Evidence-Based Guidelines for Advancing Continuous Experimentation

Ros, Rasmus; Runeson, Per; Bjarnason, Elizabeth

Published in:
IT Professional: technology solutions for the enterprise

DOI:
10.1109/MITP.2024.3397541

2024

Document Version:
Peer reviewed version (aka post-print)

Link to publication

Citation for published version (APA):
Ros, R., Runeson, P., & Bjarnason, E. (2024). Evidence-Based Guidelines for Advancing Continuous Experimentation. *IT Professional: technology solutions for the enterprise*, *26*(5), 20-27. https://doi.org/10.1109/MITP.2024.3397541

Total number of authors:
3

Creative Commons License:
CC BY-ND

Evidence-Based Guidelines for Advancing Continuous Experimentation

Rasmus Ros
Lund University and Theca Systems AB, Sweden

Elizabeth Bjarnason
Lund University, Sweden

Per Runeson
Lund University, Sweden

Abstract—
Continuous experimentation (CE) is used by many internet-facing companies to improve the value of their products based on user feedback gathered, e.g. through on-line experiments using A/B testing. Frameworks and theories for CE have been derived through academic research from applications in large internet facing companies. To assist practitioners in a broader range of companies, we herein present guidelines for CE, based on empirical data from interviews with 27 practitioners at 12 companies of varying sizes and CE maturity. The guidelines include advise derived from our (previously published) theory of factors that affect CE (FACE). These practitioner guidelines may assist companies in making informed decisions concerning their CE practices and contribute to efficient and effective experimentation. Our advice includes building processes and infrastructure gradually, combining quantitative and qualitative methods, focusing on goals and incentives for CE, and being mindful of ethics.

CONTINUOUS EXPERIMENTATION (CE) is used by many companies with internet-facing applications to optimise their value offering, based on user feedback. CE is an experiment-driven software engineering approach where assumptions about markets, customers, business models, product features, and requirements are continuously tested through experiments with users. The aim is to ensure offering best possible value to the users [1].

A/B testing is one method that may be used in CE, where two versions of a product, A and B, are exposed to different users and the experiment measures a desired user behavior, e.g. clicks, to assess the "better" option w.r.t. desired goals. Mostly, users do not notice that they are part of experiments, and sometimes the differences between the versions under test are negligible, as in the infamous experiment performed by Google to find out which of 41 shades of blue to use for clickable links.

The advancement of CE is driven by industrial applications to a large extent. Already in 2007, Kohavi et al. [2] published an experience report on experimentation at Microsoft and provided guidelines on how to conduct randomized controlled experiments in software engineering practice. Several industry leaders have adopted the practice of CE and published infor-

Digital Object Identifier 10.1109/10.1109/MITP.2024.3397541
Date of publication preprint; date of current version 06 maj 2024

mation on their CE processes or infrastructure, such as Microsoft [3], Google [4] and Facebook [5].

Academic researchers have generalized the experience from industry reports and case studies from different perspectives, such as the process and activities for CE, and factors that affect CE. Fagerholm et al. [6] defined the RIGHT process model and infrastructure architecture. Fitzgerald and Stol [7] positioned the activities in a reference model that integrates CE into business, development, and operations. Our previous research complements these process-oriented views with a theory of *factors affecting continuous experimentation* (FACE) [1] that explains why CE is more or less effective for different contexts. The FACE theory is summarized in Figure 2.

There is an increasing need for guidelines on how to implement CE in practice. This is due to the wider adoption of CE in a range of companies of varying sizes, domains, and business models, where the general approaches reported from companies with millions of users may not be applicable as-is. To meet this need, and to reach beyond existing academic research towards guiding practitioners, we here provide guidelines based on empirical evidence for *advancing towards best CE practices.* While Schermann et al. [8] guide the technical implementation of CE, we take a broader scope and *address engineering, organizational, and business aspects of the uptake and evolution of CE for companies of different sizes*. We include companies in the business-to-business (B2B) domain, as well as, the more commonly explored business-to-customer (B2C) domain. An overview of our research approach is presented in the sidebar *Research approach.*

Research approach

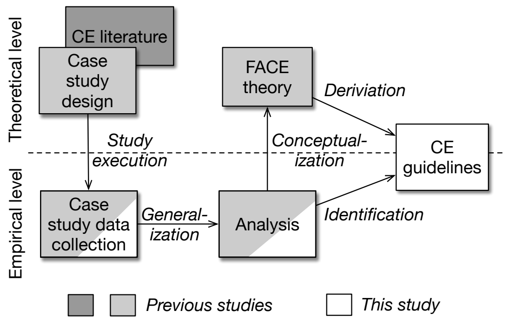

Figure 1. Overview of our research approach, where the previous studies [9], [1] are shaded.

