# The HURRIER process for experimentation in business-to-business mission-critical systems

Citation for the original published paper (version of record):
Issa Mattos, D., Dakkak, A., Bosch, J. et al (2023). The HURRIER process for experimentation in business-to-business mission-critical systems. Journal of Software: Evolution and Process, 35(5).
http://dx.doi.org/10.1002/smr.2390

(article starts on next page)

Received: 30 October 2020 | Revised: 28 June 2021 | Accepted: 16 September 2021

DOI: 10.1002/smr.2390

SPECIAL ISSUE - EMPIRICAL PAPER

Journal of Software: Evolution and Process

WILEY

# The HURRIER process for experimentation in business-to-business mission-critical systems

David Issa Mattos[1] | Anas Dakkak[2] | Jan Bosch[1] | Helena Holmström Olsson[3]

[1]Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

[2]Ericsson AB, Stockholm, Sweden

[3]Department of Computer Science and Media Technology, Malmö University, Malmö, Sweden

**Correspondence**
David Issa Mattos, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden.
Email: davidis@chalmers.se

**Funding information**
Knut and Alice Wallenberg Foundation; Software Center

## Abstract

Continuous experimentation (CE) refers to a set of practices used by software companies to rapidly assess the usage, value, and performance of deployed software using data collected from customers and systems in the field using an experimental methodology. However, despite its increasing popularity in developing web-facing applications, CE has not been studied in the development process of business-to-business (B2B) mission-critical systems. By observing the CE practices of different teams, with a case study methodology inside Ericsson, we were able to identify the different practices and techniques used in B2B mission-critical systems and a description and classification of the four possible types of experiments. We present and analyze each of the four types of experiments with examples in the context of the mission-critical long-term evolution (4G) product. These examples show the general experimentation process followed by the teams and the use of the different CE practices and techniques. Based on these examples and the empirical data, we derived the HURRIER process to deliver high-quality solutions that the customers value. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

KEYWORDS

continuous experimentation, experimentation process, business-to-business, mission-critical systems

## 1 | INTRODUCTION

Companies are expected to continuously deliver fast, high-quality software that provides value to customers. For example, mobile networks are constantly evolving in the telecommunication domain to support new user equipment and improve service quality. Additionally, the deployed software is becoming increasingly complex and has a high degree of interdependence with the operating environment. These aspects make it hard for the development organization to evaluate the delivered value and the quality of the software.[1,2]

With the success stories from the web-facing systems,[1,3,4] organizations in other domains have also been adopting continuous experimentation (CE) practices.[5–7] CE is used not only to validate the value delivered to customers but also to assess quality aspects that cannot be verified during internal development and pre-deployment quality assurance activities.[8] CE has been primarily focused on Software-as-a-Service and web-facing systems, in both research and industry.[5] Despite a few papers that explored the introduction of CE in a business-to-business (B2B) context,[9–11] no publications studied the industrial usage of a CE process in B2B mission-critical systems.

Fowler defines mission-critical systems as those that, in the presence of failures or degradation, can lead to property damage, reputation damage as well as to prevent the main task from being completed.[12] Fowler[12] also points that such systems are often subject to regulations and standards (e.g., the 3GPP specification[13]). Mobile communication is an integral part of payment and banking systems, medical and transportation devices, and others which, if disrupted, can lead to severe-major failures for different businesses and society.[12] Experimentation in the B2B mission-critical systems domain has different characteristics compared to most web-facing applications. For example, there is a difference between customer and users, ownership of the product and the data, who has control over new deployments, service level agreements, risk analysis, and the impact of failure and strategies to overcome them, among others. In the mobile communication domain, the mobile operators control which version of the software will be deployed and how and when the deployment will occur. These decisions are based on the risks and benefits of the new software version. Some of the risks can be mitigated by controlled deployments. Therefore any experimentation activity requires in-depth collaboration between the development organization and the customers.

To investigate the use of CE in B2B mission-critical systems, we conducted a case study in collaboration with Ericsson. Ericsson is a multinational networking and telecommunications company, with an R&D organization of over 24000 people[1]. Ericsson is arguably one of the largest software development companies operating in the B2B domain. By observing the CE practices of different teams, we were able to identify the key activities and derive an experimentation process that addresses the deployment of experiments with customers in the B2B mission-critical systems.

This paper provides four main contributions. First, we provide a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems. These techniques are discussed in the context of existing research in other domains. We identified four types of experiments that are conducted: business-driven, regression-driven, optimization, and customer support experiments. Second, we present and analyze four examples, one for each of the four types of experiments. These examples show the general experimentation process followed by the team as well as the usage of the different practices and techniques. Third, based on the empirical data, we derived the HURRIER process (**H**igh val**U**ed softwa**R**e th**R**ough cont**I**nuous Expe**R**imentation). This process combines different experimental techniques and practices to deliver high-quality solutions that the customers value. The HURRIER process can help the R&D organization to validate feature functionality, increase coverage, identify and trace stochastic faults and increase the confidence in the developed solutions much faster than without the field experiments. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

This paper is an extension of the paper "Experimentation for Business-to-Business Mission-Critical Systems: A Case Study"[14] presented at the International Conference on Software and Systems Process 2020. This paper provides the following contributions compared to the conference paper. First, we provide a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems (Section 4). Second, we provide a revised version of the HURRIER process to include relevant information to complement the different types of experimentation (Section 6). Third, in addition to the original example (in regression-driven experiments), we added three new examples for the other types of experimentation (Section 5). Finally, we have revised and expanded the background and discussion sections in light of the new research questions.

The rest of this paper is organized as follows. Section 2 presents background information in CE and related work in CE in the B2B domain. Section 3 describes the research method and validity considerations. Section 4 presents a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems. Section 5 presents four examples, in the context of the Long Term Evolution (4G) product, of different types of experiments as well as the practices and techniques. Section 6 presents the derived the HURRIER process. Section 7 presents a discussion on the results addressing each of the research questions. Finally, Section 8 concludes the paper.

## 2 | BACKGROUND AND RELATED WORK

### 2.1 | Continuous experimentation

Fitzgerald and Stol[6] provide a review on several different initiatives around the term continuous. They present a holistic view of the various software development activities throughout the entire software life cycle. These activities are divided into three phases: business strategy and planning, development, and operations. Continuous experimentation acts as a link between the strategy and operations and the development, where repeated cycles of build, measure, and learn[15] guide the product improvement, evolution, and innovation inside the company.

CE aims at minimizing the risk of developing software that does not deliver value to the customer through continuously identifying, prioritizing, and validating critical product assumptions during all development phases.[7] A significant amount of research has been conducted in the context of continuous experimentation. Auer and Felderer[5] performed a systematic mapping study in CE from 2007 to 2017 and identified a total of 82 publications. Their work identified that most of the research with industry participation is in the context of web-facing companies such as

[1] https://www.ericsson.com/en/about-us/company-facts

Microsoft, Yandex, Facebook, Google, and LinkedIn. Additionally, CE is discussed mainly through randomized controlled experiments (A/B testing). However, CE constitutes a group of techniques that goes beyond randomized controlled experiments,[16,17] and that can encompass many other activities and techniques.

Auer et al[18] provide a taxonomy synthesized from a systematic literature review on the characteristics of experiments in the online domain. These characteristics are divided into five themes overall, analysis, ideation, execution, and design. These characteristics reflect on decisions during a complete A/B testing iteration, such as selection of metrics, scope and hypothesis definition, telemetry and alter conditions, segmentation, number of participants, and risk evaluation.

Schermann et al[8,16] identify two groups of goals when conducting CE activities. The first group, a business-driven experiment, aims to evaluate the delivered value, the development ideas, business changes, and design decisions. The second group, regression-driven experiments, aims to identify and mitigate the impact of software changes in existing behavior, functional and non-functional bugs that evaded or can not be detected in the pre-deployment quality assurance activities, and scalability issues.

A critical aspect of CE in the business-to-business domain is the difference between customers and users. Customers acquire or subscribe to a product or service for the users.[7,9] In the business-to-customer domain, the customers are also users who generally acquire or subscribe to the product for themselves. Therefore, in the B2B domain, vendors usually sell products and services to other companies that sell products or services to users. A distinctive factor is that user data, product usage, and user feedback are not readily or easily available for the vendors without prior agreements. This can restrict the data collection, user feedback, and even new deployments aimed at product improvement.

Yaman et al[11] describe the process of introducing continuous experimentation in companies with an established development process using two company cases with pure software products, Ericsson and a digital business consulting company. The study investigates the introduction of experimentation in a cloud service platform. It describes relevant decision points taken (such as the target of the experiment and how to update the experiment design), benefits from the experiment (new insights, reduced development effort, and so forth), and challenges (access to end-users, inexperience with experimentation, length of the process, and so forth). Rissanen and Münch[9] investigate challenges, benefits, and organizational aspects when introducing CE in the B2B domain. They identified that customers play a major role when designing and deploying an experiment.

In the context of mission-critical systems, continuous experimentation has started to gain visibility in the automotive domain. Giaimo and Berger[19] investigate specific software architecture and design criteria to enable experimentation in automotive systems further. Mattos et al[20] discuss challenges and lessons from the automotive industry when starting to run the first A/B experiments. While some challenges are visible in other domains, such as the number of variants, suppliers, or the low number of users to run, others are specific to the automotive domain and do not generalize to other domains, such as restrictions imposed by the AUTOSAR architecture. However, experimentation in the automotive domain has only recently started. The current industry practices and the state-of-art research require more evidence to generalize to other companies or mission-critical systems.

## 2.2 | Experimentation processes

Several academic publications discuss the experimentation process used when introducing, scaling, and deploying experiments in online systems.[21-23] In this subsection, we provide an overview of these processes, and we compare them with the HURRIER process in Section 7.

### 2.2.1 | The build-measure-learn model

The Lean Startup methodology[15] proposes an approach for companies to continuously and systematically innovate from a startup perspective. The methodology employs a Build-Measure-Learn cycle to ensure that the software development is aligned with the customer's wishes. One of the critical aspects of this Build-Measure-Learn cycle is running scientific experiments to validate customer needs and ensure that the product is aligned with these needs. The build phase reinforces the use of a minimum viable product to steer the product roadmap's direction in a startup environment. The measure phase emphasizes instrumentation needs in the products to measure users' and systems' behavior. Finally, the learning phase uses collected post-deployment data to understand and learn movements in hypothesis metrics. This methodology describes a general experimentation process similar to experiments for learning and innovation.

### 2.2.2 | The ESSSDM model

The Early Stage Software Startup Development Model (ESSSDM)[24] proposes an operational support, based on lean principles, for practitioners to investigate multiple ideas in parallel and scale their decision-making process. The model consists of three steps. The first is the generation, in which the startup (or the existing company) generates ideas to expand their product portfolio. The second is the prioritization of the potential ideas in a backlog. The third is the systematic validation funnel using a Build-Measure-Learn loop similar to the Lean Startup methodology. In this step, multiple ideas can be investigated and validated in parallel. The funnel is divided into four stages: the validate problem, the validate solution, the validate the minimum viable product on a small scale, and the validate the minimum viable product on a large scale. In addition, this model supports the use of experiments for learning and innovation similarly to the Build-Measure-Learn model.

