THE STRUCTURE OF DEVOPS IN PRODUCT-SERVICE SYSTEM DEVELOPMENT

Srinivasan, Rajaram (1); Eppinger, Steven D. (1); Joglekar, Nitin (2)

1: Massachusetts Institute of Technology; 2: Boston University

ABSTRACT

We study a DevOps software development process for a Product-Service System (PSS) using a design structure matrix (DSM) representation. We find unique features such as nested, planned iterations at differing rates which are not evident in conventional engineering development projects. We describe the impact of integrating ongoing operations into a development process and identify some of the enablers that lead to adoption of a DevOps process. We conclude by discussing the implications of our findings and raise questions for further research.

Keywords: Product-Service Systems (PSS), Design process, Agile Development, Process modelling

Contact:
Eppinger, Steven
Massachusetts Institute of Technology
Sloan School of Management
United States of America
eppinger@mit.edu

Cite this article: Srinivasan, R., Eppinger, S.D., Joglekar, N. (2019) ‘The Structure of Devops in Product-Service System Development’, in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.318

1 INTRODUCTION

Development of most software-based products and systems takes place using methods very different from creation of hardware and other engineered systems. We believe there is much we can learn from modern software development techniques that may allow other engineering design processes to become faster, be more flexible, and reduce planning burden. We can attempt to build generalizable lessons from wide-ranging practices including spiral design, evolutionary release, lean start-up, agile development, scrum teams, periodic sprints, and others perhaps yet on the horizon. This paper specifically explores a recent software development practice known as DevOps in order to identify how it works and how we may be able to usefully apply similar methods in new domains.

A traditional approach to software development, popularly known as the waterfall method, involves sequential phases of planning, conceptual design, architectural design, coding, testing, and launch. This staged development approach is also familiar to engineers in other technical fields. Indeed, it works quite well for many development projects where requirements can be specified in the early phases and implemented through the later ones. However, in many software projects, use cases are initially unclear, scope may grow over the project duration, and thus requirements continue to evolve during development. Agile software development methods were developed to address the frustration of evolving and uncertain requirements through a deliberately iterative process of incremental specification, design, coding, integration, and testing (Agile Alliance, 2001).

The recent development of cloud-based software infrastructure has enabled a further refinement of agile, known as the DevOps process. In addition to the rapid, incremental iterations characteristic of agile, DevOps couples software development directly with automated testing and ongoing support operations to enable evolutionary delivery of frequent releases. In some settings, DevOps has reportedly enabled teams to deploy updates to operating software as rapidly as multiple times in a day (Puppet Labs, 2015). DevOps has quickly become the development and delivery mechanism of choice for cloud-based software systems in which software is offered as a service to enterprise customers and distributed users. In this paper, we explore how DevOps enables software teams to manage their work and deliver frequent feature enhancements. Whereas cloud-based software is a type of product-service system (PSS), we aim to learn how DevOps methods may be applied in other realms of engineering and PSS operations.

After a brief review of related literature, we describe a field study of software development using a DevOps process. We use the design structure matrix (DSM) method to document the DevOps process. We then analyze the DSM model to explore the structure of information flows and highlight unique characteristics in comparison to conventional product development processes. Finally, we discuss our observations and how the lessons learned may apply to PSS development in other engineering domains.

2 LITERATURE REVIEW

Conventional development follows a sequentially staged process with review gates to assure quality at the end of each phase before proceeding to the next (Cooper, 2008). For more complex systems, a system architecture phase defines sub-systems that are subsequently developed in parallel (Ulrich *et al.*, 2019). Many of our own earlier studies have explored the nature of iterations within engineering design processes (Smith and Eppinger, 1997; Unger and Eppinger, 2009; Eppinger and Browning, 2012).

Agile methods were developed by the software community as an alternative to the waterfall approach to software development process which reflected that of hardware development (Feldhusen *et al.*, 2009). The Agile Manifesto laid the philosophical foundation for agile development. Agile principles welcome changing requirements even late in the development process and focus on delivering working software in small increments (Agile Manifesto, 2001). The method is widely popular for the flexibility it affords and for its strong belief that motivated individuals in self-organized teams can serve the best interests of customers even without a detailed plan for doing so. Agile is credited with increases of productivity in many software development organizations (Ahmed *et al.*, 2010).

https://doi.org/10.1017/dsi.2019.318 Published online by Cambridge University Press

Scrum is a specific implementation of agile which has also been widely used for software product development. The scrum framework consists of several roles, events, artifacts, and rules (Schwaber and Sutherland, 2017). These include a product owner, scrum master, scrum team, product backlog, periodic sprint, daily scrum meeting, sprint deliverables, customer testing, and kanban boards. After many years of refinement by software teams, there is wide interest in applying scrum in other types of development projects and professional work.