Our overall research approach follows Stol and Fitzgerald's principles for theory-oriented software engineering [10], and is depicted in Figure 1. Previous studies comprised a review of *CE literature* [9] and an interview based *multi-case study*, resulting in the *FACE theory* [1], see Figure 2. The *data collection* included semi-structured interviews with 27 industry practitioners in 12 companies selected through job listings on CE topics, see Table 1. We *analyzed* the interviews through thematic coding where codes were clustered into themes, and then performed cross-case analysis of these themes. For further details, see Ros et al. [1].

The interviews also revealed examples of successful CE practices and experiences from the companies' introduction of those. For this study, we *identified* these examples of best CE practice descriptions by clustering similar approaches across companies. Further, we *derived* practitioner guidelines on which strategies to apply to achieve best practice CE by analytically validating recommendations from the interviews in relation to the factors in the FACE theory [1] and the RIGHT model [6].

CE in a Nutshell

CE is essentially a continuous process of launching randomized controlled experiments on a company's market offering (product, service, or feature), to optimize the utility for users and thereby the business value for the provider. At the core is the experiment process, summarized in five major steps: 1) ideation, 2) implementation, 3) experiment design, 4) execution, and 5) analysis [6].

Hypotheses that are to be tested in experiments are defined during *ideation* in terms of a desired outcome, e.g. in e-commerce contexts typically increased sales. These hypotheses may range from the above mentioned shades of blue in clickable links, to variants of user interfaces or information elements to be presented to the users. For most companies, more hypotheses are generated than for which there are resources available for experiments, and thus they must be prioritized.

Next, the product or feature under experimentation is *implemented*. Either two or more new versions are compared, or a new version is compared with the existing one. Furthermore, experimentation infrastructure is needed to control version handling, randomized distribution of users across experimental subjects, and data collection mechanisms to monitor the experiment.

The *experiment design* involves choosing a design for the specific experiment, e.g. A/B test. The design also involves calculation of the power of the experiment—i.e. the likelihood of a hypothesis test detecting a true effect, if there is one—given the amount of data collected and the duration of the experiment. Using proxies for the desired outcome metric may be a design option, where more experimental data, and thus power, is traded against less precision of the experiment. For example, the time of user activity or number of clicks in the web-shop may be used as proxies for sales.

The experiment *execution* involves releasing the features under experimentation for usage. The experiment infrastructure then assigns users to each experiment version and continuously monitors the outcomes. The data can be continuously *analyzed* and the experiment adapted after each iteration. Depending on the final outcome, one of the versions is selected for all users, or new experiments are defined and launched.

In case major changes are needed beyond the scope of sequences of experiments, a *pivot* may be launched. A pivot may involve changes to the product or service, but could also entail changes to, e.g. revenue model or sales channels. Pivots have a larger scope and are to lesser extent based on evidence compared to a change involved in an experiment.

The roles involved in experimentation range across software engineering, data science, and business analysts. This is a consequence of CE bridging the engineering and the business sides of a company.

Frameworks for CE

There are several frameworks describing different aspects of continuous experimentation (CE), including the RIGHT *process model* [6], the continuous software engineering *activities model* [7], and our FACE *theory* [1]. Together, they provide condensed summaries of current knowledge of CE.

The RIGHT Process Model

Fagerholm et al.'s RIGHT process model for CE provides a reference model for activities and technical infrastructure needed in CE [6]. It is based on the *Build–Measure–Learn* cycle of the Lean Startup methodology, which in turn is rooted in the Shewart and Deming cycles on quality improvement from the 1930's and 1950's, respectively. The *Build* activities identify and prioritize the hypotheses to be tested in an experiment. A new version of the system is then implemented for the hypothesis selected for evaluation. The *Measure* activity executes an experiment to evaluate the hypothesis. In the *Learn* activity, collected user data is analysed. The analysis underpins decision making on the outcome of the experiment.

