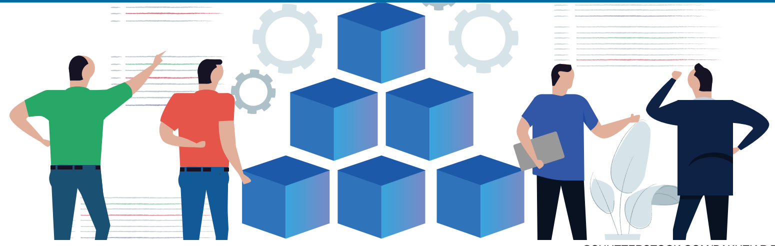
An Ecosystem for the Large-Scale Reuse of Microservices in a Cloud-Native Context

Muhammad Usman and Deepika Badampudi, Blekinge Institute of Technology

Chris Smith and Himansu Nayak, Ericsson AB

// This article presents an ecosystem that Ericsson developed to systematically practice large-scale reuse of microservices in a cloud-native context. We discuss how various ecosystem aspects facilitated the development and reuse of microservices across Ericsson. We also share lessons learned while developing the ecosystem. //



©SHUTTERSTOCK.COM/BAKHTIAR ZEIN

THE USE OF microservices in delivering software-intensive products has been on the rise during the last few years.[1,2] Many companies have shared their experiences of using microservices to achieve benefits such as faster delivery, scalability, and independent deployment.[1–3] Microservices are used primarily as a cloud-native architectural style, in combination with various DevOps practices (for example, continuous deployment) and container-based solutions, to quickly deliver and scale software-intensive products.[2–4] In this article, we share Ericsson's journey of moving toward a cloud-native approach of developing software applications—focusing on how technology, practices, and teams are combined in an ecosystem to collaboratively develop common functions as microservices that can be reused and integrated across all applications within Ericsson. We also share lessons learned in developing the ecosystem, which is referred to as the *application development platform* (*ADP*) ecosystem.

An Overview of the ADP Ecosystem

Ericsson developed the ADP ecosystem to support its transition to the cloud-native approach of developing applications and services. Figure 1 provides an overview of the current state of the ADP ecosystem. The ADP ecosystem provides technical, process, and organizational support for developing cloud-native applications (CNAs) or cloud network functions within Ericsson. The ADP ecosystem is based on a modern architectural style—microservices and containers. The ecosystem also supports more large-scale reuse of microservices across applications within Ericsson. Due to their size, bounded context, and loose coupling, microservices are

Digital Object Identifier 10.1109/MS.2022.3167447
Date of current version: 22 August 2022

potentially a good unit of reuse and can thus support large-scale reuse across the entire organization.[5]

The large-scale reuse of microservices across Ericsson is not possible unless they are accessible to everyone within the company. The ecosystem developed a marketplace to increase the visibility and knowledge about the microservices that are available for reuse. Furthermore, the ecosystem also provides process and onboarding support to the InnerSource (IS) ways of working in Ericsson's journey to collaboratively develop applications.

ADP microservices are frequently released and assembled into different applications. In such a scenario, it becomes important that microservices have consistent mechanisms for handling issues such as configuration, integration, product and artifact handling, and backward compatibility. The ADP ecosystem has codified these consistency requirements on microservices as design rules (DRs). The ecosystem provides continuous integration and delivery (CI/CD) pipelines that support the development, integration, and delivery of microservices and the assembly of those microservices into applications using Spinnaker (https://spinnaker.io/) as a critical component to orchestrate these pipelines and Helm (https://helm.sh/) to perform deployment and upgrades.

The ecosystem was started in late 2017, and the various aspects described here have emerged during the following four years as the needs became apparent. In spring 2020, due to the increase in the number of DRs, the need for providing a clear milestone-driven pathway for achieving maturity arose. In autumn 2020, the concept of the maturity staircase was established, and DRs were connected with different maturity levels. The connection of different DRs with varying maturity levels has made it easy to understand and express in which order DRs should be implemented by services to achieve a certain level of maturity. It also clarified what users could expect from a microservice at a certain maturity level.

