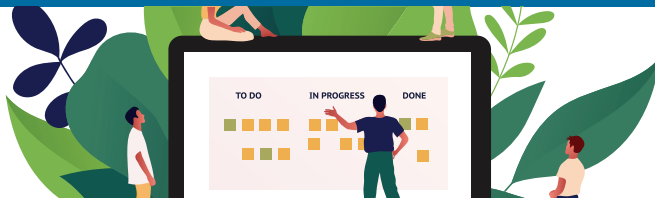
# Agile Scalability Engineering: The ScrumScale Method

Gunnar Brataas and Geir Kjetil Hanssen, SINTEF Digital

Nikolas Herbst, University of Würzburg

André van Hoorn, University of Stuttgart

*// Scalability is a property that must be carefully designed into a system. A case study in the largest Norwegian public portal, Altinn, illustrates how developers and scalability experts improved scalability and spent less time during scalability testing. With minor adjustments to an agile development process, stakeholders spend more up-front time together. //*



**EVERY YEAR IN** April, fingers are crossed at Altinn, the largest Norwegian public portal operator. Will the capacity be sufficient when 4 million

Norwegians want to investigate their tax reports simultaneously? At times, the entire Altinn portal crashes due to overload situations. This creates problems for other public services provided by the same portal, e.g., for customs when declaring all of

the trucks with salmon crossing the Norwegian border. An official investigation in 2011 following a severe portal outage concluded that neither scalability testing had been addressed properly nor had scalability requirements been captured.<sup>1</sup> To address this, Altinn has invested in scalability testing. This has improved scalability but turned out to be costly when severe issues are found close to release. In an effort to reduce costs and react faster to customer needs, Altinn is moving toward agile development with smaller and more frequent releases to production, thus creating the challenge of ensuring a rapid development process.

## Agile Methods

Although existing performance and scalability models, tools, methods, and guidelines are valuable, they are usually time-consuming and require considerable manual effort from skilled personnel.<sup>2</sup> “Fix it later”<sup>3</sup> remains the most common approach and imposes great risk along with slow and costly development. Agile methods such as Scrum enforce a rapid development process but do not address specifically how to deal with nonfunctional qualities, e.g., scalability.

In this article, we describe a lightweight extension to Scrum, named *ScrumScale*. ScrumScale manages the scalability requirements (concerns) of a system to achieve a faster development process. We discuss the lessons learned from applying ScrumScale during the development of a new building application system, one of the major innovations from Altinn in 2018. We follow the related case study from the early preparation stage to its transition from waterfall-oriented to an agile development process. We supplement these lessons learned

Digital Object Identifier 10.1109/MS.2019.2923184  
Date of current version: 20 August 2020

0740-7459/20©2020IEEE. TRANSLATIONS AND CONTENT MINING ARE PERMITTED FOR ACADEMIC RESEARCH ONLY. PERSONAL USE, IS ALSO PERMITTED BUT REPUBLICATION/REDISTRIBUTION REQUIRES IEEE PERMISSION. SEE [HTTP://WWW.IEEE.ORG/PUBLICATIONS\\_STANDARDS/PUBLICATIONS\\_RIGHTS/INDEX.HTML](http://www.ieee.org/publications_standards/publications/rights/index.html) FOR MORE INFORMATION.

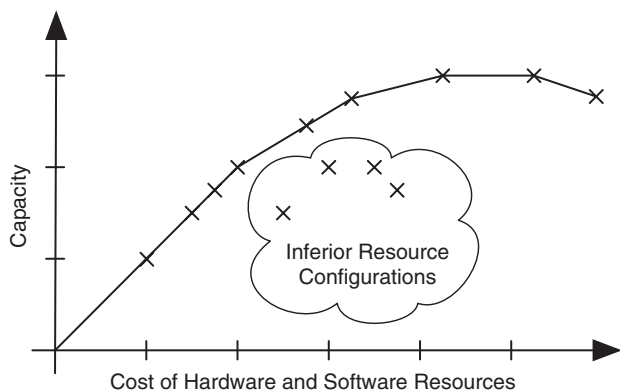


FIGURE 1. Scalability, relative to cost of resources and overall capacity.