In the RIGHT model, Fagerholm et al. propose a technical infrastructure comprising a back-end system to drive the experiments, analytics tools, and features to handle multiple versions and assignment of users accordingly.

Continuous Software Engineering Activites

Fitzgerald and Stol [7] defined an activity framework, named *Continuous* * (star), consisting of 16 activities, among those continuous integration and deployment (CI/CD). They clustered the activities around three major processes in a company, namely *Business strategy, Development* and *Operations* processes. *DevOps* refers to the continuous interplay between software development and the operations. Further, they add the concept of *BizDev* to denote a similar continuity between business strategy and development. Thereby, their framework clarifies the role of CE, providing feedback from operations to development and business strategy.

FACE Theory on Success Factors

Our FACE theory [1] complements the frameworks above and presents empirically observed factors that contribute to an organization's ability to gain value through CE. Figure 2 shows an overview of the theory constructs (**C1–C6**), i.e. key concepts in CE, and propositions (**P1–P6**), i.e. claims about relations between constructs. The theory is a condensed summary of our findings from empirical observations.

FACE theory states that the value of CE for a company is gained through *effectively conducting experiments* that enable improving the problem–solution fit (**P2**) and/or the *product–market* fit (**P3**). Problem–solution fit is a measure of how well the solution solves users' needs and product–market fit is how well the product fits the market. Product–market fit is more challenging to address and measure in experiments, e.g., due to the need to position the product towards competitors or making sure the product is visible for customers. In practice, actual sales figures is a

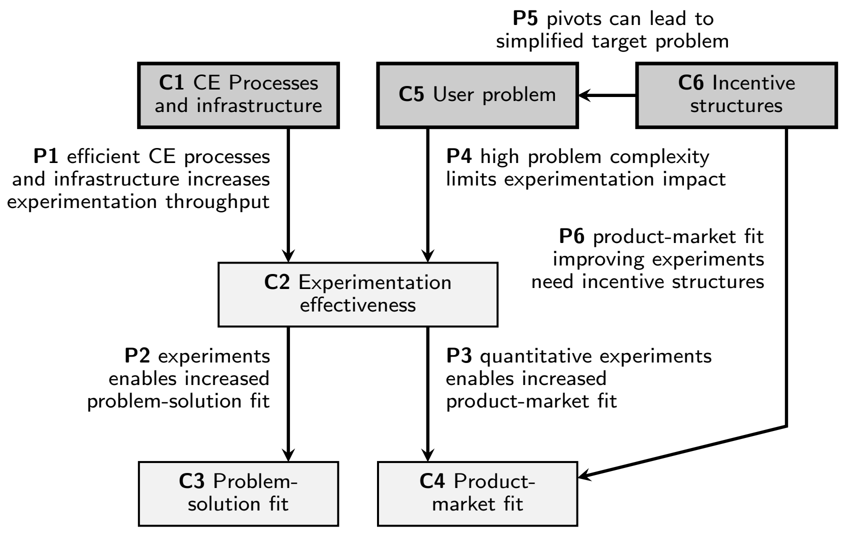
P5 pivots can lead to simplified target problem
C1 CE Processes and infrastructure
C5 User problem
C6 Incentive structures
P1 efficient CE processes and infrastructure increases experimentation throughput
P4 high problem complexity limits experimentation impact
P6 product-market fit improving experiments need incentive structures
C2 Experimentation effectiveness
P2 experiments enables increased problem-solution fit
P3 quantitative experiments enables increased product-market fit
C3 Problem-solution fit
C4 Product-market fit

Figure 2. An overview of the Factors Affecting Continuous Experimentation (FACE) theory, reprinted from Ros et al. [1]. The boxes represent the *constructs* (**C1–C6**) and the arrows the *propositions* (**P1–P6**) of our theory. The top boxes in dark grey correspond to the *root causes* derived from the theory (**C1**, **C5**, and **C6**).

good indicator of product–market fit, while problem–solution fit can be more easily observed through users' interactions with the product.

In essence, the theory states that the effectiveness of experiments can be increased through:

- efficient processes and tools for CE (**P1**),
- addressing a user problem that is sufficiently simple to be measurable (**P4**),
- pivoting the business model to simplifying said problem complexity (**P5**), and by,
- selecting a business model that provides incentives to conduct experimentation (**P6**).