While research on open source software ecosystems is extensive, we know little about internal software ecosystems (ISECOs). ISECOs face

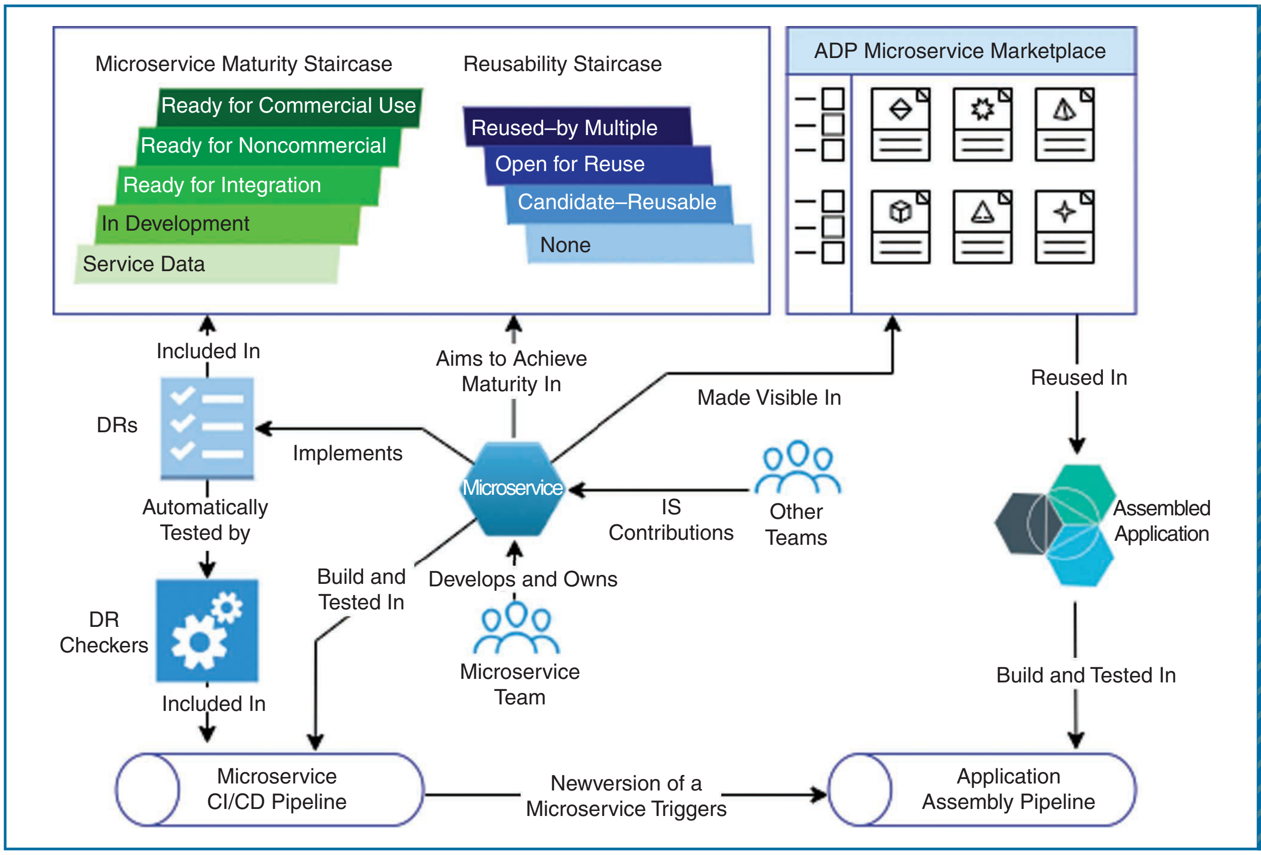


FIGURE 1. An overview of the ADP ecosystem. DRs: design rules; IS: InnerSource.

additional challenges (for example, Siemens's ISECO reports challenges such as the guarantee of software qualities across the ecosystem and compliance with cross-cutting regulations).[6] To address ISECO challenges, the ADP ecosystem provides a more holistic collaboration that is not limited to providing a product/platform for reuse but also supports the IS way of the collaborative development of microservices, automated checking of their compliance with DRs, and their CI into CNAs. In the coming sections, we will elaborate on different dimensions of the ADP ecosystem in detail.

Architectural Framework

The architectural framework in the ADP ecosystem centers around the concept of microservices and containers. It builds on the ecosystem of open source projects from the Cloud Native Computing Foundation (CNCF) (https://www.cncf.io/). The architectural framework consists of the following three key components.