Scaled agile (known as SAFe) is a more recent approach that includes techniques for implementing agile development in large-scale software systems. It provides methods for work at the portfolio, program, and team levels (Leffingwell *et al.*, 2013). Taking agile a step further, Allspaw and Hammond (2009) first described DevOps in a conference presentation titled "10 Deploys per Day: Dev and Ops Cooperation at Flickr". The focus of DevOps is to have engineers not only develop code, but also support the entire pipeline from concept to ongoing usage. To facilitate this process, many levels of automation have been developed. For example, to accelerate the test and launch processes, a protocol called continuous integration and continuous deployment (CI/CD) has been established for validation and release of software as a service (SAAS) products. DevOps and CI/CD have enabled teams to frequently and reliably deliver code changes and to release updates (sometimes) daily to end customers.

This work is related to the literature on the design of PSS (Sakao and Lindahl, 2009). Much of the research into PSS addresses methodologies for cohesive development, operation, and lifecycle management of products as service systems. In many instances, PSS designs are aimed at reducing environmental impact. If the concept of DevOps may be applied to PSS to leverage agile methodologies, we may expand the techniques available for PSS design and lifecycle management by integrating development and operations tasks.

There is an emergent body of literature comparing and combining conventional and agile methods of development. Agile and scrum have been integrated into design thinking (Grashiller *et al.*, 2017) by showing how information exchanges enable symbiosis of creative and agile processes and how such activities may be interleaved at optimal time slots. Böhmer *et al*. (2017) discuss the role of prototype and test cycles in agile development of mechatronic systems. Cooper and Sommer (2016) explain how agile methods may be utilized within traditional staged development processes. Anderson *et al*. (2017) studied the impact of increasing release frequency and argue that if the feedback is noisy, very rapid cycles can lead to longer cycle times and significant rework. However, we are yet to find literature that either maps or assesses either DevOps or PSS development in a similar manner. We offer such an assessment by mapping a DevOps process using the DSM representation. In doing so, we look at the efficacy of information exchanges across different steps in DevOps and allied PSS delivery processes.

3 DSM MODELING OF SOFTWARE DEVELOPMENT

Design structure matrix is a network modeling tool depicting the elements of a system and their interactions. DSM is widely used to represent the structure of engineering processes and of complex technical systems (Eppinger and Browning, 2012). The DSM consists of a N x N square matrix, mapping the interactions among the set of N system elements. In the current study, we utilize a process architecture DSM in which the elements are the activities of the process and the interactions are the flows of work or information between the activities. The process architecture DSM model therefore describes the way activities work together to deliver the process results. This type of DSM analysis highlights important patterns of work flow and their implications for process characteristics such as iterative cycles and completion time. A stylized DSM depicting a conventional (waterfall) software development process is shown in Figure 1.

The DSM in Figure 1 captures – at an abstract level– a stylized process for conventional software development. This DSM representation can be enriched in many ways, for example, by characterizing interaction types using different blocks, and by using various labels or shading to more completely describe the process. Process activities are generally sequenced to represent the overall process flow. Grouped processes may represent coupled iterations or the boundaries of a phase or stage of the process. The DSM graphic is intended to convey to process owners the relevant activity groups and their interactions at a level of abstraction where it can be used for project management and/or process improvement.

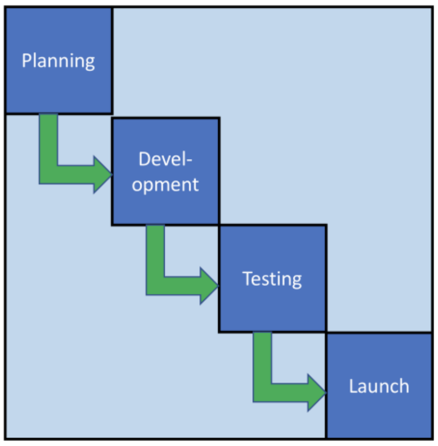

Figure 1. Stylized DSM of a conventional software development process

The DevOps process is characterized by numerous iteration cycles, feedback loops, and incremental releases over time, making its DSM representation somewhat different from the staged development processes with which we are more familiar. The DevOps DSM may represent one or more complete product release cycles, known in agile terminology as a "product increment" (Leffingwell *et al.*, 2013). The product increment (PI) is a defined period of time during which a team delivers incremental value in the form of problem solving and testing software and systems. We can represent various iterative cycles (sprint, release, and planning) by shading regions of the DSM as separate blocks to highlight the differing periodicities with which the process occurs.