The theory points out three main *CE factors* that are intrinsic root causes of an organization's ability to perform experimentation. These are marked with dark grey in Figure 2 and correspond to the constructs **C1** *CE processes and infrastructure*, **C5** *user problem*, and **C6** *incentive structures*. Thus, these factors represent hurdles that companies need to overcome to reach efficient experiment-driven development.

CE Implementation in Case Companies

The companies in our study are of widely varying sizes, types of business models (B2B, B2C), and business domains, and have widely varying expertise and extent of CE implementation, see Table 1. We observe differences in CE implementation between companies with advanced CE processes and infrastructure, and those without. Our empirical research provides a complementary view of CE compared to those provided by the industry leading companies [3], [4], [5] and academic reference models [6], [7].

Two of the case companies in our study have achieved the highest degree of CE implementation, namely *Cases D and G*. They conduct experiments frequently and on all parts of their software, including most software changes, and use business relevant metrics. CE is an important and integrated part of the software development processes of these companies. In addition, these companies have shifted from a mindset of shipping features as-is, to an experiment-driven approach where only functionality proven to solve user or business needs is released. These companies are able to affect both problem–solution fit and product–market fit with their CE practices.

The fact that the remaining companies in our multi-case study do not experiment to the same degree can be explained by the FACE factors. There are companies with business models that do not provide *incentive structures* (**C6**) for direct product–market fit improvements (Cases A, B, C, F, J, and L), and thereby are not inductive to making CE a priority. Some companies have under-developed *CE processes and infrastructure*

Table 1. Overview of our 12 case companies, containing business model, size, and state of CE, reprinted from Ros et al. [1]. Small companies have less than 50 employees, medium have less than 250, large companies have less than 1000 employees, and huge more than 1000 employees. CE expertise and extent is estimated based on our interviews using the ordinal scale: low, medium, and high. More details about the companies can be found in our earlier work [1].

	Business Model			CE Implementation	
Case	**Summary**	**Type**	**Size**	**Expertise**	**Extent**
A E-commerce algorithms	Product with direct sales	B2B	Small	High	Medium
B Local search	Service selling placement to B2B customers and serving ads to B2C	B2X	Small	Medium	Low
C E-commerce consultants	Consulting on a per project basis	B2B	Medium	Low	Low
D Video streaming	Freemium service with premium features	B2C	Medium	High	High
E Web shop	Web shop for subscription to physical goods	B2C	Huge	Medium	Medium
F Customer relations	Product and service with direct sales	B2B	Small	Low	Low
G Engineering tools	Freemium service with premium for larger teams	B2B	Huge	High	High
H Web shop	Retailer with small web shop	B2B	Large	Low	Low
I Web shop	Web shop and retail	B2C	Huge	High	Medium
J Product information	Service selling customer information and leads to B2B and free for B2C	B2X	Medium	Medium	Low
K Business intelligence	Product with direct sales	B2B	Large	Medium	Low
L Employee management	Service with direct sales	B2B	Small	Low	Medium

(**C1**) (Cases E, H, and I), such that experimentation is a daunting task. Finally, some companies address a complex *user problem* (**C5**) (Cases K and L) which makes it hard to quantify the user behavior in an experiment.

Evidence-Based Guidelines for CE

We present the following guidelines on how to make CE provide the most value to a company and its customers. The guidelines have been identified and derived from our empirical evidence and are grounded in our FACE theory. The guidelines are categorised into *engineering, organizational*, and *business* aspects. Our guidelines provide recommendations for practice beyond the basic principles and activities provided by the RIGHT and Continuous * frameworks [6], [10], summarized above. While the guidelines are based on a broad set of case companies, we advise practitioners to consider and adapt these guidelines to their specific context.