Container Orchestration

The ecosystem selected Kubernetes as the container orchestrator and Helm as the packaging and deployment mechanism. The Kubernetes application programming interfaces (APIs) form the portability interface, allowing ADP microservices and applications to use any Kubernetes-certified cloud platform. The CNCF provides the support to software and cloud vendors to certify their offerings and also maintains an inventory of Kubernetes-certified offerings (for more details, see Cloud Native Computing Foundation[7]).

Microservices for Reuse Across Applications

The ecosystem provides various microservices corresponding to different functional areas, such as security, data, network, management, monitoring, and messaging. These services are mostly based on open source projects, and their configuration and lifecycle management are adapted to fit them in the ADP ecosystem. In some areas, their reuse is mandatory to promote consistent user experience and a common way to serve Ericsson customers. The Ericsson teams working with microservices that are based on CNCF projects are also active in the corresponding communities of these open source projects, contributing bug fixes and features where appropriate. Ericsson also contributes to the Kubernetes project. Thus, the larger open source ecosystem (for example, CNCF open source projects) and the ADP ecosystem both learn and benefit from each other.

Ecosystem Support Services and Tools

A large number of individuals and teams from different units across Ericsson participate in the ADP ecosystem: for example, to integrate, reuse, and contribute to microservices. In such a scenario, it becomes necessary to establish some ground rules to avoid inconsistent behaviors and enable automated mechanisms. The ecosystem developed common principles and DRs that need to be followed by all services participating in the ADP ecosystem. The DRs are formulated to ensure consistency across microservices: for example, how to ensure that logs and metrics are consistently structured, how to annotate Kubernetes resources, how to name and version images, or how to ensure a consistent security posture across multiple microservices.

The ecosystem developed a family of DR checkers, which automatically check the compliance of microservices toward these DRs. The developers can integrate DR checkers into their CI pipeline, and the checkers will inform them if they are violating any DRs. These checkers have brought in much-needed automation to the otherwise complex and time-consuming compliance testing process.

The ADP Marketplace

The ADP marketplace is an intranet website developed by Ericsson. The marketplace indexes all microservices that are open for use and contributions. In addition, there is a series of hands-on tutorials and guides attached to the marketplace that guides users on how to use and contribute to microservices and a microservice chassis that developers can use as a starting point for new microservices. As shown in Figure 2, it is possible to view all the microservices and apply filters to search for specific microservices based on their category, area, reusability level, and maturity. Other companies have implemented similar portals for publishing their IS projects.[8] SAP (https://www.sap.com/index.html), for example, also has a similar portal (https://sap.github.io/project-portal-for-innersource) where IS projects are shared for reuse and contributions. The ADP marketplace provides various filters for displaying and grouping the microservices, which are discussed next.

Microservice Category

Ericsson categorizes microservices as follows.

Generic Services and Reusable Services—Both Generic and Reusable Services are expected to be reused in multiple applications and across multiple domains within Ericsson. However, they are funded differently. Generic Services are developed and maintained by dedicated central teams. However, there is no such central funding for

Reusable Services. They are developed, contributed, and maintained by application development teams right across Ericsson. Both types of services are open for IS contributions; however, Reusable Services rely more heavily on the IS community model.

Domain-specific services—These are specific to application teams within the same domain.

Application-specific services—These are specific to an application.

Reusability Level

The reusability level indicates to what extent a service is fit for reuse at a given point. The different levels are:

Level 0—None: This service is not evaluated for reuse.

Level 1—Candidate: This service is potentially reusable and verified to be consistent with ADP architectural principles.

Level 2—Open for reuse: This service is ready to accept contributions and to be included in application assemblies.

Level 3—Reused: This service is effectively used by more than one application.

Microservice Maturity

The microservice maturity indicates the commercial readiness of a microservice and which DRs need to be implemented to meet each level. The different levels are:

Level 0—Idea/PoC: This is the proof-of-concept and experimental stage.

