Innovative Practices for Knowledge Sharing in Large-Scale DevOps

Aymeric Hemon, Brian Fitzgerald, Barbara Lyonnet, Frantz Rowe

Innovative practices for knowledge sharing in large-scale DevOps

Aymeric Hemon, Brian Fitzgerald, Barbara Lyonnet, Frantz Rowe

Publication date

01-01-2019

Published in

IEEE Software; 37 (3), pp. 30-37

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

Innovative Practices for Knowledge Sharing in Large-Scale DevOps

Aymeric Hemon, ESSCA School of Management and University of Nantes

Brian Fitzgerald, Lero and University of Limerick

Barbara Lyonnet, University of Nantes

Frantz Rowe, University of Nantes and SKEMA Business School

Abstract:

Agile development methods and DevOps require adaptation during implementation to meet the needs of a constantly changing software development environment. The emergence of knowledge-sharing practices for large-scale DevOps has not been the subject of much research. Our in-depth case study, comprising 106 interviews at a large multinational company operating in a DevOps at scale environment, identified a number of innovative practices which had emerged, principally to resolve knowledge-sharing challenges. These practices seem to be more likely to emerge in large-scale DevOps environments. While similar results might have been achieved due to the large-scale nature of the projects, it is difficult to determine definitively whether the main causal factor is project size or DevOps. We believe that self-organization and continuous improvement over a long period of time are also critical influencing factors.

Keywords:
Agile method, DevOps, Large-Scale, Innovative Practices, Knowledge Sharing

1. Introduction

I find myself more and more exasperated with the great inflexible sets of rules that many companies pour into concrete and sanctify as methods...Use the prevailing method only as a starting point for tailoring. -- Tom DeMarco (1982)

Agile methods emerged from the inability of conventional (e.g. waterfall) methods to deliver software to meet the needs of a rapidly changing environment. Initially agile methods were considered to have a "home-ground" where they were best suited, namely small projects, with co-located developers, in non-critical domains. This home-ground has been challenged considerably in the past 15 years as researchers and practitioners have tailored the application of agile methods with new roles, ceremonies and artefacts to meet the needs of the development context, for example in critical and regulated domains.

In addition, a bottleneck has emerged due to a lack of alignment between the Operations function (Ops), coordinating the software release, and the Development function (Dev). Consequently, releases to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

customers took more time. To solve this problem, Debois[1] advocated for more collaboration between the Dev and Ops functions through a tighter integration, a rapprochement called DevOps. This term DevOps comes from the fusion of two words related to specific component activities, Development and Operations.

While agile methods can achieve a more frequent cadence of development of software and a better alignment with customer expectations, DevOps strives for a continuous delivery of value through continuous integration, delivery and deployment. DevOps is hence an extension of agile[2] to the entire software delivery pipeline, aiming to optimize lead time between code writing and its use by end-users in a real production environment.

We consider large-scale DevOps as we focused on the four DevOps pillars (Culture, Automation, Measure, Share) that involve a large number of actors, systems and interdependencies with more than two teams working in the same project[3] and applying continuous integration, delivery and deployment.

Roles and responsibilities do still matter in the agile world, particularly when using prescriptive methods as Scrum. DevOps breaks silos and hence possibly blurs lines (i.e. organizational, hierarchical) regarding jobs, roles, collaborations, responsibilities, skills and practices. DevOps potentially amplifies this blurring between boundaries when applied at large-scale.

In this paper, we focus on innovative practices for knowledge sharing (KS) which is one of the four DevOps pillars and will argue that these practices are more likely to emerge in large-scale DevOps. A KS practice occurs when an individual transfer what he knows to another individual.