### 2.2.3 | The QCD model

The QCD model (Quantitative/qualitative Customer-driven Development)[25] explores the continuous validation of customer value instead of relying on up-front requirement specification. The QCD model treats requirements as hypotheses that need customer feedback for validation at the beginning of the development process. All hypotheses are gathered in a hypotheses backlog, where they are prioritized and selected for evaluation. In the validation cycle, the selected hypothesis is evaluated through both quantitative and qualitative feedback. If the hypothesis is not confirmed through the evaluation techniques, it can be refined in another hypothesis for a future iteration or abandoned. This model provides a higher-level experimentation process abstraction. It considers both qualitative and quantitative data analysis methods.

### 2.2.4 | HYPEX model

The HYPEX (Hypothesis Experiment Data-Driven Development) model[21] is a development for companies aiming to shorten the feedback loop to customers. Instead of spending engineering efforts on large pieces of nonvalidated functionality, the HYPEX model reinforces the need for an iterative and incremental approach. The model divides the development process into six steps: (1) Generation of a feature backlog. (2) Feature prioritization and gap specification. (3) Implementation of a minimum viable feature (MVF). (4) Analysis and comparison of the actual behavior with the expected one.(5) Generation of hypotheses to explain the actual behavior of the MVF. (6) Deciding if the feature should be abandoned, iterated once more, or if it should be considered completed.

### 2.2.5 | The RIGHT model

The RIGHT (Rapid Iterative value creation Gained through High-frequency Testing)[23,26] is a model for driving experiments in a startup environment. The RIGHT process model uses the Build-Measure-Learn loop to help a startup company to achieve the company's vision through continuous experiments. Hypotheses that implement business strategies are generated and prioritized, minimum viable features or products are implemented and instrumented, and data are collected. The analysis of the collected data helps the decision-making process in a similar manner to the HYPEX model,[21] where decisions to continue iterating, abandoning, or moving on to the next cycle are made. The RIGHT model describes a high-level experimentation process that can be used in innovation and learning experiments.

## 3 | RESEARCH METHOD

The purpose of this research is to gain an in-depth understanding of the use of continuous experimentation, including the objectives, the practices, the process, and the current challenges and opportunities when applied to a company that develops B2B mission-critical systems. Based on this purpose, we formulated the following research questions:

- **RQ1**: What are the types of experiments that are conducted in Ericsson and that are relevant in the development of B2B mission-critical systems?
- **RQ2**: What are the current continuous experimentation practices used in Ericsson in the development of B2B mission-critical systems?
- **RQ3**: Can the HURRIER process be used to drive CE in B2B mission-critical systems in Ericsson?
- **RQ4**: What are the current CE challenges and opportunities in B2B mission-critical systems observed in Ericsson?

### 3.1 | The case study

This study was founded on a qualitative case study design for two main reasons. First, it allows the researchers to study and understand the phenomenon in its context in more depth.[27] Second, since CE in B2B mission-critical systems, to the best of our knowledge, has not been discussed and investigated in research, a case study is an appropriate method for understanding a particular phenomenon in an industrial context.[28–31]

We followed the five steps for a software engineering case study using the guidelines proposed by Runeson and Höst:[31] (1) case study design: the objectives are defined, and the study is planned, (2) preparation for data collection: procedures and protocols for data collection are defined, (3) collecting evidence: execution with data collection on the study case, (4) analysis of the collected data and (5) reporting of the results.

### 3.1.1 | The case company

This research was conducted at Ericsson AB. Ericsson is a multinational networking and telecommunications company that develops, produces, and sells telecommunication equipment, services, software, and infrastructure to telecommunication operators in mobile and fixed broadband. Ericsson employs over 95,000 people in around 180 countries. Over the last ten years, Ericsson started the transition from traditional development to agile and towards DevOps. In the previous 3 years, CE began to get attention and promotion inside Ericsson. Although continuous experimentation is not a well-defined process throughout the company, several teams independently conduct over a thousand field experiments a year in different products and parts of the system. In Ericsson, experiments are used in many use cases ranging from innovation and new feature development to legacy assurance and performance optimization. Although this case study was conducted with a single company, we investigated CE practices in multiple teams, areas, and products spread over six locations in four countries.

### 3.1.2 | Data collection

The data collected consists of different data sources, including transcripts of semi-structured interviews, notes from meetings, emails, documentation, project plans, and presentations. The first author is conducting research on-site and working in close collaboration with Ericsson employees. The second author is employed by Ericsson. He was involved in several project and interview meetings and was also responsible for selecting the interview participants for this study.

We utilized a combination of criterion sampling with convenience sampling, where we interviewed the practitioners who were knowledgeable and accepted to participate in this study. We interviewed both participants who were part of the development and design of features validated using experimentation and participants who were not involved but had extensive experimentation experience inside Ericsson.

We conducted semi-structured individual and group interviews. The interviews were on-site and through phone conferences since the 25 practitioners were distributed in six locations in four countries. They had experience ranging from 3 to 25 years working in a range of different roles, as shown in Table 1. The interviews were conducted in English, were designed to last approximately one hour, and had an average duration of 55 minutes with a minimum duration of 41 min and a maximum duration of 1 h and 8 min. The interviews were conducted between December 2018 and May 2019.

Since part of the interviewees were actively engaged in the deployment of a mission-critical feature that was being deployed and validated with field experiments, we conducted additional four follow-up interviews with ten members, as discussed in Section 5.2. These interviews were used to evaluate the experience with the experimentation processes in this mission-critical feature. In Table 1, the follow-up interviews are identified in the second part of the table.

At least two authors were present in all interviews. In addition, interview guides were created depending and specific questions were asked based on the role of the interviewee. This allowed us to focus on their expertise and knowledge in the particular part of the experimentation process.

All interviewees were asked about their background and experience with customer experiments, live trials, data collection, and feedback from both customers and users. Relevant concepts to continuous experimentation already identified in literature such as gradual releases, dark deployments were also asked whether the subjects had previous experience with these concepts or related ones. We asked more in-depth questions about the process used, project timeline, impediments, lessons learned, perception of the benefits, and the disadvantages for all participants that have conducted entirely or partially projects with an experimentation component. For the interviews with the subjects involved in feature development, additional questions regarding the feature, the impact of the feature in the 4G product, the specific experimentation process used, results, lessons learned, among others. For interviewees in managing positions, we also asked their reflections on experimentation projects they supervised or followed and how the development organization moved towards experimentation. For participants in customer units, we asked questions regarding customer feedback and perception in experimentation projects, data collection and their current and future interest in collaborating in experimentation activities. The additional 10 interviews evaluated the use of experimentation in a mission-critical feature. Therefore, we asked specific questions about the project, the results, customer perception and feedback, lessons learned, and impediments created by using a CE process and its application in a mission-critical feature.

**TABLE 1** Overview of the interviews

| Interview | N. of Interviewees | Role | Location | Years of exp. |
|---|---|---|---|---|
| A1 | 1 | Operational product owner | L1 | 3 |
| B | 1 | Test manager | L2 | 28 |
| C | 1 | Program manager | L2 | 23 |
| D | 1 | Technical specialist | L2 | 15 |
| E1 | 6 | Developers and testers | L1 | between 3 and 5 |
| F1 | 1 | Product guardian | L1 | 6 |
| G | 1 | Product introduction manager | L2 | 20 |
| H | 1 | Product introduction manager | L2 | 19 |
| I | 1 | Product manager | L6 | 12 |
| J | 1 | Principal project manager | L2 | 23 |
| K | 1 | Program manager for field analysis | L2 | 22 |
| L | 1 | Customer solutions manager | L3 | 7.5 |
| M | 2 | Operational Product Owner, Developer | L4 | 8 and 12 |
| N | 1 | Customer support manager | L2 | 22 |
| O | 1 | Release manager | L2 | 24 |
| P1 | 4 | Technical coordinator, Field feature tester, Continuous deployment for customer support, Project manager support | L5 | between 3 and 15 |
| A2 | 1 | Operational product owner | L1 | 3 |
| E2 | 6 | Developers and testers | L2 | between 3 and 5 |
| F2 | 1 | Product guardian | L2 | 6 |
| P2 | 2 | Technical coordinator, Field feature tester | L5 | between 3 and 15 |

For all interviews, the academic purpose of the study and a statement about participants' anonymity in the analysis and results were explicitly shared before the start of the interview and agreed by the participants. All interviews were recorded and transcribed for the qualitative analysis, both the questions and participants' answers. Since all authors have nondisclosure agreements with Ericsson, the interviewees could utilize internal examples and freely discuss their experiences and practices.

Additionally, we collected data from over 30 documents, including project documentation, feature development plans, solutions, and product presentations for both internal employees and external customers. These additional documents were shared, mentioned, or discussed during the interviews, meetings, or emails and were available to the authors through the internal network. These documents contained detailed information about the development and release process of different features and products, the sequence of steps taken, customer feedback on both the continuous experimentation process and specific feedback used by the development team (e.g., feedback given in the customer feedback channel of the HURRIER process). We utilized these documents as a triangulation source to support the data collected from the interviews, particularly to help the ordering and timeline of activities to derive both the HURRIER process and answer the other research questions.

The described collected data was the main source of information to answer all research questions. The interview protocols, as well as how they connect to each research question, are available as supporting information at https://doi.org/10.5281/zenodo.4943011.

### 3.1.3 | Data analysis

The collected data were added to the qualitative analysis software NVivo, and thematic coding was used as the data analysis method. We utilized the six-phase process proposed by Braun and Clark.[32] We utilized inductive thematic coding, as the themes were first identified and linked to the data, rather than on previous subject coding frames.

The first phase consists of familiarizing with the data. The authors familiarized with the data in several ways, such as participating in the interview process, the transcription process, reading the transcriptions and interview notes. In the second phase, we generated the first set of codes, representing interesting concepts and ideas captured in the interviews or discussed in the additional documents. These codes identified in the interviews the goal of the experiment, perceived advantages and disadvantages, technical limitations, organizational limitations, deployment and experimentation techniques, customer perception, steps taken, deployment prerequisites, showstoppers, and so forth. In the third phase, we discussed potential themes for the identified codes. In the fourth phase, we reviewed the potential themes and merged similar codes when possible.

Then, we classified them as part of the theme-groups: experimentation objective, experimentation practices, experimentation activities, B2B challenges, mission-critical challenges, perceived advantages, customer involvement, among others.

In the fifth phase, we analyzed each theme individually, generating the main results and discussion points to answer each research question. Additionally, in the fifth step, we selected, summarized, and removed company-specific information for the examples presented in this paper. The last phase consists of this publication, where the results are presented and the research questions explicitly answered in the discussion. Figure 1 provides an illustrative view of how different sources of the collected data was combined in themes.

## 3.2 | Identification of the HURRIER process

The HURRIER process was identified by observing the CE practices of different projects, teams, and products. Utilizing the thematic coding approach, we were able to identify the key activities performed by each team, the order of these activities, techniques used, pre-conditions, challenges and impediments, results, and lessons learned. We ordered all these activities in time and terms of importance. From these, we iteratively derive an experimentation process that addresses the deployment of experiments for all of the analyzed teams and features involved in this research. All were developed in the context of B2B. While some of them are not mission-critical, over half are considered mission-critical. In Section 5.2, we illustrate the HURRIER process in one mission-critical project. The HURRIER process itself did not trigger the CE process used in this example, but it is our conceptualization and generalization of the observed CE experimentation process.