## SCALABILITY

*Scalability* is defined as the ability of a system to increase its capacity by consuming more resources.<sup>4</sup> The *capacity* of a system refers to the maximum workload a system can handle within a given performance objective. Collectively, we term all factors required to describe the scalability of a system as *scalability concerns*:

- The performance objective is often measured by 90th percentile response times for the services within the system boundaries.
- The workload is the product of work and load.
- Work comprises what is done each time we invoke an operation and is related to the amount of data to be processed, stored, or communicated.
- The amount of data is related to specific work objects such as documents, drawings, images, movies, and so on.
- Operations identify basic work components of the system interface as different ways of interacting with the system, e.g., functions, calls, transactions, queries, and jobs.
- Load describes how frequently an operation is invoked and is often specified by an arrival rate such as transactions per second.
- Cloud and/or hardware resources are software as a service, infrastructure as a service, CPUs, disks, and networks. Software license costs are also part of the resources.
- Data consistency describes how up to date different replicas should be. Because replication and caching are key architectural patterns for achieving scalability, strict consistency requirements will make it more difficult.

and the detailing of ScrumScale with experience from other organizations and case studies.

Figure 1 shows the relationship between the cost of the resources involved and the capacity of the system. Initially, the capacity will often be proportional to the cost; this pertains to spending resources on bottlenecks, e.g., by adding CPU resources to the database server when database processing is the bottleneck.<sup>4</sup> (For more information on capacity, scalability, and scalability concerns, see “Scalability.”) For the same bottleneck, adding application servers will likely result in inferior resource configurations, as shown in Figure 1.

At some point the correlation between resource cost and capacity ceases to be linear. When trying to add more resources, we typically reach maximum system capacity. By adding even more resources, the overall capacity may decrease as well. The resulting relationship (Figure 1) between the amount of resources and the resulting overall capacity for humans is termed the *Ringelmann effect*<sup>5</sup> and is caused by either laziness or coordination overhead. Only the latter applies to computers.

Scalability problems are often deeply rooted in the system architecture and may be hard to tune away, e.g., using a centralized SQL database when a non-SQL document-based database with relaxed consistency requirements would be more appropriate. Such problems are important to spot early. Moreover, we want to avoid overengineering, where some parts of the system are gold plated with respect to scalability.

### Extending Scrum: ScrumScale

In the ScrumScale method shown in Figure 2, we propose extensions both

to normal sprints (steps 1–4 on the right-hand side) and a sprint 0 (steps A–C on the left-hand side). The key idea is to involve a scalability expert as early as possible both in the evaluation of user stories (e.g., “Which user stories may have scalability issues?”) and the evaluation of design ideas (e.g., “Will this design create scalability issues?”), in addition to the familiar code review and scalability testing. This “shift left” is a faster approach compared to building costly and time-consuming models for scalability analysis.<sup>2</sup> Scalability experts span different competences and, accordingly, different persons and existing roles, e.g., software architects and performance testers. One scalability expert (champion) may assist several teams, thus transferring expert scalability knowledge across the entire organization.

Sprint 0 is widely used to indicate the necessary preparations taken prior to regular sprints. We use sprint 0 to construct the initial product backlog through three steps: A, B, and C, as described in the following section.

### Step A: Define User Stories

The initial product backlog, i.e., a list of functional user stories, is defined by the product owner (who represents users and other internal stakeholders) in collaboration with the team.

### Step B: Scalability Triage

A scalability triage is an expert group meeting. Triage is a concept borrowed from emergency medicine, where a doctor quickly determines whether a person requires immediate treatment or can wait. A user story is tagged with a scalability concern if at least one of the following conditions is true

1. The user story affects functionality, which already has a scalability risk.
2. The user story invokes considerable processing, storage, or communication.
3. We may coarsely classify work and load as expected to be imposed by a user story as small, medium, and large, and performance objectives as loose, medium, or tough. If the relation between them seems nontrivial, we have a scalability risk.<sup>6</sup>

In this informal step, the work, load, and performance objectives are not specified in detail; therefore, considerable expertise is required. Only for user stories tagged with a scalability concern do we continue with the method.