When moving to large-scale agile, challenges of KS and related success criteria have been identified[4, 5] which raises the question about ensuring and improving learning and KS practices. Ghobadi et al. identified barriers[6] and risks to effective KS in agile teams. Risk perception is even higher in DevOps than in agile[7]. Consequently, KS challenges are larger and more complex, since more points of view come into consideration when using large-scale DevOps. Managing knowledge dependencies become critical in a large-scale DevOps context[8]. Indeed, some individuals could form a bottleneck to knowledge transfer when moving to large-scale DevOps. Despite the need for innovation and tailoring, as expressed by DeMarco in the opening quote, new KS practices emerging for large-scale DevOps have not been extensively studied. In that respect Nielsen et al. proposed a DevOps knowledge sharing framework (DOKS)[9] and used the CESI dimensions (Combination, Externalisation, Socialisation, Internalisation) to increase awareness of KS modes within the organization. They concluded that the size of the team and the size of the company influence the move towards continuous delivery and DevOps. They showed that larger companies would need a more structured plan when moving to DevOps to ensure KS among their IT teams while smaller companies do not need it. Consequently, moving to large-scale DevOps might impact KS practices.

In sum, the following KS challenges in DevOps have been identified in the literature:

1) More intense cross-functional collaboration between Dev and Ops[10, 11]

2) Multiple environment incompatibilities leading to specialized teams[12, 13]

3) Capability to self-organize[14]

4) Loss of global vision of the project and knowledge of application due to automation[7]

5) Confinement of knowledge sharing to hierarchical organisational structure and process ‘red tape’[7]

6) Limited sharing when parts of development or operations are outsourced[15]

Moving from DevOps to large-scale DevOps amplifies the challenges linked to DevOps, i.e. coordination and collaboration

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

improvement, dependency management, knowledge development and sharing. Three of the six KS challenges above are particularly salient. First, the capability to self-organize could be affected by the scaling-up of DevOps due to a larger number of team members covering a wider organizational perimeter. Second, notwithstanding a higher level of automation at scale, more frequent commits (release) require manual coding adjustment[16] which needs a common coordination mechanism, hence more intense cross-functional collaboration. Third, scaling DevOps could impact KS due to more frictions with organisational structure and process 'red tape'.

2. Context

Our study was conducted in a large multinational company (more than 100,000 employees) that has been practicing DevOps for eight years, being one of the very early adopters of the DevOps approach. We followed the case study method combining interviews, observations and documentation. To identify KS practices implemented by project teams, we carried out direct observations during field visits, attended 20 meetings plus an eight-hour DevOps coaching day, and semi-structured interviews with 106 employees in total. These teams included multiple functions associated with software development, such as developer, project manager, release engineers, UX designers, PO and architects.

For each project, three of the four authors worked together on a deep analysis. Firstly, we collected information regarding the agile methods and actual practices supporting KS both inside and outside of teams and related to the use of space (i.e. common work area), to agile methodologies (i.e. scrum of scrum meetings, daily standup), and to collaborative tools enabling KS (i.e. Slack or Atlassian JIRA). Secondly, we compared our findings with a baseline reference (12th State of Agile Report) to identify potential innovative practices. We studied specific adaptations of agile practices and the development of KS practices which had emerged to suit the contingencies of the development environment. Thirdly, we investigated the literature where we found practices in use in the company but not found in the baseline reference report.

Identifying innovative KS practices thus meant identifying those most advanced within the 18 projects investigated in this company. Those innovative KS practices are likely to be "new to the firm", but we do not claim that they are "new to the world". However, they help address challenges when moving towards large-scale DevOps.

Six out of the 18 projects studied are practicing DevOps, and these innovative KS practices were only found in two of the projects which were using large-scale DevOps. Project X started eight years ago, operating in DevOps mode from the beginning, and Project Y matured to eventually operate in DevOps mode since 2016. Both projects are advanced in DevOps and they practice continuous delivery or continuous deployment.