## 3.3 | Validity considerations

Below we provide an overview of the main validity threats we identified and discussed the different strategies used to minimize these threats. We also acknowledge the limitations of this study.

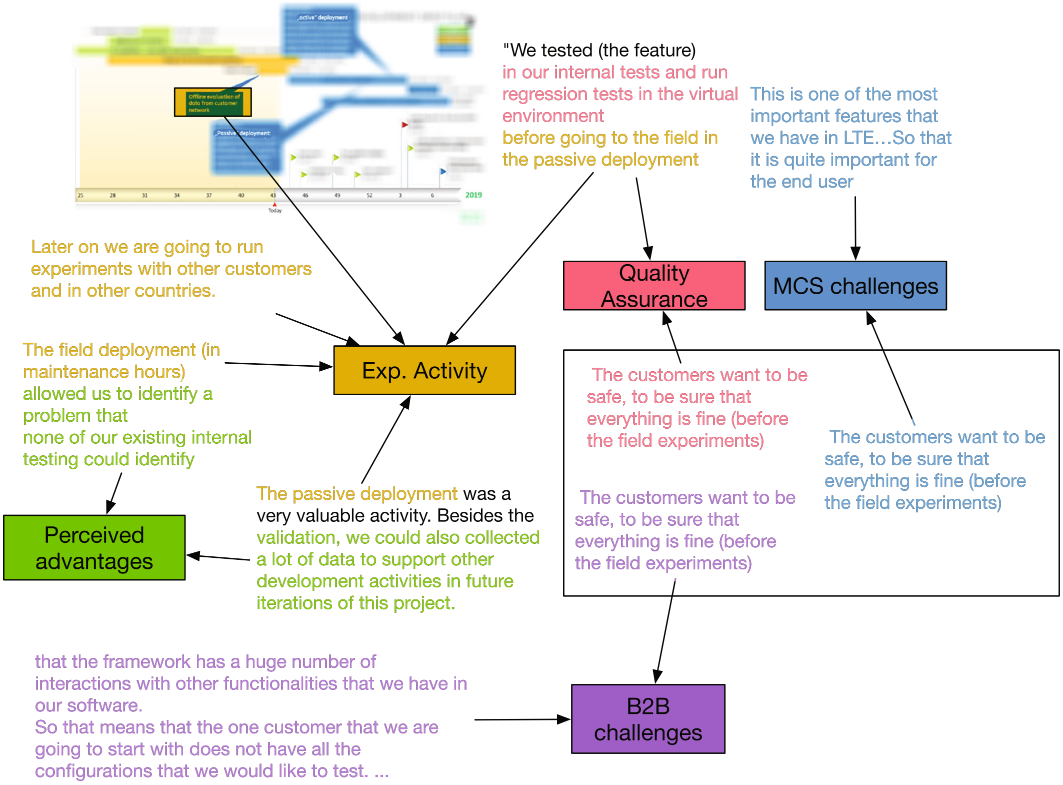
"active" deployment
Offline evaluation of data from customer network
Passive deployment
2019
Today
"We tested (the feature) in our internal tests and run regression tests in the virtual environment before going to the field in the passive deployment
This is one of the most important features that we have in LTE…So that it is quite important for the end user
Later on we are going to run experiments with other customers and in other countries.
Quality Assurance
MCS challenges
The field deployment (in maintenance hours) allowed us to identify a problem that none of our existing internal testing could identify
Exp. Activity
The customers want to be safe, to be sure that everything is fine (before the field experiments)
The customers want to be safe, to be sure that everything is fine (before the field experiments)
The customers want to be safe, to be sure that everything is fine (before the field experiments)
Perceived advantages
The passive deployment was a very valuable activity. Besides the validation, we could also collected a lot of data to support other development activities in future iterations of this project.
that the framework has a huge number of interactions with other functionalities that we have in our software.
So that means that the one customer that we are going to start with does not have all the configurations that we would like to test. …
B2B challenges

**FIGURE 1** An example of the coding process where both interviews and additional data were added in the themes. In colors, we represent the different themes that emerged from the data. The text represents different quotes and is color-coded to map how they link to each theme. Some quotes map into more than one theme. The project diagram refers to the plan of feature development discussed in example B in Section 5.2. The project diagram indicates the use and order of passive deployment before the active deployment (experimentation activity theme). Parts of the project diagram were blurred to hide specific details of the project (at the company request)

External validity: This case study was conducted within multiple teams, areas, and products in a single software company. To minimize the bias of working with a single team inside the company, our identified results were based on several teams' experimentation and practices in four countries. The different identified experimentation types and practices are used in specific parts of the organization, depending on the focus of the different teams. Some parts of the organization conduct only a specific type of experiment related to their specific task, such as a tuning experiment, and are not involved in customer support experiments. The practices and types of experiments are general enough to be valid in other domains since they are based on general scientific tasks used in other areas of science. However, other companies might not identify these practices since they may require specific conditions and maturity to emerge.

The presented HURRIER process is used entirely or a subset of it by the different teams and parts of the organization. We abstracted the activities of the HURRIER process performed by Ericsson in terms of common development activities used in different industries and research, such as continuous integration, passive launch, simulations, laboratory evaluations, gradual rollouts, etc. Although identified in Ericsson, the HURRIER process does not restrict specifically to Ericsson or the telecommunication industry, and therefore can be instantiated, used, or adapted by other software companies striving to introduce CE in mission-critical in the B2B domain. However, due to the nature of this study within a single company, we do not claim generalization of the process to the entire software industry.

Construct validity: The authors of this paper are well familiar with continuous experimentation practices and related research. However, the participants not always utilized the same nomenclature as research in continuous experimentation. To mitigate the threat to construct validity, the second author, who works as a full-time employee at Ericsson, was present in all interviews. When the practitioners asked for clarification or misunderstood a question, the second author explained or rephrased questions and concepts in the technical vocabulary used internally and given examples of well-known internal practices at Ericsson that exemplified the concept or question.

Internal validity refers to whether the unaccounted factors could impact the results of the investigated factors when causal relations are examined.[31,33] Given the choice of the research method, the data collection, and the analysis, it is impossible to separate the observed results from a complex context where this study takes place or establish causal relationships. Although not causal, the inferences made in this research have been checked with *explanation building* and *addressing rival explanations* as tactics to increase internal validity in case study research.[27]

Conclusion validity refers to the particular reasons, methods, and procedures we use to conclude a possible covariation between variables.[34,35] This research does not analyze covariation between variables. The conclusions presented are based on the thematic coding analysis method presented earlier. We utilize the well-establish thematic coding procedures from Braun, and Clarke.[32] However, the themes and results obtained are not a property of the data, but they emerge from the links and understanding we make from them and cannot be separated from the researchers.[32,36]

## 4 | CONTINUOUS EXPERIMENTATION AND PRACTICES

The diversity of the data from the different teams, areas, and products that are actively conducting experimentation indicates many experimentation objectives and practices. Although these objectives and practices are not restricted to Ericsson, or the B2B mission-critical systems, they greatly impact how the experimentation process is planned and conducted.

Next, we present the types of experiments identified during the case study and used within Ericsson. Then, we present the different practices and techniques used in these experiments.

### 4.1 | Types of experiments

One of the central aspects differentiating how different teams and part of the organization plan and conduct experiments is the purpose of the experimentation activity. This study identified four main groups: business-driven, regression-driven, optimization/tuning, and customer support experiments. We provide here a definition and specify how those types are used in the process in Section 6.

#### 4.1.1 | Business-driven experiments

These experiments are used to validate and assess business hypotheses by quantifying the value a particular change in the system results in and how it impacts higher-level customer/stakeholders metrics and KPIs. This type of experiment has been subjected to extensive research.[16,23,37] In Section 5.1, example A analyzes a business-driven experiment in the context of a machine learning feature in the 4G product.

### 4.1.2 | Regression-driven experiments

This is a quality assurance technique where the experiment is designed to observe if the new variation or system change harms one or several quality factors and system properties. These experiments are used when laboratory and internal tests cannot assess the modification's impact because they cannot accurately replicate all customer's equipment configuration or deployment conditions, such as complex environments, traffic profiles, and usage behavior. In Section 5.2, example B analyzes a regression-driven experiment in the refactoring of a mission-critical feature in the 4G product.

### 4.1.3 | Optimization and tuning experiments

These are experiments done in post-deployment stages, where a system's constants, configuration or calibration parameters are modified to tune the system for a particular operating condition. This type of experiment does not require a new deployment if the necessary metrics and the configuration system are already in place. In this type of experiment, both system internal and business metrics can be used for tuning purposes. In the telecommunication domain, LTE optimization with tuning experiments is a common procedure conducted by both the supplier and the mobile operators.[38–40] In Section 5.3, the example C analyzes the use of A/B testing to optimize a feature performance in the 4G product.

### 4.1.4 | Customer support experiments

These experiments are usually after a negative impact has been identified. Since the negative impact cannot always be traced back to a particular change, depending on the deployment conditions, usage behavior, configuration parameters, and other factors, experiments are needed to identify the root cause of the failure or negative change. Since errors in such complex situations can be stochastic, the experiments, in this case, consist of running controlled experiments with the current release of the software and one or multiple previous releases to identify where the error was introduced and which conditions trigger this error. In Section 5.4, example D describes the usage of customer support experiments to identify faults in stochastic scenarios in collaboration with customers in the 4G product.

## 4.2 | Experimentation practices and techniques

In this subsection, we present the different continuous experimentation practices and techniques identified in the collected data. When available, we contextualize these practices in relevant research in continuous experimentation and present other techniques not discussed before. We categorize the different practices and techniques into four groups: experimental design and analysis, variation assignment, implementation, and release techniques.

A critical difference regarding web-facing systems and business-to-consumers is the presence of two experiment levels. The first level aims to measure and evaluate the impact on the final users metrics (such as mobile phones). The second level aims to evaluate the impact of changes in the network level, such as radio base station metrics. The experiment's level impacts the choice of experiment design, the variation assignment, the implementation, and the release technique.

### 4.2.1 | Experiment design and analysis

An activity that impacts the planning and process for continuous experimentation is choosing the experimental design and the analysis method. At Ericsson, we identified the four experiment design types presented below.

*Randomized experiments*

A randomized (and possibly controlled) experiment is a type of scientific method used to investigate cause-effect relationships.[1,41] The randomized experiment is the most common CE practice and has wide adoption in web-facing systems. It consists of randomly assigning participants to different experimental groups. These groups differentiate by the treatment they are exposed to, while one group is held as a control group. In CE, this group of techniques encompasses A/B testing and multivariate testing. According to multiple interviews and as discussed in Section 5, Ericsson uses controlled experiments in regression-driven, business-driven, and customer support experiments when observing the impact of changes in mobile equipment metrics (first level experiment).