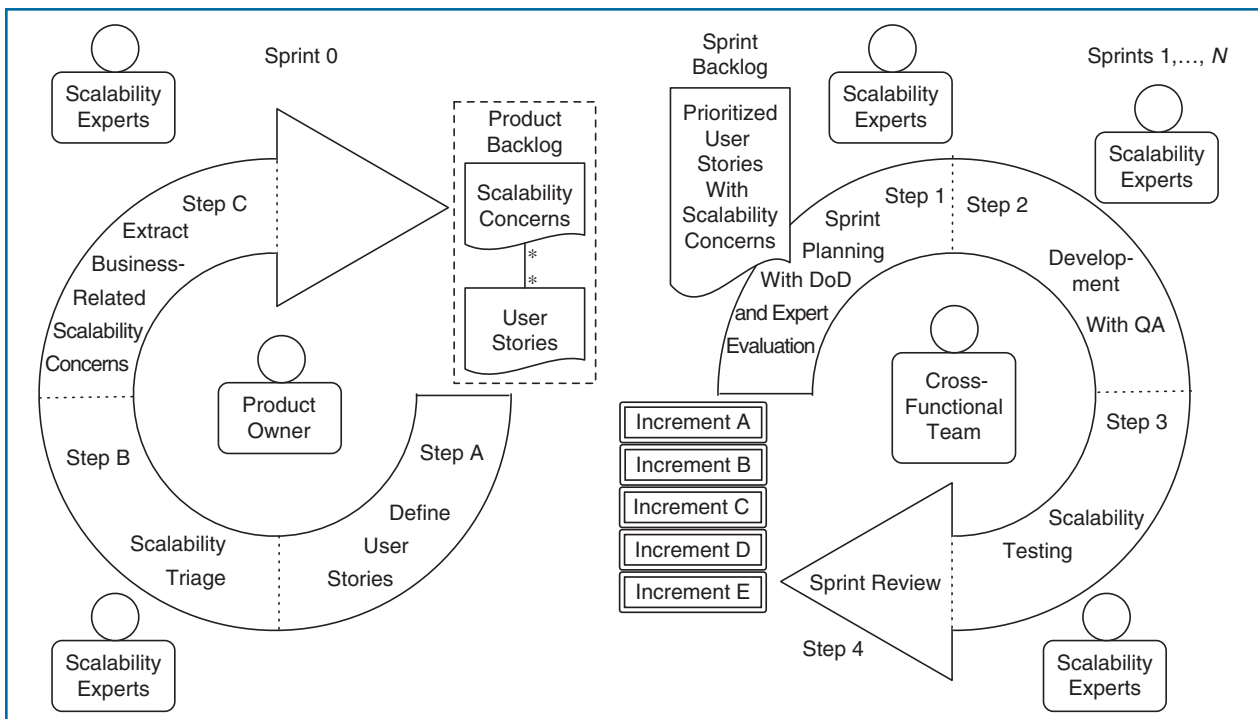


FIGURE 2. The ScrumScale method. DoD: definition of done.

### Step C: Extract Business-Related Scalability Concerns

When specifying scalability concerns, the following issues may be used as a guide:

1. Consider business plans and their connection to the projected workload, which may be represented in different scenarios that reflect the degree of increasing optimism, i.e., realistic, possible, and extreme.
2. The planning horizon describes how long into the future we want to explore the scalability of our system, e.g., two or five years from now on.
3. Clarify which components are within the system boundaries.
4. What seem to be the most important operations from a scalability point of view?
5. Which work objects will critically affect resource demands for these operations, while at the same time also vary in size?
6. Approximately what will the work parameters be that characterize the work objects, e.g., the average size and number of work objects?
7. Which types of users does the system have, and roughly how many are there of each type?
8. Approximately what will the load be?
9. Approximately which types of performance metrics will we use?
10. What are approximate performance objectives for the operations: 1 s or 1 min 90th percentile response time?
11. Which other systems, platforms, databases, or infrastructure will the system interact with? Particularly when a large workload is anticipated, these systems should get an early warning.
12. Are there relevant consistency concerns?
 

Having completed sprint 0, we have the initial user stories in the product backlog, including related scalability concerns. This is the starting point for the subsequent development sprints in steps 1–4, where scalability concerns are further refined, tested, and also potentially relaxed.