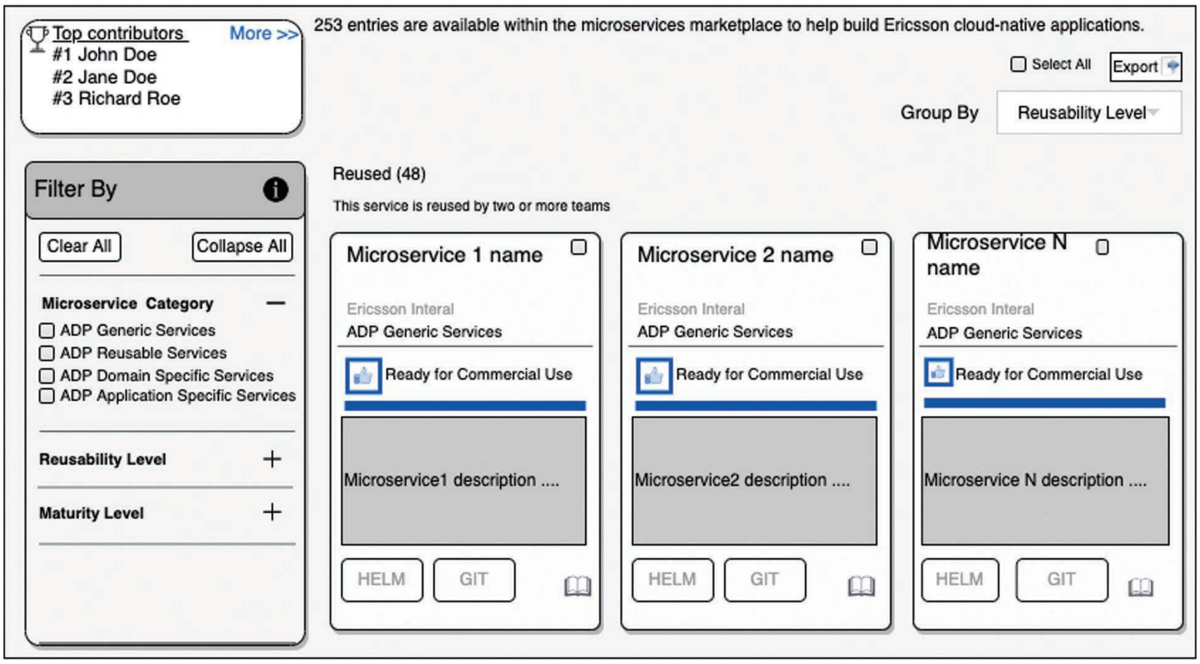
Level 1—Development started: This level indicates development ongoing with the intention to reach higher maturity levels.

Level 2—Ready for integration: This level can coexist with other ADP services and has a CI pipeline that can be connected to an application staging environment.

Level 3—Ready for noncommercial use: This indicates limited release for use in demos or testing at customer site.

Level 4—Ready for commercial use: This level is telecommunications company-grade ready and can be used in commercial deployments.

For each microservice, the marketplace provides the following information: the overview, the documentation, compliance (met/not met/exempted), Helm charts to install the microservice, a link to the source code repositories, the team owning the microservice, discussions related to the microservice, and finally, the list of contributors for the microservice. In addition, the marketplace appreciates


253 entries are available within the microservices marketplace to help build Ericsson cloud-native applications.

Top contributors More >>
#1 John Doe
#2 Jane Doe
#3 Richard Roe

Group By: Reusability Level

Reused (48)
This service is reused by two or more teams

Microservice 1 name	Microservice 2 name	Microservice N name
Ericsson Interal	Ericsson Interal	Ericsson Interal
ADP Generic Services	ADP Generic Services	ADP Generic Services
Ready for Commercial Use	Ready for Commercial Use	Ready for Commercial Use
Microservice1 description	Microservice2 description	Microservice N description ....

FIGURE 2. The microservices marketplace.

the top contributors to encourage more contributions.

More than 250 microservices have been made available on the marketplace for potential reuse and collaboration by January 2022. These include both functional and infrastructure-related services (for example, metrics). Seventy of these 250 have reached the highest level of maturity; that is, they are ready for commercial use and are being reused by multiple applications.

CI/CD

The ADP CI/CD strives to enable Ericsson-wide reuse of services developed by any unit. If an application development unit finds a microservice in the ADP marketplace that satisfies one of their needs, they can integrate it to try it out. However, a one-shot integration is of course not acceptable—and so, to enable this kind of reuse, it must be possible to connect onto the CI/CD pipeline of the reused microservice also to receive and integrate new versions of the microservice as and when they become available. It is important to note that the CI and release of microservices are decoupled from the continuous assembly and release of applications. When a new version of a microservice is released, it is pushed immediately to the staging environment of those applications that have integrated it to test if it should be included in the applications baseline in place of the previous version.