### Crossover experiments

Crossover experimental design is a particular type of design that the same subjects receive a series of treatments over time.[42] In this design, each subject serves as its control since we measure and evaluate within-subjects variance. One of the challenges of crossover designs is when the presence of carryover or learning effects is identified. If not controlled for that, the variance can significantly increase (or decrease), invalidating the design because the subjects change as they are exposed to different treatments. However, if the presence of carryover is known, different groups with different treatment sequences can be created to estimate the carryover effects.[42] One of the advantages of crossover designs is when a limited number of subjects do not present carryover effects. According to empirical data (both interviews and internal documentation), Ericsson utilizes crossover designs when there are limited systems for an experiment. For example, in the second level, experiments evaluate the impact of changes in the network level. Crossover experiments are used in all four types of experiments.

### Multi-armed bandits experiments

Multi-armed bandit experiments are a particular type of experiment design aiming to minimize cumulative regret by allocating fewer users to under-performing variants. As an example, if A is the current system and B is the system with a modification. Initially, both A and B are allocated with 50% of the users. If A is under-performing B, the design shifts the user allocation to B, with more than 50% of the users. This type of design aims to minimize the average number of users exposed to worse variations. Mattos et al[43] provide an overview and comparisons of multi-armed bandit designs and controlled experiments. At Ericsson, multi-armed bandit experiments are used in optimization and tuning experiments[40]

### Quasi-experiments

Quasi-experiments are a specific type of experiment that supports causal and counterfactual inference similarly to randomized control experiments, with the key characteristic that it lacks random assignment.[35,44] The assignment of variations to the subjects occurs by using cut-off criteria to divide the groups. The criteria can be based on natural conditions such as demographic data or other criteria or artificial conditions such as clustering methods based on different characteristics. Since quasi-experiments do not use randomization to minimize selection bias, this can decrease internal validity since additional confounding factors can be introduced during the assignment. However, well-planned transparent designs can minimize internal validity threats. One of the key motivating factors to use quasi-experimental designs compared to randomized designs is when randomization is impractical or unethical. From the interviews, Ericsson relies on quasi-experimental designs to investigate deployments in mobile networks, when the operators are responsible for selecting which parts of the network will receive an update first, or when geographical constraints do not permit complete randomization.

### Optimization

Experimental optimization is a general group of CE approaches used to optimize software based on a subject behavior (system or user). The approaches are generally used in optimization and tuning experiments to optimize system constants or calibration parameters. We group these approaches as a single practice because of their common experiment type. However, these techniques are based on very different premises and theories, ranging from Bayesian analysis,[45] response surface methodology,[41] Taguchi optimal designs[1,46,47] to search-based heuristics.[48,49] These techniques are commonly used in network optimization and can be performed by both the mobile operators as well as Ericsson.[40]

## 4.2.2 | Variation assignment

The choice of experimental design influences how each variation will be assigned to the users or customers. However, even in a specific design, assignment considerations should be made. The choice of the experimental design described earlier is interconnected with how the variants are assigned in the experiment. Variant assignments also have a large influence on the implementation and release techniques and the data analysis, the causal inference, and the validity of the experiment.

The variation assignment is often not a CE choice but rather a restriction imposed by the combined effect of the type of system, restrictions on how the data can be collected, how and to whom new variations can be deployed. We identified three main assignment choices: complete randomize, cluster-based and manual assignment.

### Complete randomization

Complete randomization is the variation assignment practice used in controlled experiments and crossover designs. It consists of assigning the system with or without the modification randomly to all units or users. This practice allows the differences between the units to average out as they are randomly sorted. Therefore, the observed changes are due to the modifications on the system and not by individual differences between the units. Although this practice has a higher internal validity, it is not always possible to conduct. Complete randomization is common in level 1 experiments (with the final users) inside a single or a similar group of radio base stations. From documentation and interviews, such functionality is implemented in specific features of the product.

*Cluster-based randomization*

Cluster-based randomization is a type of variation assignment technique that divides the population into clusters that are randomized together.[50–52] This kind of randomization is common in natural restrictions such as geographical location, as discussed in the interviews.

*Automatic assignment*

Automatic assignment is a type of variation assignment that is based on a criterion that is not necessarily randomization. For instance, multi-armed bandit systems can use different metrics to perform assignments, such as the upper confidence bound.[43] Also, optimization experiments can use a different range of variation assignment heuristics such as the expected improvement[45] or nature-inspired exploration techniques.[49]

*Manual assignment*

In B2B, the R&D organization might not have control over the assignment process. In such cases, they rely on the customer to assign how each variation will be assigned to users. In this manual process, the customer utilizes existing metrics to create two comparable groups. This manual process might also be aided by matching tools such as propensity score matching.[53]

### 4.2.3 | Implementation techniques

As part of the design of the experiment, the R&D organization, in collaboration with the customers/users, might decide on different implementation techniques to deliver the software change. We refer to implementation techniques as to how the modification in the software are implemented to be activated to the customers/users.

*Feature toggles*

Feature toggles are conditional statement blocks in the code that allows enabling and disabling features based on configuration parameters.[8,54] In the telecommunication domain, feature toggles are extensively used by customers and by the R&D organization to configure and customize specific parts of their network. Besides the configuration flexibility, feature toggles facilitate both the R&D organization and the customers to conduct and launch experiments without needing a new software build or deployment.

*Software versions*

The simplest form of deployment of a software modification for experiments is to generate different software versions and deploying them manually. Although such an approach might not be feasible for large-scale experiments, in the second level of experiments (the network level), the sample size is small, and this technique is viable. Additionally, customer restrictions in the deployment strategy can restrict other techniques such as feature toggles. As observed in the interviews, this strategy is more common in the first iterations of prototypes with customers.

*Traffic routing*

The technique consists of redirecting requests randomly between different instances of the system (one with the modification and one without).[1,8,16] This technique is commonly used in controlled experiment designs, websites, and back-end systems where concurrent experiments are low. However, it can provide low experimentation scalability or degradation on the user experience.[1] In the context of 4G product, this technique is not used or recognized among the interviewees, as redirecting traffic to different radio base stations is not practical and can severely impact the whole network. However, this technique can be used in other web-facing products developed by Ericsson.

### 4.2.4 | Release techniques

After having the software ready for an experiment, the R&D organization decides how this software will be released in the field with the customer. These release techniques are often risk minimization techniques aimed at preventing negative effects that have a large impact. If a negative impact is not observed, the scope of the release is increased.

*Canary release*

Canary releases are a practice of deploying a release to only a small percentage of the active customers and monitoring the behavior of the system for negative impact.[16,55,56] Since large negative effect sizes can be observed in smaller samples, this technique effectively minimizes the exposure of a negative change to the general population.[41]

*Passive deployment or dark launch*

Dark launch, also known as shadow, dark or passive deployment, is a technique where the new piece of functionality is deployed invisible to the customer.[16,55,56] In dark launches, the silent modification is exposed to real-world data but without acting on the system. The modification can be analyzed to see how it performs in a production environment and compares it to the existing solutions. While not a standard practice, this solution has been explored at Ericsson, as seen in Section 5.2.

*Gradual rollouts*

Gradual rollouts consist of gradually increasing the number of customers exposed to the new release.[57] This technique is commonly used together with canary releases. When no negative impact is observed, the sample of customers exposed to the new release is gradually increased.[16,56,58] This technique is also known as ramp-up and is commonly used in combination with A/B testing. Gradual rollouts are commonly used by internal development and customers in project plans and documentation.

*Ring-based releases*

Ring-based release is a common release technique in experimentation, where the software change is deployed to customers that have previously agreed with the release conditions.[59] The inside rings are exposed to a faster deployment cycle with new features and bug corrections at the expense of less stable builds. The problems identified earlier in the inside rings are corrected, and slower and more stable builds are released to customers in the outside rings, such as the General Availability (GA) features to all customers. Although not referred to by this name, a ring-based release is often mentioned in documentation and meeting notes.

*Time-window releases*

Time-window release is a technique where the system modification is released only when the application has a lower risk of negatively impacting users. Usually, the period corresponds to the lowest traffic/usage period or in maintenance hours. Similar to canary releases, this technique is also based on the fact that large negative effect sizes can be observed in smaller samples. As the development organization gains more confidence in the system's performance with live but low-risk data, the application can gradually roll out to larger time windows. This technique is often combined with other techniques, such as canary releases or gradual rollouts in the whole period. While not a standard practice, this solution has been explored by both customers and at Ericsson, as seen in Section 5.2.

Table 2 summarize the types of experiment and practices identified in the empirical data.

## 5 | EXAMPLES

This section presents four examples that investigate or utilize the discussed experimentation practices in the context of the Long Term Evolution (4G) system. The identified experimentation activities and techniques presented in Section 4, the presented examples lay the foundations of the inductively derived HURRIER process described in Section 6.

### 5.1 | Example A: Business-driven experiments

This example investigates the development of a new machine-learning feature aimed at replacing an existing solution. The new feature utilizes a machine-learning algorithm to provide faster and better hand-over decisions for the user equipment. Although the team also performs regression-driven and tuning experiments, we focus here on the business-driven experiments as captured in the quote below:

TABLE 2 Overview of the types of experiment, practices, and techniques

| Type of experiment | Experimental design | Variation assignment | Implementation | Release |
|---|---|---|---|---|
| Business-driven | Randomized | Complete randomization | Feature toggles | Canary release |
| Regression-driven | Crossover | Cluster-based | Software version | Passive deployment |
| Optimization and tuning | Multi-armed bandit | Automatic assignment | Traffic Routing | Gradual rollouts |
| Customer support | Quasi-experiments | Manual assignment | | Ring-based |
| | | | | Time-window |

> “We cannot prove the feature benefit for the customer in the lab, as it is very hard to simulate the customer network and traffic patterns … Also, customers value different aspects of how the feature impacts their network. So this type of (business-driven) experiment is the most valuable for us.” - Interview M

This feature came from developments in research and was selected by the team to proceed to the pre-study phase. The pre-study phase consisted of simulations to support (and replicate) the research results and determine the instrumentation and data collection.

> “The idea came from research. We had some simulations but nothing concrete to become a product. The question was not only if the idea would work in a live network but also if it is possible to do it in an embedded system. How much data would we need? Would the (ML) models fit the hardware constraints? Some of these questions we could answer before doing a prototype.” - Interview M

Initially, the prototype supported only the data collection in collaboration with a customer. These data supported the iterative development of the machine learning models and the introduction of the software to the embedded system. After several iterations in the prototype led to a first minimal viable feature that could be deployed in a live network. The first deployment of the feature was done with passive deployment, where both the customer and the development team could verify the behavior of the feature. Given the positive impact observed in the passive deployment, the prototype was tested in a limited number of radio base stations. The evaluation of this deployment was conducted in two steps, first with a quasi-experiment with a manual assignment and second with a crossover experiment. For the quasi-experiments, the customer manually assigned two comparable zones for the deployment. In one zone, the feature was activated, and in the other, it was in a passive deployment (the feature was available together with the instrumentation but not impacting the network). For the crossover experiment, the active zone was compared against its own metrics baseline, while the passive zone was monitored to see if there were changes in the metrics baseline.

> “This feature started as a prototype. We had the idea of what we wanted to do. After a few experiment iterations, we collected some field data and analyzed it. Together with good customer feedback, we got strong support for it to become a product.” - Interview M

With the expansion of the prototype to a product, the R&D team is able to conduct multiple customer validations and optimization experiments to improve the feature and the machine learning models.