### Step 1: Sprint Planning With Definition of Done and Expert Evaluation

For the selected user stories tagged with a scalability risk, the scalability concerns captured in the previous step must be detailed iteratively, based on the following questions:

1. Have the system boundaries and the content of the operations changed? Are there new operations?
2. What will be the load during the busiest hour, on the busiest day, during the busiest week, and in the busiest year for our planning horizon? *Burstiness*, defined as the relationship between peak and average load, may be a helpful concept; e.g., if the busy hour load is three times the average hour load, then burstiness is three.
3. If documents are the critical work object (as in our case study), we must find the work parameters, i.e., what will be the maximum size and number of documents?
4. Currently, what are the strictest performance objectives and consistency concerns of the planning horizon for the critical operations?
5. For which critical operation can we see that the product of work and load gives a risk of not fulfilling the performance objectives?

6. Should we communicate with the product owner about relaxing some of these scalability concerns?

The answers to these questions are valuable inputs to the definition of done (DoD) as well as a scalability test plan. A spreadsheet may be used to derive load and work based on parameters, which are easier to estimate or measure. Uncertainty in these parameters may also be indicated.

For user stories tagged with scalability concerns, the scalability expert evaluates possible design patterns or detailed design ideas and gives feedback before coding begins. For instance, a specific design idea may impose a large number of heavy databases or network operations. The scalability expert may therefore advise the development team to redesign or consult with the product owner to reconsider the user story. This quality assurance (QA) at the design level differs from the scalability triage step, where the focus is on user stories. For scalability, it is critical to have a proper architecture that strikes a good balance between *laissez faire* and overengineering.

### Step 2: Development With QA

During (test-driven) development, scalability experts give feedback on ways to write scalable code, e.g., effective ways to write SQL queries and using indexes in database management systems. One positive side effect of this is that the scalability expert learns more about the system under development and may use this insight in preparing scalability testing (step 3).

### Step 3: Scalability Testing

Scalability testing requires considerable skills because we are typically faced with partial information about

the solution, test hardware, test data, and workload.<sup>7</sup> Ideally, testing done by the end of the sprint should be quick, and therefore automated.<sup>8</sup>

#### Step 4: Sprint Review

Results from the sprint-end scalability testing are used during the sprint review meeting where the product owner reviews the outcome and makes decisions on how to potentially refine the backlog.

During the planning and design stages, human expert knowledge is needed. The further the system enters into production, the more tools can be leveraged<sup>9</sup> but the higher the cost of change will be. When the system is in production, we obtain measurements for both the scalability and workload of the solution, which is useful for additional scalability analysis and development.<sup>9</sup>

### ScrumScale in the Building Application Case Study

We focus on the building application case study conducted at Altinn. A building application consists of documents describing projected buildings and its surroundings and is submitted to regulatory authorities. From 2016 onward, a new building application system for all Norwegian citizens was developed, initially using iterative waterfall with traditional requirements, before the Altinn organization employed two-week sprints in January of 2018. Ten different organizations were involved in setting requirements, making different parts of the service, modifying the platform, managing development, performing operations, and so forth. From the beginning, it was anticipated that larger attachments would require re-writing of the REST API in the Altinn platform. Aside from this, scalability

concerns were not formulated when SINTEF functioned as action researchers in March of 2017. By then, SINTEF had already learned during structured interviews with 12 different stakeholders in Altinn as well as the surrounding organizations that they did not elicit scalability concerns, even if this was vital. Based on experience from the CloudScale project,<sup>2</sup> we extracted a consistent description of scalability concerns.<sup>10</sup>

During meetings, we gradually increased our understanding of the scalability concerns. From March to June 2017, SINTEF conducted 19 video conference meetings lasting an average of 1 h with an average of 2.1 external stakeholders. A total of 50 work hours were spent by external stakeholders answering emails and follow-up questions during these meetings. Training sessions on the ScrumScale method was an integral part of these meetings. Moreover, ScrumScale has been formally introduced to the organization via Confluence online guidelines.

As Altinn's research partner, we applied principles from action research<sup>11</sup> to guide our collaboration with Altinn:

1. The Principle of the Researcher-Client Agreement: To become involved in the case as a key source of learning, we took an active collaborative role.
2. The Principle of the Cyclical Process Model (CPM): We worked in iterations with practitioners to diagnose, plan improvement actions, implement interventions, evaluate results, and reflect on what was learned.
3. The Principle of Theory: To identify proper interventions (improvement ideas), we consulted literature on agile development and scalability.