Project X involves 85 people, organized in seven teams, almost all co-located except for an Ops team located a few miles away. Teams are not necessarily the same size and do not have to be composed the same way. Project members work in DevOps and use mainly a hybrid agile method derived from Scrum, Kanban and XP. They scaled-up their agile method, first with large-scale Scrum (LeSS), then one year later they moved to the Scaled Agile Framework (SAFe). Project Y involves 65 people divided into 5 teams distributed over four geographical sites and two distant countries, distant at many levels according to Ghemawat's CAGE model[17]. Development activities are essentially based in the same country while Operations are mostly in another country. Project members use mainly Scrum and Scrumban and scaled-up with LeSS adopting practices such as Scrum of Scrum. They also implemented an international cross-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

functional daily DevOps meeting.

3. Innovative Practices for Knowledge Sharing

We propose three different levels of KS practices based on three levels of DevOps maturity (agile, continuous integration, continuous delivery) in the transition from agile to large-scale DevOps (see Table 1). At the first DevOps maturity level, agile teams do not cross functional silos and DevOps is inexistent. At the second level, teams practice continuous integration, hence across silos. At the third level, teams practice continuous delivery and/or deployment. We identify four practices which have emerged through tailoring of the method and which address the needs of DevOps in a large-scale development context: Cross-Functional Dynamic Role Rotation, Technical Thursdays, Heads-Up Grooming & Planning, and the Circle. We discuss each in turn below.

Table 1. Knowledge Sharing Practices in Transition to Large-Scale DevOps (After [15])

Maturity Level	*Characteristic of KS Practice*	*Path to Large-Scale DevOps*
Level 1: Agile	• More frequent KS due to iterative process and more frequent releases. • Better communication and sharing between customer and developers. • Limited sharing among the team and little common culture with PO.	• Large-Scale agile methods are used (e.g. SAFe, LeSS). • DevOps is not achieved because silos still exist. • KS is limited to specific silos (mostly Dev, Biz to a lower level).
Level 2: Continuous Integration	• KS through the performance of various tests (unit and non-regression tests) synchronized with code development. • Automation as far as possible, task automation knowledge transfer. • Partial KS. Developers need quick feedbacks. • Partial common vocabulary and culture boosting KS. • Some common tools fostering KS. • Different metrics and measurement systems limiting KS.	• Some large-scale frameworks are used and meet their first limits (i.e. rigidity of SAFe) along the entire pipeline • Alignment of Ops function on Dev function is achieved. • Some DevOps pillars like Sharing are partially reached. • KS is applied across silos between Dev and Ops.
Level 3: Continuous Delivery	• Integration tests with other components, end-to-end tests, performance tests, user acceptance tests are then co-designed, performed and preferably automated by Ops in conjunction with the Dev function. • Learning and Extensive KS. • Shared backlog. • Shared work system. • Shared metrics and measurement	• Large-scale frameworks are particularly challenged on KS along the entire pipeline • Higher degree of alignment, sharing and automation. • Full DevOps practices on 4 pillars with high level of KS across teams and beyond.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

	tools. • Innovative KS practices appear.	

3.1 Cross-Functional Dynamic Role Rotation (DRR)

A more significant involvement of certain functions (Project Manager, PO and Scrum Master) was highlighted during Daily Standup Meetings. In some projects, this involvement was even more pronounced as several of these functions were performed by the same individuals. To overcome this difficulty, one of the large project teams at an advanced level of DevOps maturity proposed a new practice which involved a cross-functional DRR and therefore associated responsibilities. Job rotation in the software development context has been identified in a previous study[18] as a mean of addressing organizational concerns and stimulating innovation. The instantiation of this role rotation in the cross-functional DRR practice in our case enabled large-scale learning and KS since all team members were able to perform several roles and became more knowledgeable. DRR is usually limited to a specific area i.e. the development team or the operations team. A developer can become a scrum master for a week or a sprint, then he goes back to his position. In our cases, DRR is cross-functional and for instance, Production Engineers (Ops) performed code reviews. All members from both sides, Dev and Ops, were able to meet and work together alternatively. Cross-functional DRR also ensured that the 'heavy-lifting' workload did not fall on the same individuals repeatedly. It also fulfilled the role of 'succession planning' in that team composition became more flexible as individuals could be 'swapped out' without causing major perturbation to the extent that the teams could not function.