## 5.2 | Example B: Regression-driven experiments

This example investigates the usage of CE in a project that consists of the refactoring of a framework that implements several functionalities of the 3GPP specification.[13] This framework is used to increase the speed, coverage, and capacity of mobile communication systems. It is considered a mission-critical system in the LTE context, that is, without the proper function of this framework, critical functionalities of the 4G are compromised with possible mobile traffic disruptions for the affected region. The refactoring procedure aims to increase performance, scalability of the system for new solutions and specification modifications, support for a number of new future user equipment, and open the space for new machine-learning and artificial intelligence solutions. The importance and critical aspect of the framework are captured in the quote:

> “This is one of the most important features that we have in LTE … as it has a great impact for the end-user” - Interview F1

The framework is highly complex as it interacts with over 20 different functionalities in the LTE system. It needs to interact and perform well with over the 5000 different configurations of user equipment available in the 3GPP specification and the additional new 5G systems. Combined with the different traffic profiles and optimizations that mobile operators can have, verifying and validating this system in all different conditions in-house is unfeasible. In terms of cost to create such a testing facility, the evolution of the testing facility to include the continuously increasing number of user equipment and the time to validate the solution. Internal testing of the framework can verify functionality interaction of the framework with new features and how new features can impact the framework. However, it is not possible to achieve high coverage and quantify the improvement of the solution without running field experiments in a customer network. This project involved design teams, development units, and customer units in 4 different countries.

> “We want to test it in the field with the customers … the main reason is that the feature that we are working on is highly dependable on the configuration that is used in the field, and the configuration is different in different countries. It is different between operators in the same country, and it is different between different types of user equipment that we have on the market. And due

> to that, it is very hard and probably impossible to verify all the functionality with internal testing, or in our lab. In our lab, we only have a limited set of user equipment, a limited set of configurations. So to secure the quality of the product that we are delivering, we want to have the ability to deploy that in some of the customers' networks, before we go full scale." - Interview F1

During the pre-study phase, one mobile operator had a network profile that could be used to validate a large part of the refactored framework and had a high interest in the evolution of the system after the framework has been deployed. The operator provided initial field data to aid the initial stages of the development. Following an incremental approach, the first version of the new framework was developed. In parallel, the R&D organization secured that internal verification is set up to cover major use cases and apparent configurations. The internal feedback channel was implemented with the Ericsson procedures for quality assurance, including CI reports, simulation status, and laboratory validation tests.

> "We tested (the feature) in our internal tests and run regression tests in the virtual environment. The field is a second step for validating. The customers want to be safe, to be sure that everything is fine (before the field experiments)." - Interview E

The customer feedback was implemented together with the local customer unit. It contains both automated data collection of the instrumented software in addition to an ad hoc manual collection of further performance and diagnostic data, if necessary. Because of the critical aspects of the framework, the single customer validation followed all activities. After passing the customer laboratory evaluation, the passive launch activity allowed benchmarking the responses from the new framework with the existing one. This activity allowed the R&D organization to identify corner cases from the live network traffic that were not covered during the internal validation and customer laboratory. The frequent feedback and iterations allowed the software to have enough confidence for a restricted launch in the live network.

> "The passive deployment was a very valuable activity. Besides the validation, we could also collect a lot of data to support other development activities in future iterations of this project." - Interview F2

The restricted launch enabled the framework only during maintenance hours for a week. Maintenance hours correspond to lower traffic and requirements on the network, making it a lower risk scenario in cases of faults. The analysis of the restricted launch was made by comparing key metrics and time series of the maintenance hours of the experiment week against a control week with the old framework. When the new framework reached enough quality level in key metrics, it proceeded to the gradual rollout.

> "The field deployment (in maintenance hours) allowed us to identify a problem that none of our existing internal testings could identify" - Interview F2

The gradual rollout followed the existing customer deployment plans for new software, where the software is deployed in groups, and the performance of each group regarding KPIs is measured before the deployment of the next group occurs. In this stage, quantitative feedback regarding the KPIs ensured that the new framework behaved as designed.

> "Later on we are going to run experiments with other customers and in other countries. The reason for that is that the framework has a huge number of interactions with other functionalities that we have in our software. So that means that the one customer that we are going to start with does not have all the configurations that we would like to test. … If we get positive results from this first customer, we want to expand that to the other customers" - Interview A

In parallel with the first (single) customer validation, the R&D organization contacted other customers for further field experiments. The contacted customers that also showed great interest in the framework had different network profiles to increase the coverage of the solution. Since the first customer validation already validated the most critical aspects of the framework, the R&D organization performed the gradual rollout (after the customer verified the software in their laboratories). Therefore, these new field experiments with multiple customers aimed to increase the coverage and confidence in the framework in special and corner cases.

The field experiments with multiple customers generated enough data and evidence for the R&D organization to proceed towards final documentation and release the framework to other customers in GA.

## 5.3 | Example C: Optimization and tuning experiments

This example discusses a feature developed in the 4G system to improve the configuration parameters of existing features utilizing experiments. The feature consists of an A/B testing framework that performs randomization in the first level (user equipment). The feature can randomize the users into multiple groups, where each group receives a different set of parameters. The feature utilizes existing performance metrics and relies on the configuration parameters already implemented in the system.

> “You can see exactly how each set of parameters impact the system at the same time, so you don't have the disadvantage of running the experiment for one week, and a holiday changes the traffic profile and the datasets are no longer comparable” - Interview D

The parameters to be investigated come from team-level performance goals, where it is hypothesized the expected impact and the proposed modifications. If the feature is already deployed in the customer network, it does not require a new deployment. However, the customer manually selects the network cluster where this feature will activate (second level). Suppose a particular set of parameters outperform in this limited release. In that case, it can be gradually rolled out to other parts of the network, where the aggregated effects can be investigated in a crossover experiment.

However, optimization experiments are not exclusive to feature configurations but are also possible at the network level (second level) and even without the participation of the R&D organization.

> “The customers tune their network over many years. They have a lot of knowledge over the dependencies between the features and configurations in their network. However, new customers or customers that do not have the same maturity in tuning the network will benefit from tuning expertise from our R&D teams.” – Interview D

### 5.4 | Example D: Customer support experiments

Customer support units are often responsible for managing the feedback between the mobile operator and the development unit. After the general availability of the software, customers can still find unexpected behavior or metric degradation. If the source of degradation can be traced, software patches are created. The software patch delivery is often done with regression-driven experiments using a crossover design experiment by the customer support units.

> “We deliver corrections to customers on issues that they have but since we are not able to fully verify the patch without the specific configurations, profile of the network and customer KPIs, we do regression experiments” - Interview N

However, customer reports of degradation and faults might not be able to specify the part of the system the failure was observed or even the version of the software where the fault was introduced due to changes and the stochastic behavior in the operational environment. In such cases, the customer support unit conducts “customer support experiments” to identify the fault source so the development unit can provide the appropriate software patch. Customer support experiments are used to diagnose faults in the software and configuration issues in the network.

> “In troubleshooting, not everything we get is a software issue. Sometimes it is a configuration fault. And we need to do changes and run experiments on the live network to identify the source (of the fault/behavior)” - Interview N

The customer provides specific requests to the R&D organization. The customer unit acts as the direct customer feedback channel. The request is studied to see if they are able to identify the source of the fault. If the fault can be traced, the development unit provides software corrections experimented in a regression experiment. Otherwise, the customer unit conducts direct experiments with the customer. Customer experiments are usually quasi-experiments, with a manual assignment and with software versions or feature toggles. The use of software versions is to verify which version the fault was introduced, while feature toggles help identify the features.

When the source of the fault is identified and the development unit provides a patch, regression experiments are conducted to verify the behavior of the patch before full deployment to the whole network and other customers.

## 6 | THE HURRIER CONTINUOUS EXPERIMENTATION PROCESS

The deployment of new software in B2B mission-critical systems, unlike many web-facing applications, requires extensive verification, risk analysis, and customer approvals. The R&D organization must comply with specifications, pre-established testing procedures, and service level agreements. In these situations, field experiments must be planned and agreed upon in collaboration with the customers to ensure that the R&D organization receives adequate field feedback and minimize the risk imposed on the service provided by the customer. In this section, and based on the empirical data, we present the HURRIER process for conducting experimentation in mission-critical features in the B2B domain. The HURRIER process was identified and formulated based on the current experimentation process and practices of different teams inside Ericsson, as discussed in sections 4 and 5. The process is used in its entirety or just a subset of its activities, depending on the scope and area of the development project.

The process is composed of a set of generic activities that can be organized in four main areas around two feedback channels. The areas are: (1) the R&D organization, (2) the internal validation, (3) single customer validation, and (4) multiple customer validation. Next, we discuss each of these groups in detail, the set of commonly found activities, and the feedback channel. Figure 2 shows an overview of the HURRIER process. In this process, the square boxes represent activities. The thin arrows indicate the sequence of the activities inside an area. The thick arrows represent feedback data. At any point in this process and based on the feedback data, an activity can be interrupted and returned to the R&D organization, either in the form of new requests and ideas, by adding new use-cases and providing additional data for the pre-study activity, or providing continuous feedback for the incremental development activity.

## 6.1 | The R&D organization

The R&D organization is responsible for developing the feature or change that will be deployed. The R&D starts after a development idea is generated. These development ideas can come from different sources such as direct customer request, market needs, competition, innovation, or to meet the internal goals of the R&D organization.

### 6.1.1 | Prestudy

The pre-study activity consists of scoping the project and planning its development. Metrics and success criteria are determined, and the expected improvement in internal system and customer level metrics is defined. The feature is divided into incremental steps that can be rapidly evaluated in the field in collaboration with the customer. In this activity, potential customers are selected to evaluate the first experiments. These customers