### Scalability Concerns

We detailed the scalability concerns in collaboration with Altinn:

- We began by focusing on two basic HTTP operations: posting attachments and putting (submitting) the complete building application. As we learned more about the building application workflow, in total, we found 10 different operations, three of which were critical. In addition to the two basic operations, we also identified the operation for getting (retrieving) the message box, which is a central/bottleneck component across many services.
- For load, we estimated 100,000 users per year; however, prior to regulatory amendments, it was observed that approximately 500 partial building applications per hour caused the old solution to crash. Altinn estimated twice this load during the busy hour.
- To estimate work, we started to analyze the data volume in six different building application types. Because it was difficult to estimate all of the required parameters, we ended up with two application sizes: an average of 0.6 GB and 70 MB for large applications and small applications, respectively.
- Initially, we formulated several performance objectives before we simplified them to one performance objective. The performance objective now reflected the timeout limit, i.e., a 90th percentile response time of 40 s.

### Scalability Expert Evaluation

The expert evaluation in step 1 gave the following key insights relevant to the scalability of the building application system:

- **Larger attachments:** When clarifying the scalability concerns, we found that attachments for one building application could in some cases be 6 GB, in contrast to a maximum 0.5 GB, as originally anticipated. Knowing this early could help better prepare the Altinn platform.
- **Data consistency:** An original requirement to change a supplier in the middle of the building application workflow would result in frequent and expensive database updates in the national Altinn portal. As a result, this requirement had to be relaxed in cooperation with the relevant stakeholders.
- **Neighbor notifications:** As part of a wider building application process, neighbor notifications would be sent out. This was not part of the original scalability concerns, but was discovered during iterative scalability concern collection. A straightforward process of sending out notifications with several large documents to mailboxes would jam them; therefore, a service hotel was designed for reading notifications on demand.

Note that all of these issues relate to imprecise, overly demanding, or missing scalability concerns. Discovering these issues early prevented code rewrites and delays for a nationwide, mission-critical public infrastructure.

### Positive Results

To evaluate the effects of the ScrumScale method in Altinn, we interviewed four key stakeholders late in 2018: one that was responsible for scalability in Altinn, the scalability tester, the project manager, and the product owner. Because ScrumScale

contains only a few extensions familiar to Scrum, the internal process shift was straightforward. Importantly, we found that scalability experts are involved earlier in the process, which is one of the main goals of ScrumScale. The responsible scalability expert explained:

*Developers and experts now use more up-front time together, resulting in less time spent at later stages for individual, unstructured rework.*

To improve the quality of scalability concerns, Altinn introduced recurring backlog refinement meetings every month that evaluate scalability vulnerabilities. Where appropriate, entire epics and user stories are tagged with a scalability risk. The affected user stories undergo an expert evaluation where a scalability expert looks at design sketches. Coupling developers and experts is particularly relevant to reduce hot-fixes near release—which has been a common practice earlier, and that has created a need for later rework. This is also considered to be beneficial for improving knowledge transfer from scalability experts to developers.

The scalability tester reported that he now receives input for the test plan, in particular data volumes (work), for which he previously had to request or even guess. He also confirms improved interaction with developers during the scalability triage, where all user stories have been evaluated for scalability implications. Because of the early focus on scalability during development, the effort on scalability testing of the building application could be relaxed.

Scalability testing in the building application case study did not provide any surprises and consumed 28 h of work time compared to the expected 80 h. More importantly,

the system is now in operation with no scalability issues.

Our interview with a product owner confirms the positive view of addressing scalability early in the process, which helps developers improve their competency and grow their responsibility regarding scalability.

In summary, the experience gained from using ScrumScale suggests improved communication and more transparency in managing scalability.

### Remaining Challenges

The most important current limitation is that the scalability tester finds it difficult to maintain the “big picture” with respect to scalability in the ongoing development. In practice, the tester (being a scalability expert) must respond to a stream of minor pull requests as part of code management to determine what has actually changed in the code. Earlier, with months between releases, the tester could analyze plans and prepare scalability tests more efficiently. The increased speed in the development process has resulted in sparse and fragmented documentation in different systems, which has allowed serious scalability risks to enter production. Although minimum documentation is a virtue in Scrum, ScrumScale must enforce improved documentation from developers to support good scalability testing.