Engineering Aspects

- *Feasibility analysis* of experiments should be done before experiments are conducted. Companies with High experimentation expertise (Cases A, D, I, and G) conduct this step. This analysis could be as simple as checking how many users use a feature or a power analysis to control false discovery rates [11].
- *Conduct rigorous analysis* after experiments are completed. Only the cases with High expertise in CE, monitor statistical significance on multiple metrics of different types: sales metrics, user experience metrics, and computation performance metrics. The calculations are simple [11], so there is no reason to ignore statistical significance (or effect sizes, or Bayesian alternatives). The cases with Low expertise (C, H, and L) skip this step. Rigorous qualitative analysis is performed at Case K only, with a transcription and coding process.
- *Gather multi-faceted evidence* with mixed-methods experiments. Mixed-methods experiments, as demonstrated at Cases D and G, harness the strengths of both quantitative and qualitative data to provide complementary insights. Other companies use one or the other. Prototyping pre-deployment experiments can be evaluated qualitatively, to pre-validate ideas and to find hypotheses on what to experiment on. Following up with post-deployment experiments give definitive evidence. Though, it does require additional roles in the form of both user experience (UX) researchers and data scientists to conduct analysis of the results.
- *Use methods that fit the development process* to experiment in contexts with high user problem (**C5**) complexity. Qualitative experiments are much easier to implement since they do not require high numbers of users and hence lower processes and infras-

tructure requirements (**C1**), nor require specifying a quantitative metric. Cases K and L are able to conduct CE by validating software changes using qualitative experiments. At Case K, they include standardized qualitative experiments as part of their release cycle so that they can compare to previous releases.

Organisational Aspects

- *Encourage knowledge sharing*, for example, in workshops, mails, or internal wikis. One of the strengths of an experiment-driven development method is the opportunity to create domain knowledge through rigorous research. That effort risks being wasted if the knowledge obtained from experiments is not curated. Companies with Medium or High CE expertise have these practices in place and it seems to be especially important when the experimenters are in a stand-alone team.
- *Prioritize developing infrastructure* (**C1**) to increase experimentation throughput (**C2**). The companies with High CE expertise have: 1) *data infrastructure* to support telemetry on all parts of the software stored in a way that can be used for experiments, 2) a central *experimentation platform* that enables advanced functions such as, parallel experiments, making inquiries into results for different user segments or metrics, and segmentation of users in order to obtain high throughput of experiments, and 3) finally, the *competences* to cover the necessary roles of experimentation and all software engineers need to be trained in experimentation so that they can partake in CE. Data infrastructure is especially challenging (observed at Cases B, H, and I) due to the overarching scope of the required data aggregation.
- *Build processes and infrastructure gradually (**C1**).* While some companies are able to get started and ramp up experimentation due to prior investments in data infrastructure (such as Cases A and to some degree B), we recommend starting with experimentation practice instead of infrastructure processes. Starting with experimentation is easy, which is illustrated by Case D, where they implemented their first version of their experimentation platform in an off-hours Hackathon. However, scaling to both high frequency and high extent of experimentation is very challenging. The knowledge obtained from continuous experimentation will also be useful for continuously scaling up the necessary infrastructure.
- *Be mindful of ethics* when providing strong incentives to individuals (**C6**). Team goals should be set with collaboration between the teams and management in mind, to ensure the goal is good for both business and users. For example, one of the experiments mentioned in Case E was to hide the pause subscription button in a menu, which drastically improved the churn rate and revenue generation. This also highlights the need for knowledge sharing since other developers from the company were able to point out the ethically questionable experiment.

Business Aspects

- *Prioritization should be* based on business relevant user data. Companies use scoring methods to quantitatively prioritize expected impact on business and difficulty of implementation. Both Cases D and G use the Impact, Confidence, Ease (ICE) scoring model [12] and use user data to base the numbers on, when possible.
- *Seize opportunities to pivot* that simplify the problem the software solves for users. User problem (**C5**) complexity severely limits the ability for experiment-driven development. Case D was able to pivot successfully and was able to change to an experiment-driven process as a consequence of their pivot from having business users to having consumer users and a subscription based revenue model. However, pivoting is likely a challenging endeavor. Failed examples of pivoting might go unnoticed in the software engineering literature, because a likely outcome of a failed pivot is bankruptcy or a cancelled product.
- *Find relevant goals* to experiment on that makes sense for both business and users. Metrics are hard to specify since they should resist unintentional and intentional attempts to game it, i.e., an improvement in the metric should always result in positive business and user value. For example, clicks are easy to increase by adding additional steps users must complete, but that could result in a worse user experience. Many case companies are only able to use user experience metrics (such as clicks) that can improve problem–solution fit (**C3**). However, by targeting product–market fit (**C4**) both users' and business' value can be optimized for simultaneously. Companies should put effort into finding a relevant goal, for example, by carefully using proxy metrics, using qualitative methods, or by providing

team goals. This is one aspect of having incentive structures (**C6**) suitable for experimentation; aligned between users and business value.