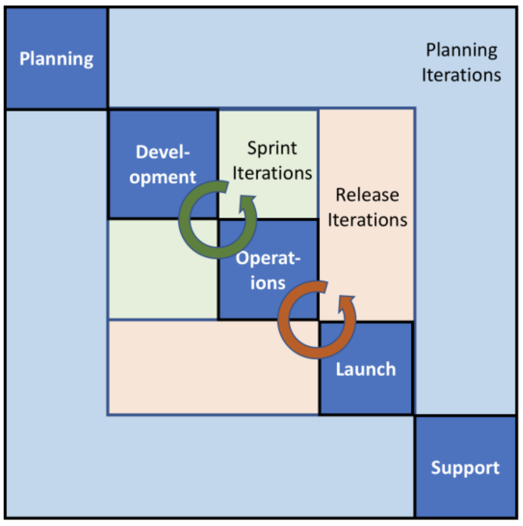

Figure 2. Stylized DSM of a DevOps process

In the stylized DSM of a DevOps software development process shown in Figure 2, we represent groups of planning, development, operations, launch, and support activities. The planning, release and sprint iterations may occur at different rates. All of these iterations contribute to the development and support of a product or a service increment. By nesting the loops as shown in the stylized DSM, we consider that it may take more than one iteration of an inner loop before the outer loop continues its execution. For example, a release iteration may comprise multiple sprint iterations; and similarly, multiple release iterations may take place before a planning iteration. The number of iterations would depend on how much progress is made in each iteration. One particular product increment may release after two sprints, while another may take four. Consistent with agile principles, a feature may be released when it is ready, rather than waiting for a specific launch cycle. Since we have not seen such a DSM in the literature to date, our field study attempts to document the structure and underlying information exchanges of a DevOps process in detail.

4 FIELD STUDY

In collaboration with Swisscom AG, we studied the DevOps process utilized in development and support of the Swisscom TV product-service system. This internet-based platform involves unique hardware, software, and network designs, developed by the Swisscom TV team and several suppliers. Importantly, it is a PSS in which DevOps plays a central role in terms of system development and service delivery.

4.1 Swisscom TV as a Product-Service system

Swisscom AG is a leading telecommunications provider in Switzerland. Swisscom holds a market share of 60% for mobile, 67% for broadband internet, and 35% for TV telecommunication in its domestic residential and commercial markets. Swisscom is known for its premium quality offerings, which command a premium price. Swisscom TV launched its first set-top-box product in 2014 by building its own internet-based TV product after combining several TV-related product offerings. Since 2014, the TV product has evolved significantly, including launch in 2016 of a new set-top-box with ultra-high-definition video and voice search functions.

In terms of a PSS, the set-top-box in a customer's home is only a means to consume an expanding range of services on an ongoing basis. Hence, the TV team and its DevOps process must support ongoing operations at thousands of customer sites in thousands of configurations of hardware, software, and firmware. The product elements of the PSS include the set-top-box hardware; service elements include offerings such as live TV, video on demand, recording, program guides, third-party applications, and other value-added services. All of these services are consumed over the internet, making the offering a comprehensive internet TV service. Efficient operations and support are therefore critical to customer satisfaction. Hence, the system boundaries of the product offering spans beyond just Swisscom TV, to other service providers with which the team partners to provide an integrated customer experience.

4.2 Data collection and DSM modeling

We conducted the field study through multiple visits to Swisscom in Zurich. Data collection consisted of over 20 interviews with the TV team, generally lasting 40 to 60 minutes each. Subjects included experience designers, program managers, DevOps engineers, software architects, and senior leadership including chief architect, design head, and product line executives. Interviews covered Swisscom's journey into internet-based video and entertainment, current engineering and operational processes, as well as key technical and management challenges.

These data were used to construct the DSM which was then analyzed for insights. Initially identified relationships were validated through a second round of discussions with the leadership team. We present a summary of the results here using the DSM interface labeling convention shown in Figure 3. We use nomenclature described below to present key aspects of the DSM structure shown in Figure 4.