There is no central service governance policy that applications must follow. However, the applications are aware of:

1. the benefits of switching to the latest microservice versions—that is, updates and corrections are delivered in the latest versions
2. the risks of staying with the older versions—that is, older versions are more likely to result in several vulnerabilities in the security scans that are mandatory before release.

The microservice maturity indicates the commercial readiness of a microservice and which DRs need to be implemented to meet each level.

The ADP CI/CD uses Spinnaker pipelines to connect the individual microservices with the staging environments of different applications (see Figure 3). The ADP development environment provides Spinnaker pipelines as a service. An application may consume one or more microservices from elsewhere. Each connection point (arrows in Figure 3)—between the individual microservice and the staging environment of an application—represents one reuse instance. There are more than 600 such connection points, demonstrating a large-scale reuse of microservices by applications. Some microservices are used only in one or two applications, but the most reused ones are currently included in as many as 46 applications. These include both functional and infrastructure services.

The use of software flow principles, Spinnaker, and its interfaces is mandated. However, the teams still have a high degree of freedom behind these mandated interfaces to use the most appropriate CI technology in their pipelines. Despite this autonomy, the ecosystem encourages teams to share their experiences of using particular CI/CD technologies to facilitate community-based implementations.

The ADP Program

There is a central organization—the ADP Program, which is responsible for sustaining the ecosystem. The ADP Program includes an architecture team that drives and governs principles, DRs, and guidelines related to microservice architecture and the cloud-native approach, enabling alignment in how microservices are developed and maintained. The governance body is an architecture council where ADP architects meet with an applications' architects to agree on architecture direction and any further improvements related to principles, DRs, and guidelines. The ADP Program also includes the development teams (more than 100 developers in teams of varying sizes) that own the development and maintenance of more than 50 Generic Services that are heavily reused across Ericsson.

The ADP Program includes a unit (the ADP Anchor Unit) that is tasked with devising and evolving the structures and policies of the ecosystem.

This unit also works to make it easier to create, release, and reuse microservices—the unit can be compared to the role that the CNCF takes in the open source Cloud Native Community. More than 100 practitioners, independent of the development teams working on the centrally funded Generic Services, perform these anchoring activities. The name *Anchor* was given because this unit "anchors" the ecosystem into the company structure—wherein individual services, applications, and tools are developed, owned, and contributed to by many different organizations across Ericsson. To keep track of the progress, the ADP Anchor Unit collects several metrics, such as the number of IS contributions made by different units, status of DR fulfillment by each ADP microservice, and pass/fail rate of each microservice's integration into different user applications.

Adopting IS to Sustain the ADP Ecosystem

The IS contributions are critical to sustain the ADP ecosystem; otherwise, it could become a bottleneck wherein the service users are demanding only new features, resulting in unmanageable backlogs to be handled by the central teams, as experienced in other similar contexts.[9,10] Taking inspiration from the IS Commons,[8] the ADP ecosystem has defined IS roles and principles to support the IS collaboration model within Ericsson.

IS Roles

The ADP services are handled as IS projects. Each IS project needs to have the following roles.

Guardians: Guardians, being the technical owners of the service, are responsible for providing contribution guidelines, participating in the relevant discussion forums, and reviewing and approving feature requests and contributions. The guardian is normally part of the development unit that created the ADP service and the corresponding IS project.

Trusted committers: Trusted committers,[8] who are more experienced and frequent contributors who have the required knowledge to guide new contributors, participate in discussion forums and the review process for discussing contribution requests. The recognition of active contributors as trusted committers is pivotal in creating a sustainable community for an ADP service.