The scalability tester should be even more involved in expert evaluation and code QA, thus enabling knowledge transfer to the development team so that they can discover and resolve similar issues. This will reduce the need for hot fixes by the scalability tester, who may introduce functional errors.

### Evolution of the ScrumScale Method

Back in 2015, we coined the idea of the ScrumScale method as a vision where scalability is introduced in



**GUNNAR BRATAAS** is a senior research scientist with the Process Innovation Group at SINTEF Digital. His research focuses on sound and practical methods for characterizing, assessing, and improving the scalability of information and computer systems. Brataas received a Dr.-Ing. for software performance engineering of information systems from the Norwegian University of Science and Technology in 1996. He is a Member of both the IEEE and ACM. Contact him at [Gunnar.Brataas@sintef.no](mailto:Gunnar.Brataas@sintef.no).



**NIKOLAS HERBST** is a research group leader at the chair of software engineering at the University of Würzburg. His research interests include predictive data analysis, elasticity in cloud computing, autoscaling and resource management, performance evaluation of virtualized environments, and autonomic and self-aware computing. Herbst received a Ph.D. from the University of Würzburg in 2018 and serves as elected vice-chair of the Standard Performance Evaluation Corporation Research Cloud Group. He is a Member of the IEEE. Contact him at [Nikolas.Herbst@uni-wuerzburg.de](mailto:Nikolas.Herbst@uni-wuerzburg.de).



**GEIR KJETIL HANSSEN** is a senior research scientist with the Process Innovation Group at SINTEF Digital. His research focuses on software engineering methodologies such as agile methods, software process improvement, and safety-critical systems. Hanssen received a Ph.D. in software engineering from the Norwegian University of Science and Technology in 2010. He recently coauthored the book *SafeScrum—Agile Development of Safety-Critical Software* from Springer. Contact him at [Geir.K.Hanssen@sintef.no](mailto:Geir.K.Hanssen@sintef.no).



**ANDRÉ VAN HOORN** is a researcher with the Institute of Software Technology at the University of Stuttgart. His research focuses on investigating challenges and opportunities in the context of continuous software engineering and DevOps as well as designing, operating, and evolving trustworthy distributed software systems that focus on quality attributes such as performance, reliability, and resilience. Van Hoorn received a Ph.D. from Kiel University in 2014. He is a member of ACM and serves in several roles within the SPEC Research Cloud Group. Contact him at [van.hoorn@informatik.uni-stuttgart.de](mailto:van.hoorn@informatik.uni-stuttgart.de).

the Scrum cycle. The ScrumScale method was inspired by SafeScrum<sup>12</sup> and similar ideas from which Scrum is extended with guidelines on how to manage nonfunctional requirements.<sup>13</sup> SafeScrum addresses challenges specific to functional safety and compliance with mandatory standards that are not relevant for scalability. We have, however, applied three core principles inherited from SafeScrum:

1. Scrum is a useful basis that emphasizes QA by frequent evaluation of outcome (sprint review) and frequent evaluation of requirements and design ideas (sprint planning).
2. Functional requirements and scalability concerns are separated but linked, to maintain the relationship. The relationship and potential impact that functional requirements may impose on

- scalability requirements are important to evaluate continuously.
3. Scalability experts, as a scarce resource, support the team on scalability decisions as early as possible during the development process (i.e., in sprint 0) and the consecutive sprints.

There are other approaches for handling nonfunctional requirements, such as Architecture Tradeoff Analysis


Method (ATAM);<sup>14</sup> however, ATAM is an extensive up-front analysis that relies on an existing architecture. Our goal has been to find a more continuous and lightweight approach.

Following our experiences with the case studies, we gradually evolved into the first version of the ScrumScale method.<sup>7</sup> Compared to this first version, we now introduce sprint 0. In addition, several steps have been clarified, e.g., the DoD in step 1 and the sprint review in step 4.