3.2 Technical Thursdays (TT)

Another practice which emerged in the context of technical knowledge-sharing was labelled "Technical Thursdays". These 'tech-talk' events took place over a half-day every two weeks. The goal was to disseminate and share technical knowledge, particularly in relation to technologies, tools, skills, or other topics. These would typically be championed and led by an individual team member. They could take the form of workshops on new technology topics (e.g., containerization, automated configuration management, infrastructure as code), and related tooling (e.g., Puppet, Docker), discussions of new initiatives (e.g., A/B experimentation) challenges (e.g. hackathons), games (e.g. Code Wars). A playful spirit and gamification were often a strong component of this practice, which had the extra benefit of boosting morale and team spirit.

The difficulty with these tech-talks lies in the fact that KS could be limited to a group a person or to communities of practice (CoP). Therefore, inter-team KS at large scale could be an issue for agile software development organizations[19]. The originality of these TT events is that they go beyond CoP and the risk of developing knowledge on a tribal basis. TT are fully cross-functional, both vertically and horizontally across silos where they existed. One could have supposed the Code War game was only intended for Developers. However, POs, Architects, Designers, Developers, Testers, Ops Engineers have been invited to this game. Participants participated in teams other than their usual one. Teams had to be mixed and represent

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

as many trades as possible at the organization level. While hackathons are mostly frequented by developers, this game fostered large-scale KS across the Dev and Ops areas and significantly helped the realization of projects, i.e. the implementation of Docker containers for deployment system proposed by Ops and accepted by Dev beyond project teams.

3.3 Heads-Up Grooming and Planning (HUGP)

Backlog Grooming, or backlog refinement, can be understood in different ways. In the 18 projects we studied, backlog grooming consisted of short-term User Story costing. However, in a large project where the team uses DevOps, one PO had instantiated a new practice to support backlog grooming and Sprint Planning, which is labelled as Heads-Up Grooming and Planning (HUGP). This practice is similar to Rolling Lookahead Planning (RLP) in the sense that there is a high-level release plan and sprints are then gradually refined as they start. In this HUGP ceremony, the PO no longer performs any immediate user story estimation with the team. Instead, the PO just provides the team with broad information on the subject. The idea is to give the team a visualization of the work to come. This meeting is short - somewhere between 10 minutes and a maximum of 30 minutes. Subsequently, the Sprint Planning meeting takes place and lasts two hours where user stories are presented, estimated and broken down into tasks. However, the effect of the previously held HUGP practice is very noticeable. The PO describes it well:

The team is more confident and self-assured when it comes to story point estimation. There is less error or difference. The team members have had time to go through the code and figure out if they had any doubts. They discuss it among themselves, around a foosball game.

We consider HUGP as an extension of RLP. Both adopt a just-in-time approach to up-front planning; both foster discussion within teams for identification and estimation of product backlog components. However, we suggest that HUGP differs because it is more than a discussion about user story estimation. HUGP strongly promotes KS, pushing team members to go into the code, to learn, to test something new, to enrich each other and even to consult Ops engineers on specific topic (i.e. production environment, deployment package). Before the discussion, there is a transfer of knowledge between members and this helps them to surface their arguments for better estimates. The benefits of such KS practices are multiple: time-saving when estimating user stories, clearer and more accurate estimations, engendering a greater sense of Dev team empowerment, at the DevOps team level, then across the project.

3.4 The Circle

One issue with Sprint Retrospectives is that, if they are done at all, they are often combined with the Sprint Review ceremony. Also, they are confined to a single team, hence limiting KS[19], thereby minimizing the opportunity for KS in achieving continuous improvement and consequently not fostering Agile and DevOps at large-scale.