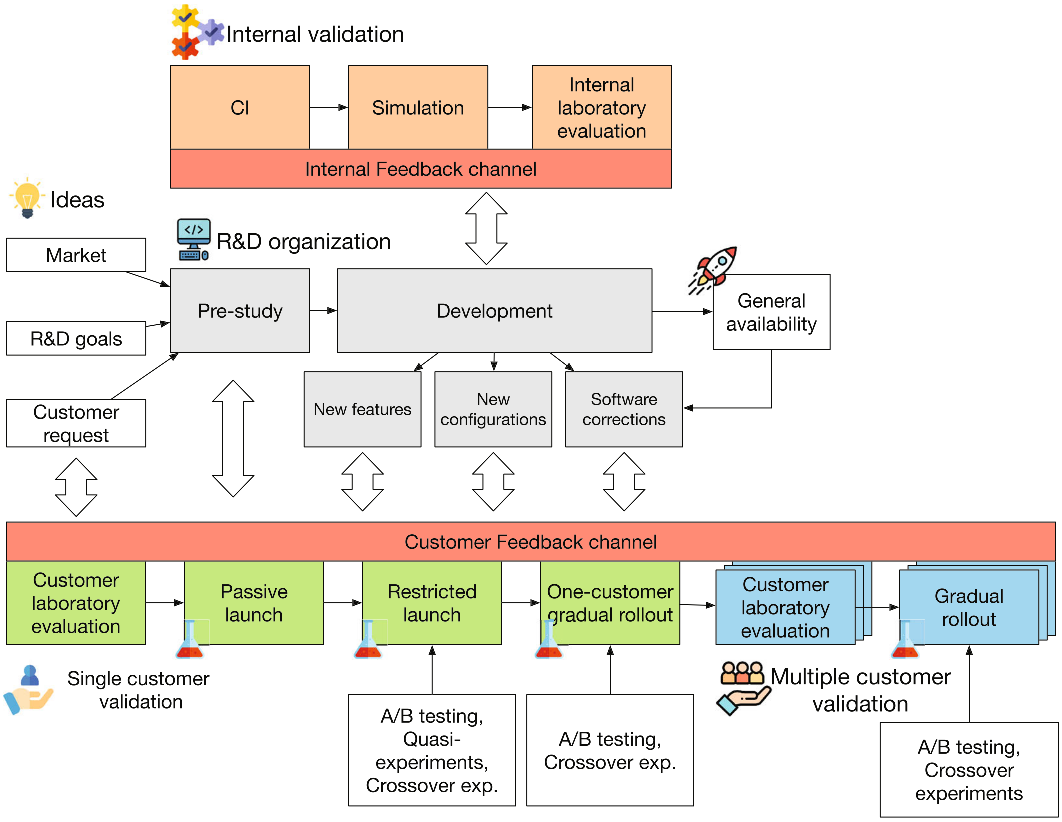
Internal validation
CI
Simulation
Internal laboratory evaluation
Internal Feedback channel
Ideas
Market
R&D goals
Customer request
R&D organization
Pre-study
Development
General availability
New features
New configurations
Software corrections
Customer Feedback channel
Customer laboratory evaluation
Passive launch
Restricted launch
One-customer gradual rollout
Customer laboratory evaluation
Gradual rollout
Single customer validation
Multiple customer validation
A/B testing, Quasi-experiments, Crossover exp.
A/B testing, Crossover exp.
A/B testing, Crossover experiments

FIGURE 2 The HURRIER Process. The different activities in the process can be organized into four main areas: (1) the R&D organization (in gray), (2) internal validation (in orange), (3) single customer validation (in green), and (4) multiple customer validation (in blue). The internal feedback channel provides continuous feedback from the quality assurance activities, while the customer feedback channel provides feedback from field experiments back to the R&D organization

usually have a significant interest in the specific feature, either because of their request or potential benefits. Customers in this step can have different degrees of interaction, from an observer role to a more active role in the design of field experiments.

> “Some customers understand the need for field experiments in their network. Because with experiments they can access earlier the latest functionalities and improvements of our software.” - Interview C

### 6.1.2 | Incremental development

After the pre-study, the development activities start. We divide the development activities into three main groups, as discussed next.

*New feature*

New features are intended to introduce new functionality in the system. After the prestudied, the feature is divided to be implemented incrementally and with constant feedback validation from both the internal procedures and feedback from the field in close collaboration with the customer.

New functionality is often designed as prototypes with a very limited scope to have a first field validation before going through new iteration cycles.

> “We always start with understanding the problem and trying to solve it in the simplest possible way, and then before we start an expensive simulation or a study how this interact with everything else to fully understand it, we try to build a prototype and test it (in the field), the minimum scope, with tremendous amount of limitations. Possibly, the prototype does not work with this or that feature, but at least we can test if there is a gain or not … We are not more efficient when it comes to building time, but we know in advance that it works (in the real-world). That is the benefit of prototyping and real world experimenting.” - Interview I

*New configurations*

New configurations consist of a larger category of changes. It includes changes in existing static parameters, configuration parameters, or proposed new parameters for the network. This activity usually does not require extensive internal laboratory evaluation if these configuration parameters are still in a predefined validated range. Some configuration proposals do not require a new software build or even a new deployment since they can be modified through existing interfaces in the software.

*Software corrections*

Software corrections consist of bug fixes and other modifications to address identified functionality faults or metric degradation. New test cases, simulation scenarios, and laboratory procedures are introduced if the fault is deterministic. However, as discussed in Section 5.4, faults can present stochastic behavior or be specific to a particular network configuration. In these cases, validation of the software correction is done in collaboration with the customer.

## 6.2 | The internal validation

The internal validation consists of quality assurance activities. These activities are performed both before and in parallel with the customer validation and the field experiments. Before the internal development reaches the customer for a field experiment, the R&D organization conducts a series of internal validation procedures to guarantee a minimum quality for first field deployment. The internal validation before the customer field experiments is not aimed at reaching a high degree of coverage as provided by a field evaluation. Instead, this internal validation captures integration problems, interaction with other features, and other common implementation errors. The first iterations of the internal validation are considered a fast procedure. It targets guaranteeing an acceptable level of quality while minimizing the leading time to deploy with the customer. In parallel to the customer validation activities, quality assurance teams incrementally validate the development.

### 6.2.1 | Continuous integration

Continuous integration (CI) is a mandatory internal validation activity. This activity aims to identify integration problems and interaction of the feature under development with various other features and many different hardware configurations. This activity is part of all deployment processes at Ericsson, regardless of the presence of field experiments or not. Continuous integration at Ericsson has been discussed extensively in Ståhl and Bosch.[60]

### 6.2.2 | Simulation

The development of some features can be verified using simulators. Ericsson has modeled several characteristics of its products and of the different environments that the products can be deployed. The development of a simulator is an intensive activity that requires extensive validation. Additionally, not all conditions can be easily simulated or have a simulator available, and for those conditions, an internal validation with simulation is not used.

### 6.2.3 | Internal laboratory evaluation

Similar to the simulation activity, Ericsson has several laboratory testing environments. The laboratory environments are designed to capture and verify a large number of use cases, including software and hardware integration. These cases give a good indication of how the deployed system will perform under controlled circumstances. However, the feature under development sometimes addresses corner cases or requires the complex interaction of live network traffic. Those cases are often hard and expensive to recreate in a laboratory environment, and in those cases, the feature requires field experiments for validation.

## 6.3 | Single customer validation

The internal validation provides software with enough quality and verification to be deployed in the field. However, as discussed previously, internal validation activities cannot cover all the quality aspects of the system. Time and cost constraints impose a limited number of scenarios that can be run in simulation and laboratory evaluation. Additionally, controlled environments lack the high complexity and variability seen in field deployments, such as in traffic patterns, types, and a number of user equipment. They cannot assess customer-specific KPIs (key performance indicators). The first field validation is done in collaboration with a single customer, and it is usually the same one that has been involved since the beginning in the pre-study phase. This customer is highly interested in the field experiment development and success and collaborates to share field data in qualitative and quantitative feedback.

### 6.3.1 | Customer laboratory evaluation

The first activity after the internal verification is the deployment of the software for evaluation in the customer laboratory. The customer laboratory is run by the customer and contains specific configurations to replicate its network. The customer can verify the software with its internal test procedures and Key Performance Indicators (KPIs) in the laboratory. This step gives confidence for the customer to deploy the software in its own network. This evaluation is also considered a fast procedure since it does not cover all cases.

### 6.3.2 | Passive launch

Passive launch, also known as a dark launch or a dark deployment,[16,61] is a CE technique that consists of deploying the new feature or change in parallel with the existing system. The new feature performs its task in the background, and it is executed by the same traffic profile and inputs of the system. However, its output and its main functionality are not exposed to the users. It is used to provide an open loop verification of how the feature would behave in production in systems where response parity is necessary, and the correctness of the response can be evaluated. This activity is not mandatory, and it is often seen as a time-consuming activity for smaller features and changes. However, in mission-critical development, a passive launch can increase the confidence level in the deployment, verify the system response, memory and CPU usage, and the quality of the response with minimal to no risks for the end-users.

### 6.3.3 | Restricted launch

Restricted launch corresponds to the system's deployment in a low-risk scenario that can help validate the development. The restricted scenario can be the selection of systems so that if they fail, then the impact is small on the final users (such as systems with high redundancy and safe fails). Additionally, the restricted scenario can be a restriction in time. The new feature is deployed only in low-risk periods, such as maintenance hours or low traffic hours. If the deployed systems in the restricted launch are compared to other equivalent systems, the customers and the R&D

organization usually follow a quasi-experimental design.[35] If the system metrics are compared with the historical metrics or metrics after the restricted launch, the R&D organization and customers plan for a cross-over experimental design.[62] If the tracked KPIs are end-user dependent, the R&D organization and the customers can utilize A/B testing or another randomized factorial design to evaluate the impact of the feature. In this last case, negative movements are further investigated, while statistically non-significant and positive significant movements give confidence for proceeding towards a larger experiment.

### 6.3.4 | One customer gradual rollout

After a successful restricted launch, the customer has enough confidence to make the gradual rollout of the feature to the whole network. This rollout can be randomized, from lower to higher risk systems, or use another pre-established procedure. Together with the manufacturer, the mobile operator can decide to run cross-over experiments or A/B experiments between each ramp-up or select two previously known regions with high correlation to evaluate the deployment. At the last stage of the gradual rollout, a full experiment is conducted to evaluate the deployment. This evaluation can be used to verify the value of the deployment and the quality aspects of the deployment. At this stage, the R&D organization can continually develop the feature towards its full scope, run additional single and multiple customer experiments, or abandon the idea and development and move to the next feature.

## 6.4 | Multiple customer validation

After one customer validation, the R&D organization aims at iterating on the feature to deliver values to multiple customers.

Suppose the field experiments with the first customer already provide enough coverage and confidence regarding quality or value in the solution. In that case, the R&D organization can decide to mark the feature for general availability (GA), which means that the feature has adequate quality and is, therefore, ready to be deployed by any customer. However, a single customer often cannot cover all the necessary validation aspects of the feature. The R&D organization may select a number of additional customers that can increase this coverage for more field experiments.

> “We have some experiments with teams where they have the customer involvement all the way (from early development). The problem with that is that we develop features that are supposed to be globally and possible to use for all our customers.” - Interview B

Since the feature has already gone through field validation and has higher confidence, some steps like the passive and restricted launch are usually not covered. The customer laboratory and the gradual rollout are similar to the single customer validation. However, the R&D organization may focus on identifying the corner cases to increase the feature coverage and the delivered value by choosing different experimental designs. Additionally, due to the significant differences between each customer network, these experiments are analyzed individually and not combined.

After the multiple customer validation, the feature is fully documented and marked for GA. At this stage, other customers can acquire or deploy the feature in their networks fully or partially, gradually or at once. However, the feedback to the R&D organization takes longer, as the development has moved towards a new idea and other experiments.

## 6.5 | The internal and the customer feedback channels

In the B2B context, the software manufacturer cannot always have direct access to field data or user data. The HURRIER process centers all activities around two main feedback channels to ensure the development team has access to all necessary field data.

The internal feedback channel consists of reports and communication between the development teams and the operation teams. This feedback channel provides quantitative data from the quality assurance teams, such as continuous integration status, simulation reports, and laboratory test reports.

The customer feedback channel is an agreed communication channel between customers and the R&D organization. It serves as a central source of feedback that developers can get from the field validation activities with customers. The provided feedback can be both quantitative and qualitative data. The customers can control what information they provide, facilitating compliance with specific regions' regulations, such as GDPR, guaranteeing the anonymity and privacy of their users, and controlling business-sensitive information. If a particular development activity requires sensitive data from users, the customers can agree to run analysis scripts on the data and only provide feedback. The customer feedback channel can be implemented in several ways, from automated data collection of instrumented software, direct contact between developers and customers, or specialized customer units.