Product owners: Product owners are responsible for maintaining the

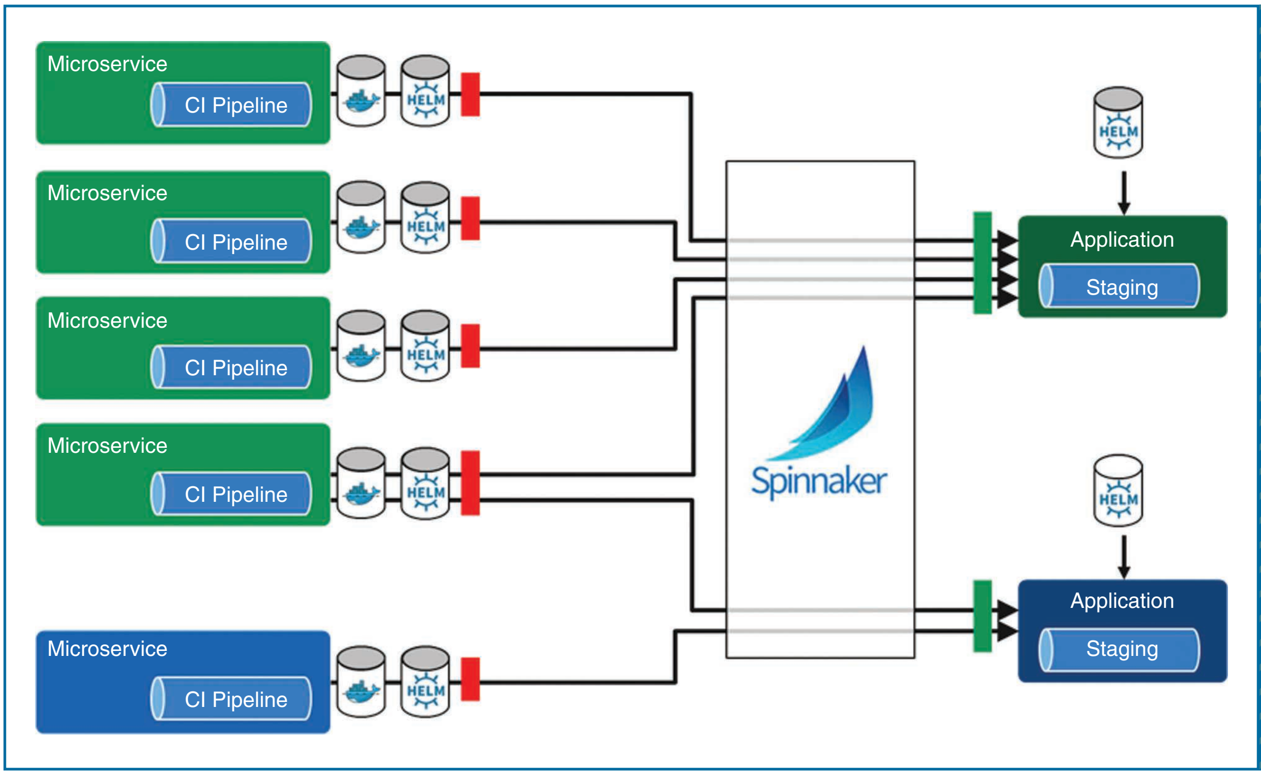


FIGURE 3. Spinnaker pipelines.

product backlog, prioritizing features for the upcoming releases, and collaborating with other stakeholders, in particular with the guardians, to discuss how upcoming contributions fit to the ADP service road map.

IS Principles

To enable IS contributions to ADP services, the ADP Program has defined some basic principles:

- The source code of the ADP services shall be visible through the ADP marketplace, but as a quality control mechanism, patches can be integrated only by the relevant guardians or trusted committers after review.
- The backlog of ADP services shall be open to all so that potential contributors are informed about the planned features in the future releases.
- The guardian and trusted committers of each ADP service shall create and moderate a forum to discuss bugs, new feature requests, and contributions.
- Before an ADP microservice is included in a shipped product, its legal analysis is performed to ensure compliance with trading regulations and the license requirements of any open source software included.

The ADP Program provides further guidance in the form of examples, templates, FAQs, and tutorials.

The IS Ownership Models

The IS ownership models can be broadly classified into *central* versus *distributed models*. In the central model, both the product and technical ownership are with one organizational unit. In the distributed model, the product ownership still rests with one organizational unit, while the technical ownership is shared across multiple units. As the community around the IS project grows, it becomes possible to assign the trusted committer role to a few contributors from other organizational units. The trusted committer could then share the technical ownership of the IS project. The distributed model is particularly important for Reusable Services in the ADP ecosystem to foster a sustainable community of trusted committers who can manage the IS project even when the original guardians are not available.

Most developers appreciate the checkers as they allow them to quickly verify their code—to see if they have done a good job of conformance to the DRs.

Lessons Learned

With the help of an exploratory case study (the complete case study, with methodological details and other findings, is to be published separately), we identified the following lessons learned by Ericsson in developing the ADP ecosystem.

IS contributions prevent the ADP ecosystem from becoming a bottleneck: It is important for the teams creating Generic and Reusable Services to be open to receiving contributions. In addition, it is also important to provide automated tools and software development kits that make it easier to contribute.