The team met difficulties with retrospectives, primarily because they were quite complex to organize with a team of 85 people. They decided to set up joint reflection ceremonies, which were known as the Circle. This practice is based on discussion among a group of people who represent all the jobs and roles across multiple project teams, i.e. managers, PO, architects, developers, testers, Ops engineers. Participation is voluntary. The circle is a forum for real exchange of knowledge across all team functions. This Circle was composed of about 20 people who met twice a month. The Circle practice fulfilled the role of the Sprint Retrospective at large-scale by stimulating and animating reflection on the daily functioning of teams

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

and the organization but had the added advantage of comprising multiple teams. This reflection was particularly useful for implementing the other innovative practices mentioned here, as the use of new practices does not always get the support of all, and therefore their implementation must be the subject of open discussion.

When the shared knowledge appeared valuable, e.g. Code Wars, it is adopted and disseminated across teams. In this way, they limit the problem of missing valuable knowledge or not capitalizing on this knowledge. The Circle allowed such discussion to take place and ensured that all these innovative practices could evolve and develop, much in the spirit of the quote by DeMarco above. This practice typically responded to KS challenge when large-scale DevOps was applied (see Table 1) and went beyond DevOps boundaries integrating the Biz domain also.

3.4 KS Major Challenges Resolution

Different innovative practices have met the challenges of KS in the context of large-scale DevOps. We summarize the main KS challenges and impacts generated by each of them (see Table 2).

Table 2. Major Challenges and Impacts of Innovative Practices

Innovative practices	Major challenges addressed	Impact
Cross-Functional DRR	- Intense cross-functional collaborations - Team self-organization challenges.	• Enables the breaking of knowledge dependencies [15] because knowledge is not anymore hold by an individual. However, this obliges team members to communicate amongst themselves to carry out their role effectively. • When team members rotate, they can punctually take on responsibilities, develop skills and acquire knowledge. This fosters autonomy of the team.
Technical Thursdays	- More intense cross-functional collaboration -Complex environments and specialized teams.	• All team participants (representing almost every area, i.e. business, development, quality...) can discover and learn new approaches without any consideration regarding the belonging to a specific functional silo or the position held. This is how Ops teams by passed organizational culture related to training and Ops made Dev teams discover Rancher, an open source tool to deploy and manage containers in production environments. • Helps to prevent from an over specialization of teams by fostering KS and by increasing the knowledge base or individuals understanding of these multiple environmental complexities and incompatibilities.
Heads-Up Grooming and Planning	-Confinement of KS due to hierarchical and organizational structure and process red tape.	• Facilitates KS at scale avoiding the repetition of red tapes and reduces the burden of agile processes and organizational rules and procedures. • By practicing HUGP, the manager gives the team members the possibility to obtain central information that they would not have had while respecting the processes of the agile method and the organization. Team members can therefore use information in advance which allows them to enrich and transform the information into knowledge. This new knowledge is shared and allows them to make more realistic and accurate decisions in their mission.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MS.2019.2958900, IEEE Software

The Circle	-Capability to self-organize -Multiple environment incompatibilities leading teams' specialization.	• Facilitates the circulation of knowledge because the Circle is fully cross-functional with representants from all jobs linked to the project and, in the same time, because the circle is only 20 individuals representing larger teams (85 people). • This fosters KS at large-scale, decision-making and consequently teams' autonomy.