## 7 | DISCUSSION

This section discusses the research questions and the challenges and lessons learned from retroactively applying the HURRIER process in the presented examples.

### 7.1 | RQ1:What are the types of experiments that are conducted in Ericsson and that are relevant in the development of B2B mission-critical systems?

In Section 4.1, we discussed four types of experiments that are commonly conducted internally at Ericsson. All four types of experiments are conducted in B2B mission-critical systems in close collaboration with the customers, as exemplified by the examples in Section 5.

Business-driven experiments have been extensively researched in the context of web-facing systems[1,4,5,17,23,63,64] and more recently in the context of embedded and automotive systems.[20,65,66] Business-driven experiments are often used interchangeably with A/B testing in the web domain and often, but not limited to, for changes in user interfaces.

The usage of regression-driven experiments has been discussed earlier by Schermann et al.[8,16] They discuss the usage of regression-driven experiments to detect functional problems that were not captured in internal testing, performance regression, or testing the scalability of a feature. In our work, we reinforce these aspects and present them as an integral part of the release process in the HURRIER framework.

Tuning and optimization experiments are widely used in web-facing systems. However, they are often categorized as business-driven experiments despite having critical differences in the planning and conducting process. This distinction becomes more evident when the customers have higher requirements in the product configuration for their use, such as in mobile networks. In the telecommunication domain, optimization experiments are widely discussed, and[38] methods to automated this process is a current subject of research.[39,40]

Customer support experiments have not been discussed in research literature before and present a new concept in experimentation. However, as companies start to introduce computational intelligence features based on machine learning, the interdependence between the system, the data, and the operational environment will increase and lead to more stochastic faults and degradation that requires customer support experiments to identify the source of the problem.

### 7.2 | RQ2: What are the current continuous experimentation practices used at Ericsson in the development of B2B mission-critical systems?

In Section 4.1, we discussed several experimentation practices and techniques identified in our data collection and contextualize them with existing research. These practices and techniques were classified into experiment design and analysis, variation assignment, implementation, and release techniques.

In the design of the experiment, research and web-facing companies have often focused on controlled experiments (A/B testing or multivariate testing) due to the control over the deployed software version and the higher number of users. However, as observed in the examples and the HURRIER process, crossover and quasi-experiments have a significant role in experimentation in B2B mission-critical systems at Ericsson. In the B2B domain, experiments need to be conducted with close collaboration with the customer. In these cases, the customer often decides on the variation assignment and the sample size. Additionally, features and metrics in the second experiment level (network) can also have geographical restrictions. These restrictions limit the experimentation design to crossover or quasi-experiments and the variation assignment to manual and cluster-based assignment.

Although the variation assignment and release techniques are similar to what is observed in other domains, there are some differences in B2B mission-critical systems. For example, the automation of the release and rollback of software is often undesired and restricted by the customer, that carefully monitors and controls each modification.

The taxonomy presented by Auer et al[18] describe characteristics of A/B experiments and A/B experimentation platforms focusing on a specific experiment iteration. In contrast, the activities and practices we discuss are presented at a higher level of abstraction. For example, after a team in the case company decides to run a business-driven controlled experiment with complete randomization, with feature toggles in a gradual rollout, it is still necessary to decide on the specific characteristics presented on the taxonomy (variant id, duration, guardrail and success metrics, and so forth). It is also worth noting that experimentation can be run and conducted ad-hoc, i.e., without an experimentation platform.[4] While Ericsson has a tailored platform and feature for conducting experiments in the first level (the user-level), ad-hoc experiments are suitable for the second level, which requires a joint experimental design with the customer since full randomization is often not possible. The release might involve different techniques depending on the maturity of the feature being experimented with.

## 7.3 | RQ3: Can the HURRIER process be used to drive CE in B2B mission-critical systems at Ericsson?

By observing the CE practices of different teams, we observed the key activities that compose the experimentation process at Ericsson. The empirical data, the set of practices, the concrete examples, documentation, and feature plans led to the development of the HURRIER process model. The HURRIER process represents a superset of the activities that are performed by these different teams. A subset of the activities in the HURRIER framework was used in each of the four types of experiments discussed in section 5. The requirements and the availability of tools (such as simulators and test rigs) in both the R&D organization and in the customers determine which set of activities are instantiated.

When deciding upon the activities and instantiating the process, the key aspects are the time-length of the activity and the value it delivers when verifying the system. Deployment activities, such as network optimization depend heavily upon the existing conditions of the operator's network. In those cases, extensive laboratory testing and simulation activities deliver little value compared to field experiments. The internal validation is kept to a minimum. The software is validated only in terms of quality, guaranteeing that it does not influence or deteriorate other features.

On the other hand, developing mission or even safety-critical features requires more extensive and lengthy internal validation, including following specifications and legislation. In this case, simulations and laboratory evaluations play a major role in increasing confidence before the field deployment. However, simulations and laboratory evaluations should be kept to the minimum necessary to guarantee the safety and basic functionality of the feature. Field experiments present a fast way to evaluate the feature and increase coverage in the operating conditions, which is often not feasible, costly, and time-consuming to implement in laboratory conditions. Deploying the mission-critical feature in the field within maintenance hours where traffic is minimal is another way to minimize the risk when the implementation is deployed in the field for the first time. Therefore, it is not only that the feature is deployed to a small subset of the network but also done at different times to reduce the risk further.

The feature should be implemented incrementally, so its value can be evaluated faster. Any evaluation of the delivered value should be left for the field experiments. Suppose the feature's minimum functionality does not deliver the expected value. In that case, the feature can be abandoned without the need to go through all the extensive work of developing, verifying, and even certifying the full feature.

The HURRIER process differs from the already existing experimentation process regarding the granularity level and type of activities. The HURRIER adds and reinforces specific activities for deploying software that needs to have high-quality assurance and in the B2B context. However, the proposed HURRIER process does not contradict existing experimentation processes since it is built on top of those models. For example, the execution step of an experiment iteration in the RIGHT model is broken down into several activities such as customer laboratory evaluation, passive launch, restricted launch, and one-customer gradual rollout. Although necessary for the context of mission-critical systems and B2B context, these activities might not be relevant for all startup experiments or web-facing companies that usually have lower risks involved in the deployment and are more in control of the data collection and deployment strategy than the customers. The HYPEX model reinforces the iterative process to increase the value of the delivered software. This is commonly seen in business-driven experiments. While the HURRIER process supports this, it also supports customer support experiments involving continuous iterations to add value.

The HURRIER process emphasizes that to continuously deliver high-quality and validated solutions, two aspects are required. The first is the presence of CI and CD. The second is the exposure of the system to live context and collaboration with customers to obtain feedback. These two aspects provide faster and more valuable feedback than focusing only on in-house validation and predefined testing scenarios.

## 7.4 | RQ4: What are the current CE challenges and opportunities in B2B mission-critical systems observed at Ericsson?

Running CE in the B2B domain and focused on mission-critical features presents many opportunities and challenges for both the company and the customers.

One of the challenges in CE at Ericsson is that any new deployment requires explicit approval and consent of the customers. For that, the customer needs to understand the need for the deployment, have a clear vision of how it can impact the system, and the potential benefits. However, successful experiments and transparency between the R&D company and the customer can help to build a trust relationship. A high trust relationship can facilitate running field experiments and evaluating new ideas for which the customer's direct benefit is not yet clear. CE in the B2B domain also requires close collaboration between companies and customers since the field experiments are run in the customer premises. The collaboration happens from the pre-study to the gradual rollout phase.

Depending on the level of interaction, trust between the R&D team and the customer, and the level of interest in the feature, customers can help shape the development activities and even steer the development of the feature. While this can bring benefits, it also creates some challenges when adapting the feature for general availability, as seen in interview B section 6. For instance, customers who have steered the feature since its conception might feel ownership and see some decrease in value when the feature is made to work in conditions beyond their network.

CE practices allow higher coverage and confidence in the development solution in mission-critical features, reducing the number of problem reports after general availability (GA) and minimizing the costs of developing and maintaining an extensive simulation and laboratory solution.

However, managing the risks of a field experiment in the B2B context relies on the customers, which requires the company to have a close collaboration and build a high-trust relationship.

CE is not seen only as positive for the case company but also the customers. For the company, CE helps to reduce the time-to-market as features are continuously validated with field data in terms of value and functionality. Other perceived advantages are: (1) the reduced time for a problem report to be addressed by the development team; (2) an increased sense of accomplishment, as the functionality is seen in the field in a shorter period; and (3) the higher sense of autonomy that comes with a higher trust from the customers, as the development teams can propose and test new ideas in the field faster. For the customer, CE has allowed them to test new functionalities, evaluate ideas and understand the impact of those ahead of the release, give them a competitive advantage on their corresponding markets.

## 8 | CONCLUSION

In collaboration with Ericsson, we conducted a case study research method to understand how CE is used in B2B mission-critical systems. The case study investigated the continuous experimentation practices of multiple teams in different locations and countries working on the 4G mission-critical product.

This paper provides four main contributions. First, we classify the different types of experiments, practices, and techniques used in B2B mission-critical systems. Then, these techniques are discussed in the context of existing research in other domains. We identified four types of experiments that are conducted: business-driven, regression-driven, optimization, and customer support experiments. Second, we present and analyze each of the four types of experiments with four examples. The examples show the general experimentation process followed by the team and the usage of the different practices and techniques. Third, based on the empirical data, we inductively derived the HURRIER process (**H**igh val**U**ed softwa**R**e th**R**ough cont**I**nuous Expe**R**imentation). This process combines different experimentation techniques and practices to deliver high-quality solutions that the customers value. At Ericsson, subsets of the HURRIER process helped the R&D organization to validate feature functionality, increase coverage, identify and trace stochastic faults and increase the confidence in the developed solutions much faster than without the field experiments. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

CE has the potential to deliver value and higher-quality solutions. However, in the B2B domain, this can only be achieved with high trust and close collaboration with the customers. Therefore, we plan to introduce the HURRIER process and evaluate it in different domains in future work.

## ACKNOWLEDGEMENTS

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by the Software Center. The authors would also like to express their gratitude for all the support provided by Ericsson.

## DATA AVAILABILITY STATEMENT

The interview data is not shared as it may contain sensitive company information. Research protocols are available as supplementary material at: https://doi.org/10.5281/zenodo.4943011

## ORCID

*David Issa Mattos* https://orcid.org/0000-0002-2501-9926
*Helena Holmström Olsson* https://orcid.org/0000-0002-7700-1816

## REFERENCES

1. Kohavi R, Longbotham R, Sommerfield D, Henne RM. Controlled experiments on the web: Survey and practical guide. *Data Mining Knowledge Discovery*. 2009;18(1):140-181.
2. Mattos DI, Bosch J, Olsson HH. Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives. In: Garbajosa J, Wang X, Aguiar A, eds. *Agile Processes in Software Engineering and Extreme Programming. XP 2018*. Lecture Notes in Business Information Processing, Vol. 314. Porto, Portugal: Springer, Cham; 2018:277-292. https://doi.org/10.1007/978-3-319-91602-6_20
3. Crook T, Frasca B, Kohavi R, Longbotham R. Seven pitfalls to avoid when running controlled experiments on the web. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining ACM*. Paris France; 2009:1105-1114.
4. Fabijan A, Dmitriev P, Olsson HH, Bosch J. The evolution of continuous experimentation in software product development: From data to a data-driven organization at scale. In: *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. Buenos Aires, Argentina: IEEE; 2017:770-780.
5. Auer F, Felderer M. Current state of research on continuous experimentation: a systematic mapping study. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Prague, Czech Republic: IEEE; 2018:335-344.
6. Fitzgerald B, Stol K-J. Continuous software engineering: A roadmap and agenda. *J Syst Softw*. 2017;123:176-189.
7. Lindgren E, Münch J. Raising the odds of success: The current state of experimentation in product development. *Inform Softw Technol*. 2016;77:80-91.