Aside from the building application case study, detailing of the ScrumScale method has also been strengthened by four more case studies in the years 2017–2019: 1) credit card accounting, 2) intraday energy trading, 3) open banking, and 4) authorization in Altinn. Overall, these case studies confirmed the usability of the ScrumScale method and helped to fine-tune portions of it. For instance, the third condition in the triage technique and different load scenarios was derived from open banking,<sup>13</sup> and partial scalability testing was derived from credit card accounting.<sup>7</sup>

Our work demonstrates that the effective consideration of scalability concerns in an industrial case study requires only minor adjustments to the development process, whereas the main change is the increased and more-frequent early dialogue between developers and experts. Clear concepts for scalability concerns make them easier to capture and follow throughout development, as shown in the building application case study. The presented engineering approach has evolved over the last few years and has already helped provide control and confidence over scalability in Altinn, a critical part of the Norwegian IT infrastructure. We believe that ScrumScale is

suitable for other systems with difficult scalability concerns that evolve in complex environments with multiple stakeholders.

Based on the lessons learned from our work, we see the need for further work on 1) coordinating different types of nonfunctional requirements, 2) ScrumScale in large-scale projects, and 3) ScrumScale applied in a DevOps context.<sup>15</sup> 

### Acknowledgments

This research was supported by the Norwegian Research Council under grant 256669 (ScrumScale), the German Research Foundation under grant KO 3445/11-1, and the German Federal Ministry of Education and Research under grant 01IS17010. We thank EVRY, Powel, and Altinn for contributing case studies, and Petter Braskerud (Altinn), whose contributions were pivotal to the success of the building application case study.

### References

1. Det Norske Veritas, “Vurdering av Altinn II plattformen (in Norwegian),” Det Norske Veritas, Hovik, Norway, Rep. 2011-1239, 2012. [Online]. Available: <http://kursinfo.himolde.no/in-kurs/IBE250/pensum/DNVAltinn.pdf>
2. S. Becker, G. Brataas, and S. Lehrig, *Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications*. New York: Springer-Verlag, 2017.
3. C. U. Smith and L. G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Reading, MA: Addison-Wesley, 2002.
4. G. Brataas, N. Herbst, S. Ivansek, and J. Polutnik, “Scalability analysis of cloud software services,” in *Proc. IEEE Int. Conf. Autonomic Computing (ICAC)*, 2017, pp. 285–292.
5. D. A. Kravitz and B. Martin, “Ringlemann rediscovered: The original article,” *J. Pers. Soc. Psychol.*, vol. 50, no. 5, pp. 936–941, 1986. doi: 10.1037/0022-3514.50.5.936.
6. G. K. Hanssen, G. Brataas, and A. Martini, “Identifying scalability debt in open systems,” in *Proc. ACM/IEEE Int. Conf. Tech. Debt*, 2019, pp. 48–52. doi: 10.1109/TechDebt.2019.00014.
7. G. Brataas, G. K. Hanssen, and G. Ræder, *Towards Agile Scalability Engineering, XP 2018*. New York: Springer-Verlag, 2017.
8. Z. M. Jiang and A. E. Hassan, “A survey on load testing of large-scale software systems,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 11, pp. 1091–1118, 2015.
9. H. Schulz, T. Angerstein, and A. van Hoorn, “Towards automating representative load testing in continuous software engineering,” in *Proc. Int. Conf. Performance Engineering (ICPE)*, 2018, pp. 123–126.
10. G. Brataas and T. E. Fægri, “Agile scalability requirements,” in *Proc. ACM Int. Conf. Performance Engineering (ICPE)*, 2017, pp. 413–416.
11. R. M. Davison, M. G. Martinsons, and N. Kock, “Principles of canonical action research,” *Inform. Syst. J.*, vol. 14, no. 1, pp. 65–86, 2004.
12. G. K. Hanssen, T. Stålhane, and T. Myklebust, *SafeScrum—Agile Development of Safety-Critical Software*. New York: Springer-Verlag, 2018.
13. A. Leffingwell, *Agile Software Requirements*. Reading, MA: Addison-Wesley, 2011.
14. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Reading, MA: Addison-Wesley, 2012.
15. A. Brunnert et al., “Performance-oriented DevOps: A research agenda,” SPEC, Gainesville, VA, Tech. Rep. SPEC-RG-2015-01, 2015.