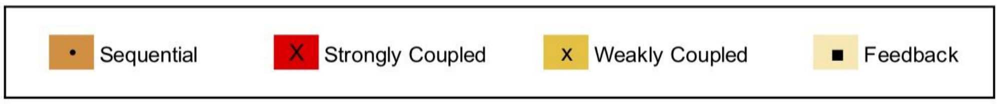

Figure 3. Labels for different types of interactions among activities

4.2.1 Sequential activities

Two activities are considered sequential if the output of the upstream activity enables execution of the downstream activity (Eppinger and Browning, 2012). For example, #15 Application Backend Development and #20 Integration Testing are sequential since the development needs to be completed before integration testing begins. Some sequential activities may be partially overlapped by starting the downstream activity before the upstream activity is completed. However, for modeling purposes we are not interested in the precise timing, so we omit any distinction of overlapped sequential activities.

https://doi.org/10.1017/dsi.2019.318 Published online by Cambridge University Press

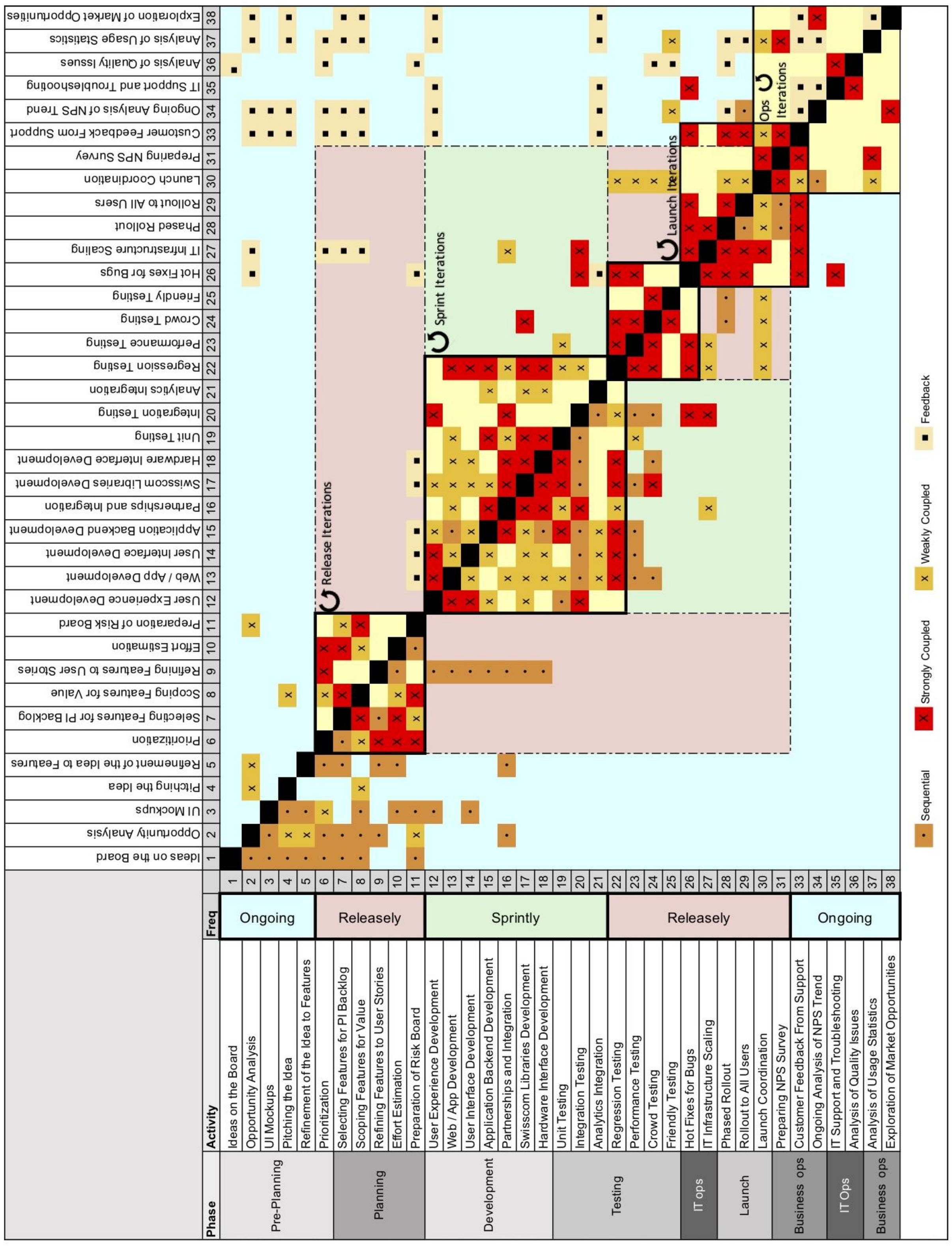

Figure 4. The DSM model for swisscom TV DevOps process

4.2.2 *Strongly coupled activities*

Strongly coupled activity pairs need ongoing input from each other and will iterate until converging on a mutually satisfying result. Such coupled activities typically take place simultaneously, with the teams interacting regularly to resolve uncertainties and trade-offs. For example, #12 User Experience Development is strongly coupled with #14 User Interface Development. This is because the user