8. Schermann G, Cito J, Leitner P. Continuous experimentation: Challenges, implementation techniques, and current research. *IEEE Softw*. 2018;35(2): 26-31.
9. Rissanen O, Münch J. Continuous experimentation in the b2b domain: A case study. In: *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*. Florence, Italy: IEEE; 2015:12-18.
10. Yaman SG, Fagerholm F, Munezero M et al. Transitioning towards continuous experimentation in a large software product and service development organisation–a case study. In: *International Conference on Product-Focused Software Process Improvement Springer*. Trondheim, Norway; 2016:344-359.
11. Yaman SG, Munezero M, Münch J, et al. Introducing continuous experimentation in large software-intensive product and service organisations. *J Syst Softw*. 2017;133:195-211.
12. Fowler K. Mission-critical and safety-critical development. *IEEE Instrument Measurement Mag*. 2004;7(4):52-59.
13. ETSI. 3GPP Technical Specification Release 14 - ETSI TS 136 300. Release 14, Valbonne, France, ETSI; 2017.
14. Mattos DI, Dakkak A, Bosch J, Olsson HH. Experimentation for business-to-business mission-critical systems: A case study. In: *Proceedings of the International Conference on Software and System Processes*. Seoul, South Korea / Online; 2020:95-104.
15. Ries E. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. New York, US: Crown Business Publishing; 2011.
16. Schermann G, Cito J, Leitner P, Zdun U, Gall HC. We're doing it live: A multi-method empirical study on continuous experimentation. *Inform Softw Technol*. 2018;99:41-57.
17. Fabijan A, Dmitriev P, McFarland C, Vermeer L, Holmström Olsson H, Bosch J. Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies. *J Softw Evol Process*. 2018;30(12):e2113.
18. Auer F, Lee CS, Felderer M. Continuous experiment definition characteristics. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Portoroz, Slovenia; 2020:186-190.
19. Giaimo F, Berger C. Continuous experimentation for automotive software on the example of a heavy commercial vehicle in daily operation. arXiv preprint arXiv:200303799; 2020.
20. Mattos DI, Bosch J, Olsson HH, Korshani AM, Lantz J. Automotive A/B testing: Challenges and lessons learned from practice. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Portoroz, Slovenia; 2020:101-109.
21. Olsson HH, Bosch J. The hypex model: From opinions to data-driven software development. *Continuous software engineering*: Springer; 2014:155-164.
22. Mattos DI, Dmitriev P, Fabijan A, Bosch J, Olsson HH. An activity and metric model for online controlled experiments. In: *International Conference on Product-Focused Software Process Improvement Springer*. Wolfsburg, Germany; 2018:182-198.
23. Fagerholm F, Guinea AS, Mäenpää H, Münch J. The right model for continuous experimentation. *J Syst Softw*. 2017;123:292-305.
24. Bosch J, Olsson HH, Björk J, Ljungblad J. The early stage software startup development model: A framework for operationalizing lean principles in software startups. In: *International Conference on Lean Enterprise Software and Systems Springer*. Galway, Ireland; 2013:1-15.
25. Olsson HH, Bosch J. Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation. In: *International Conference of Software Business Springer*. Braga, Portugal; 2015:154-166.
26. Fagerholm F, Guinea AS, Mäenpää H, Münch J. Building blocks for continuous experimentation. In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering ACM*. Hyderabad, India; 2014:26-35.
27. Yin RK. *Case study research and applications: Design and methods*. Thousand Oaks, California, US: Sage publications; 2017.
28. Easterbrook S, Singer J, Storey M-A, Damian D. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering*. London: Springer; 2008:285-311.
29. Maxwell JA. *Qualitative research design: An interactive approach*, Vol. 41. Thousand Oaks, California, US: Sage publications; 2012.
30. Walsham G. Interpretive case studies in is research: Nature and method. *European J Inform Syst*. 1995;4(2):74-81.
31. Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw Eng*. 2009;14(2):131-164.
32. Braun V, Clarke V. Using thematic analysis in psychology. *Qualitative Res Psych*. 2006;3(2):77-101.
33. Campbell DT, Stanley JC, Gage NL. Experimental and quasi-experimental designs for research; 1963.
34. Bickman L. *Validity and social experimentation*, Vol. 1. Thousand Oaks, California, US: Sage; 2000.
35. Shadish WR, Cook TD, Campbell DT, et al. *Experimental and quasi-experimental designs for generalized causal inference/William R. Shedish, Thomas D. Cook, Donald T. Campbell*. Boston: Houghton Mifflin; 2002.
36. Ely M, Anzul M, Vinz R, Downing M. *On writing qualitative research: Living by words*. Florida, US: Psychology Press; 1997.
37. Olsson Holmström H. So much data-so little value: A multi-case study on improving the impact of data-driven development practices. In: *20th Iberoamerican Conference on Software Engineering (CIbSE 2017)*. Buenos Aires, Argentina; 2017:249-262.
38. Zhang X. *LTE optimization engineering handbook*. Beijing, China: John Wiley & Sons; 2018.
39. Zain ASM, Malek MFA, Elshaikh M, Omar N. Optimization of resource allocation using taguchi's method for LTE-advanced network. In: *2014 2nd International Conference on Electronic Design (ICED)*. Penang, Malaysia: IEEE; 2014:281-286.
40. Mattos DI, Bosch J, Olsson HH, Dakkak A, Bergh K. Automated optimization of software parameters in a long term evolution radio base station. In: *2019 IEEE International Systems Conference (SYSCON)*. Orlando, FL, USA: IEEE; 2019:1-8.
41. Montgomery DC. *Design and analysis of experiments*. Hoboken, NJ, US: John Wiley & Sons; 2017.
42. Soligon SD, Lixandrão ME, Biazon TMPC, Angleri V, Roschel H, Libardi CA. Lower occlusion pressure during resistance exercise with blood-flow restriction promotes lower pain and perception of exercise compared to higher occlusion pressure when the total training volume is equalized. *Physiology Int*. 2018;105(3):276-284.
43. Mattos DI, Bosch J, Olsson HH. Multi-armed bandits in the wild: Pitfalls and strategies in online experiments. *Inform Softw Technol*. 2019;113:68-81.
44. Xu Y, Chen N. Evaluating mobile apps with A/B and quasi A/B tests. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, US; 2016:313-322.
45. Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N. Taking the human out of the loop: A review of bayesian optimization. *Proc IEEE*. 2015; 104(1):148-175.
46. Awada A, Wegmann B, Viering I, Klein A. Optimizing the radio network parameters of the long term evolution system using taguchi's method. *IEEE Trans Vehic Technol*. 2011;60(8):3825-3839.
47. Karna SK, Sahai R, et al. An overview on taguchi method. *Int J Eng Math Sci*. 2012;1(1):1-7.

48. Tamburrelli G, Margara A. Towards automated A/B testing. In: *International Symposium on Search Based Software Engineering*. Fortaleza, Brazil; 2014: 184-198.
49. Dastoor S, Dalal U, Sarvaiya J. Comparative analysis of optimization techniques for optimizing the radio network parameters of next generation wireless mobile communication. In: *2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN)*. Mumbai, India: IEEE; 2017:1-6.
50. Middleton JA, Aronow PM. Unbiased estimation of the average treatment effect in cluster-randomized experiments. *Stat Politics Policy*. 2015;6(1-2): 39-75.
51. Raudenbush SW. Statistical analysis and optimal design for cluster randomized trials. *Psychological Methods*. 1997;2(2):173.
52. Gui H, Xu Y, Bhasin A, Han J. Network A/B testing: From sampling to estimation. In: *Proceedings of the 24th International Conference on World Wide Web ACM*. Florence, Italy; 2015:399-409.
53. Xu Z, Kalbfleisch JD. Propensity score matching in randomized clinical trials. *Biometrics*. 2010;66(3):813-823.
54. Rahman MT, Querel L-P, Rigby PC, Adams B. Feature toggles: Practitioner practices and a case study. In: *Proceedings of the 13th International Conference on Mining Software Repositories*. Austin, Texas, US; 2016:201-211.
55. Neely S, Stolt S. Continuous delivery? Easy! Just change everything (well, maybe it is not that easy). In: *2013 Agile Conference*. Nashville, TN, USA; 2013:121-128.
56. Feitelson DG, Frachtenberg E, Beck KL. Development and deployment at facebook. *IEEE Int Comput*. 2013;17(4):8-17.
57. Pakarinen E, Harakkamäki T, Mikkonen T. Gradual deployment in practice: Experiences from an industrial case study. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Portoroz, Slovenia; 2020:237-241.
58. Xu Y, Duan W, Huang S. SQR: Balancing speed, quality and risk in online experiments. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London, UK; 2018:895-904.
59. Xia T, Bhardwaj S, Dmitriev P, Fabijan A. Safe velocity: A practical guide to software deployment at scale using controlled rollout. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Montreal, QC, Canada: IEEE; 2019:11-20.
60. Ståhl D, Bosch J. Cinders: The continuous integration and delivery architecture framework. *Inform Softw Technol*. 2017;83:76-93.
61. Rodríguez P, Haghighatkhah A, Lwakatare LE, et al. Continuous deployment of software intensive products and services: A systematic mapping study. *J Syst Softw*. 2017;123:263-291.
62. Johnson DE. Crossover experiments. *Wiley Interdisciplin Rev: Comput Stat*. 2010;2(5):620-625.
63. Fabijan A, Dmitriev P, Olsson HH, Bosch J. The benefits of controlled experimentation at scale. In: *2017 43rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. Vienna, Austria: IEEE; 2017:18-26.
64. Fabijan A, Dmitriev P, Olsson HH, Bosch J, Vermeer L, Lewis D. Three key checklists and remedies for trustworthy analysis of online controlled experiments at scale. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Montreal, QC, Canada: IEEE; 2019:1-10.
65. Giaimo F, Berger C. Design criteria to architect continuous experimentation for self-driving vehicles. In: *2017 IEEE International Conference on Software Architecture (ICSA)*. Gothenburg, Sweden: IEEE; 2017:203-210.
66. Bosch J, Eklund U. Eternal embedded software: Towards innovation experiment systems. In: *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation Springer*. Heraklion, Crete, Greece; 2012:19-31.

**How to cite this article:** Issa Mattos D, Dakkak A, Bosch J, Olsson HH. The HURRIER process for experimentation in business-to-business mission-critical systems. *J Softw Evol Proc*. 2023;35(5):e2390. doi:10.1002/smr.2390