4. Large-Scale DevOps Enabler of Innovative KS Practices

While the Scrum method is the backdrop for the projects we surveyed, reality is much more complex in terms of overlapping and complementary practices[20]. Thus, we were able to identify the development of innovative KS practices only in the most advanced project (see Table 1). This raises the question of how large-scale DevOps enables innovation. Our hypothesis is that these innovation practices are the result of further continuous improvement, facilitated by stronger levels of self-organization in DevOps teams acting on a broader mandate. As already mentioned, these KS practices were in fact observed in large-scale projects which commenced eight years ago, one of them operating in DevOps mode from the beginning, and the other maturing to eventually operate in DevOps mode since 2016. The other four DevOps projects were clearly sharing knowledge (e.g. common project management, common processes, common tools e.g. shared continuous integration and automated deployment tools, co-construction of deliverables) and used basic practices. However, KS was less developed than the other two. This may be because they had had less time to develop those, but also because three of these four were smaller projects.

Interestingly, in this large company, these innovative KS practices were not shared across all projects, even at the same level of software process maturity. These innovative KS practices addressed specific challenges faced by projects and helped the organization resolving initial problems partially or fully. Large-scale DevOps may require more KS practices since additional constraints have to be integrated between the business, development and operations functions. Such KS practices respond to projects operating in both large-scale and DevOps because such joint conditions multiply coordination issues and risks related to lack of competencies[15]. Moreover, we believe that project autonomy or self-organization and continuous improvement over a long period of time are also critical influencing factors. Thus, in the teams who developed KS practices:

- Projects were more advanced in agile and tended towards large-scale DevOps
- The larger they were, the more they were self-organized and used their relative autonomy to continuously improve over a long period of time,
- Also more they were more likely to develop innovative KS practices within their teams and across functional boundaries.

Clearly the emergence and implementation of these innovative KS practices is an outcome of multiple idiosyncratic conditions and we would need more data beyond the 18 projects studied here to reach any firm conclusion. However, observations in our ongoing research on this topic in other firms tend to support a large-scale DevOps effect and a continuous improvement effect related to self-organization in the generation of innovative KS practices. Using DevOps at large-scale amplifies the KS challenge as well as related solutions through the

development of innovative practices. Such effects are not fixed, as the implementation of these practices is not fixed, and is subject to continuous adjustment to avoid deterioration and rigidity, or indeed being poured into concrete!

References

[1] P. Debois, "Devops: A software revolution in the making," *Journal of Information Technology Management*, vol. 24, no. 8, pp. 3–39, 2011.

[2] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," in *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*, 2015, pp. 78–82.

[3] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What is large in large-scale? A taxonomy of scale for agile software development," in *International Conference on Product-Focused Software Process Improvement*, 2014, pp. 273–276.

[4] T. Dingsøyr and N. B. Moe, "Research challenges in large-scale agile software development," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 38–39, 2013.

[5] D. Smite, N. B. Moe, G. Levinta, and M. Floryan, "Spotify Guilds: How to Succeed With Knowledge Sharing in Large-Scale Agile Organizations," *IEEE Software*, vol. 36, no. 2, pp. 51–57, 2019.

[6] S. Ghobadi and L. Mathiassen, "Perceived barriers to effective knowledge sharing in agile software teams," *Information Systems Journal*, vol. 26, no. 2, pp. 95–125, 2016.

[7] A. Hemon, L. Monnier-Senicourt, and F. Rowe, "Job Satisfaction Factors and Risks Perception: An embedded case study of DevOps and Agile Teams," in *Proceedings of the 39th International Conference on Information Systems (ICIS)*, San Francisco, CA, USA, 2018.

[8] V. Stray, N. B. Moe, and A. Aasheim, "Dependency management in large-scale agile: a case study of DevOps teams," in *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*, Grand Wailea, Maui, HA, USA, 2019.

[9] P. A. Nielsen, T. J. Winkler, and J. Nørbjerg, "Closing the IT Development-Operations Gap: The DevOps Knowledge Sharing Framework," in *16th International Conference on Perspectives in Business Informatics Research. BIR 2017*, Copenhagen, Denmark, 2017, pp. 1–15.

[10] W. Gottesheim, "Challenges, benefits and best practices of performance focused DevOps," in *Proceedings of the 4th International Workshop on Large-Scale Testing*, 2015, pp. 3–3.