Tool support for checking the DRs: The ecosystem developed a family of checkers to automatically check the conformance of the DRs. The developers only need to integrate the relevant checker in their CI pipeline and let the checker detect the compliance issues, if any. Most developers appreciate the checkers as they allow them to quickly verify their code—to see if they have done a good job of conformance to the DRs.

Open and proactive communication: synchronizing contributions: It is important for the IS contributors to discuss their contribution plan and make adjustments to align with the purpose and architecture of the service before working on the contribution.

Widespread reuse: The widespread reuse of microservices across Ericsson has been made possible by establishing a CI/CD strategy that allows a relatively quick "plug and play" of microservices and applications' assembly pipelines. The key has been to keep the microservices' lifecycle and pipelines independent from the assembled applications' lifecycle and their pipelines.

We aim to improve IS contributions by investigating a sample of

ABOUT THE AUTHORS

MUHAMMAD USMAN is an assistant professor at the Blekinge Institute of Technology (BTH), Karlskrona, 371 79, Sweden. His research interests include empirical software engineering, software process improvement, and software reuse. Usman received his Ph.D. in software engineering from the BTH in 2018. Contact him at muhammad.usman@bth.se.

CHRIS SMITH leads the ecosystem aspects of the Ericsson cloud-native journey (ADP Anchor) at Ericsson AB, Stockholm, 164 40, Sweden. He has worked in the telecom industry for 28 years with platforms for telecom applications. Smith received his B.Eng. from the University of Birmingham, United Kingdom. Contact him at chris.smith@ericsson.com.

DEEPIKA BADAMPUDI is an assistant professor at the Blekinge Institute of Technology (BTH), Karlskrona, 371 79, Sweden. Her research interests include empirical software engineering, software process improvement, and software reuse. Badampudi received her Ph.D. in software engineering from the BTH in 2018. Contact her at deepika.badampudi@bth.se.

HIMANSU NAYAK leads the InnerSource contribution aspects of the Ericsson cloud-native journey (ADP Anchor) at Ericsson AB, Stockholm, 164 40, Sweden. He has been working in the telecommunication industry for 17 years. Nayak received his B.E. in electronics and telecommunication from Utkal University, India. Contact him at himansu.nayak@ericsson.com.

recently completed contributions for identifying the practices that worked well (or otherwise), bottlenecks, and potential improvements in the existing IS contribution practices and guidelines.

Acknowledgment

This work was supported by the Knowledge Foundation through the Open Source Inspired Reuse (OSIR) project (Reference 20190081) at the Blekinge Institute of Technology, Sweden.

References

1. J. Soldani, D. A. Tamburri, and W. J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *J. Syst. Softw.*, vol. 146, pp. 215–232, Dec. 2018, doi: 10.1016/j.jss.2018.09.082.
2. A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables DevOps: Migration to a cloud-native architecture," *IEEE Softw.*, vol. 33, no. 3, pp. 42–52, 2016, doi: 10.1109/MS.2016.64.
3. P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Softw.*, vol. 35, no. 3, pp. 24–35, 2018, doi: 10.1109/MS.2018.2141039.
4. T. Mauro," Adopting microservices at Netflix: Lessons for architectural design," 2015. https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices (Accessed: Nov. 11, 2021)
5. R. Capilla, B. Gallina, C. Cetina, and J. Favaro, "Opportunities for software reuse in an uncertain world: From past to emerging trends," *J. Softw., Evolution Process*, vol. 31, no. 8, p. e2217, 2019, doi: 10.1002/smr.2217.
6. K. B. Schultis, C. Elsner, and D. Lohmann, "Architecture challenges for internal software ecosystems: A large-scale industry case study," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2014, pp. 542–552.
7. "Software conformance (Certified Kubernetes)," Cloud Native Computing Foundation. https://www.cncf.io/certification/software-conformance/ (Accessed: Nov. 10, 2021)
8. "InnerSource portal," InnerSource Patterns. https://patterns.innersourcecommons.org/p/innersource-portal
9. D. Riehle, M. Capraro, D. Kips, and L. Horn, "Inner source in platform-based product engineering," *IEEE Trans. Softw. Eng.*, vol. 42, no. 12, pp. 1162–1177, 2016, doi: 10.1109/TSE.2016.2554553.
10. D. Cooper and K. Jan-Stol, *Adopting InnerSource Principles and Case Studies*. O'Reilly, 2018, pp. 103–117.