- *Provide incentives structures (**C6**) with team goal metrics* to encourage all software engineers to start with experimentation and give them responsibility for the experimentation process (**C1**). This is easy to implement when working with quantitative experiments since the developers have the necessary tools. Cases A, D, I, and G all use team goals. Companies that lack the right incentive structures can compromise with process-oriented goals, for example, to increase the percentage of new features that can be evaluated with experiments.

Conclusion

Continuous experimentation is a wide-spread practice—initially more among large companies, but gradually being adopted also by other kinds of businesses. While there is academic knowledge, based on observations of practice and experience reports, there is less on practitioner guidelines for making strategic decisions regarding CE practices. For this reason, we derived and provide these guidelines.

We advise companies to consider the guidelines in their strategic decisions related to business strategies, development processes, and operations, all of which affect the value which can be obtained from continuous experimentation. To achieve *problem–solution fit* and *product–market fit*, the interplay within *DevOps* and *BizDev* respectively must be achieved, as well as, an alignment between the two [7]. Problem-solution fit aims to satisfy users while product-market fit enable business goals, together creating long-term value for users and companies.

Acknowledgments

We thank all the participating anonymous companies and interviewees for their contribution to these guidelines. This work was supported by a grant from the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

REFERENCES

1. R. Ros, E. Bjarnason, and P. Runeson, “A theory of factors affecting continuous experimentation (FACE),” *Empirical Software Engineering*, vol. 29, no. 21, 2024.
2. R. Kohavi, R. M. Henne, and D. Sommerfield, “Practical guide to controlled experiments on the web: Listen to your customers not to the HiPPO,” in *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, ser. KDD, 2007, pp. 959–967.
3. S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, “The anatomy of a large-scale experimentation platform,” in *Proceedings of the 15th International Conference on Software Architecture*, ser. ICSA, 2018, pp. 1–109.
4. D. Tang, A. Agarwal, D. O’Brien, and M. Meyer, “Overlapping experiment infrastructure: More, better, faster experimentation,” in *Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining*, ser. KDD, 2010, pp. 17–26.
5. D. G. Feitelson, E. Frachtenberg, and K. L. Beck, “Development and deployment at Facebook,” *IEEE Internet Computing*, vol. 17, no. 4, pp. 8–17, 2013.
6. F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, “The RIGHT model for continuous experimentation,” *Journal of Systems and Software*, vol. 123, pp. 292–305, 2017.
7. B. Fitzgerald and K.-J. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.
8. G. Schermann, J. Cito, and P. Leitner, “Continuous experimentation: Challenges, implementation techniques, and current research,” *IEEE Software*, vol. 35, no. 2, pp. 26–31, 2018.
9. F. Auer, R. Ros, L. Kaltenbrunner, P. Runeson, and M. Felderer, “Controlled experimentation in continuous experimentation: Knowledge and challenges,” *Information and Software Technology*, vol. 134, p. 106551, 2021.
10. K.-J. Stol and B. Fitzgerald, “Theory-oriented software engineering,” *Science of Computer Programming*, vol. 101, pp. 79–98, 2015.
11. R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, “Controlled experiments on the web: Survey and practical guide,” *Data Mining and Knowledge Discovery*, vol. 18, no. 1, pp. 140–181, 2009.
12. S. Ellis and P. M. Brown, *Hacking growth: how today’s fastest-growing companies drive breakout success*. New York: Crown Business, 2017.

Rasmus Ros received his doctorate on software engineering at Lund University, Sweden. His current employment is with Theca Systems AB, as a senior Data Scientist. His research interests are data-driven optimization of software using machine learning and experimentation. Contact him at rasmus.ros@theca.com.

Per Runeson is a professor of software engineering at Lund University, Sweden. His research interests include empirical research and collaboration with industry on software development and management methods. He is particularly interested in studies on testing, continuous experimentation, and the role of open source and open data in software engineering. Contact him at per.runeson@cs.lth.se.

Elizabeth Bjarnason is an associate professor (docent) of software engineering at Lund University, Sweden. She performs empirical research and collaboration with industry on software practice, and in particular on requirements communication and modern RE practices such as prototyping and continuous experimentation. Contact her at elizabeth.bjarnason@cs.lth.se.