[11] A. Wiedemann, "A New Form of Collaboration in IT Teams-Exploring the DevOps Phenomenon," in *Proceedings of the 21st PACIS Pacific Asia Conference on Information Systems*, Langkawi, 2017, vol. 82, pp. 1–12.

[12] L. E. Lwakatare *et al.*, "Towards DevOps in the embedded systems domain: Why is it so hard?," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, 2016, pp. 5437–5446.

[13] J. Smeds, K. Nybom, and I. Porres, "DevOps: a definition and perceived adoption impediments," in *International Conference on Agile Software Development*, 2015, pp. 166–177.

[14] A. Wiedemann, "IT Governance Mechanisms for DevOps Teams-How Incumbent Companies Achieve Competitive Advantages," in *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS)*, 2018.

[15] A. Hemon, B. Lyonnet, F. Rowe, and B. Fitzgerald, "Conceptualizing the Transition From Agile to DevOps: A Maturity Model for a Smarter IS Function," in *IFIP WG 8.6 Working Conference*, Portsmouth, UK, 2018.

[16] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "The impact of continuous integration on other software development practices: a large-scale empirical study," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 2017, pp. 60–71.

[17] P. Ghemawat, "Distance still matters," *Harvard business review*, vol. 79, no. 8, pp. 137–147, 2001.

[18] T. E. Fægri, T. Dybå, a, and T. Dingsøyr, "Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work," *Information and Software Technology*, vol. 52, no. 10, pp. 1118–1132, 2010.

[19] V. Santos, A. Goldman, and C. R. De Souza, "Fostering effective inter-team knowledge sharing in agile software development," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1006–1051, 2015.

[20] B. Fitzgerald, G. Hartnett, and K. Conboy, "Customising agile methods to software practices at Intel Shannon," *European Journal of Information Systems*, vol. 15, no. 2, pp. 200–213, 2006.

Aymeric Hemon is an Assistant Professor in MIS at Essca School of Management and a researcher at the LEMNA Laboratory, University of Nantes. He is also Senior Lecturer in Software Engineering at the Mines Telecom Institute. His research focus on IS-enabled organizational transformation, particularly on agile methods and DevOps. Contact him at Essca School of Management, 1 rue Joseph Lakanal - BP 40348, 49003 Angers Cedex 01, France, aymeric.hemon-hildgen@essca.fr

Brian Fitzgerald is Director of Lero - the Irish Software Research Centre. He also holds an endowed professorship, the Krehbiel Chair in Innovation in Business & Technology, at the University of Limerick, Ireland. His publications include 15 books and more than 150 peer-reviewed articles in the leading international

journals and conferences in both the Information Systems and Software Engineering fields. Prior to taking up an academic position, he worked in the software industry for about 12 years. Contact him at University of Limerick, LERO, Castletroy, Limerick, V94 T9PX, Ireland, brian.fitzgerald@ul.ie

Barbara Lyonnet is an Associate Professor in Management Sciences at University of Nantes and a researcher at LEMNA Laboratory, France. She has published in the field of information system, quality management and supply chain management. She has also written several books including "Lean management". Contact her at University of Nantes, LEMNA, Chemin de la Censive du Tertre, Bâtiment Erdre, 44322 Nantes, France, barbara.lyonnet@univ-nantes.fr

Frantz Rowe is a Professor at University of Nantes, and a researcher at LEMNA and at KTO, SKEMA Business School, France, former CIO of his university and Professor at Telecom Paris; an AIS Fellow an Honorary member of AIM and a member of IFIP 8.2 and 8.6; He is Editor Emeritus of EJIS. He directed the research program on DevOps at LEMNA on which this paper is based. Contact him at University of Nantes, LEMNA, Chemin de la Censive du Tertre, Bâtiment Erdre, 44322 Nantes, France, frantz.rowe@univ-nantes.fr