https://doi.org/10.1017/dsi.2019.318 Published online by Cambridge University Press

interface team works closely with the user experience team over multiple iterations to ensure that the expected user experience has been adequately delivered in the designed interface.

4.2.3 Weakly coupled activities

Weakly coupled activity pairs also need input from each other, but much less frequently than strongly coupled pairs. This allows some elements of the two activities to occur independently, but they cannot be completed without some coordination. The weak coupling could also be addressed using detailed interface specifications or facilitated by well-defined procedures. For example, #19 Unit Testing and #23 Performance Testing are weakly coupled, as once a test plan is in place, the two processes need not share much information and hence can proceed in parallel.

4.2.4 Feedback

Feedback within the process is indicated by special marks above the diagonal. Feedback consists of information being passed to an earlier activity which has generally completed its current work and will use the new information to execute its next iteration. Feedback can occur even without strong interactions between the process owners. For example, a sprint retrospective meeting could highlight challenges faced in the completed sprint and the learnings from it used for the next sprint.

4.3 The DevOps DSM model

The resulting DSM, developed after iterative analysis is presented in Figure 4. Blocks of coupled activities largely coincide with stages of the PSS development process described by the far-left column of the DSM, spanning planning through development, test, launch, and ongoing operations. One of the challenges we faced with this particular DSM model representation is that the various DevOps iterations take place at different rates. Identification of three separate blocks of concurrent work helped in visualizing these different types of iteration, as discussed in further detail below.

5 RESULTS

5.1 Unique features of information exchange in DevOps

5.1.1 Interactions and multiple speeds of iteration

Within each of the high-level process stages, planned iterations occur at different speeds and typically there are multiple iterations before handing off results to the subsequent activities. For example, multiple sprint iterations take place before a release. Thus, certain development and testing activities. such as #19 Unit Testing and #20 Integration Testing, happen multiple times over the period of a two-week sprint. We designate these “sprintly iterations.” In the next phase, iterations involving #23 Performance Testing and #25 Friendly Testing take place about once every six weeks. We term these “releasely iterations.” The third pace of work flow in the DSM is that of the “ongoing iterations.” When two activities belong to regions of different speed, the interactions between the two activities occur at the faster of the two speeds. For example, the interactions between #17 Swisscom Libraries Development and #24 Crowd Testing happen every sprint, although crowd testing happens only once every release. This is because the hardware libraries are tweaked based on performance tests reported by crowd testing. Sprintly, releasely and ongoing iteration blocks, representing different rates of process flow and iteration, illustrate how the DevOps process integrates the ongoing planning and operations phases with the sprints and releases during highly iterative development and testing phases.

5.1.2 Time boxing of ongoing processes

Most activities in the pre-planning and operations phases are ongoing processes. This is because activities such as #2 Opportunity Analysis and #35 IT Support and Troubleshooting occur irrespective of release cycles. Often, organizations follow different project execution methods such as kanban for IT operations while using scrum for product development, and this may be due to the ongoing nature of supporting processes (Viszlai, 2015). However, Swisscom TV integrates scrum with ongoing IT and business operations through “time boxing.” In this usage, time boxing refers to a project management technique wherein key activities have specific, predetermined duration and/or end dates. For example, while the business team scouts for opportunities on an ongoing basis, at specific time intervals (perhaps every six months), the team pauses to prioritize opportunities to be considered for the next product increment. Similar time boxing of activities is also carried out for bug fixes and IT operations. Time boxing also provides flexibility to add new operations or support priorities if they are deemed critical. This pause-and-prioritize approach helps to keep the team on track with dynamic priorities.

5.1.3 Levels of feedback

Square marks (■) in the DSM represent a type of feedback known as "generational learning." Feedback from operational activities such as #37 Analysis of Usage Statistics to #28 Phased Rollout are operations feedbacks to the next generation's launch based on knowledge from ongoing operations. Operations provides data into the process at multiple levels, such as launch, test and even all the way back to planning. For example, #36 Analysis of Quality Issues feeds data into #6 Prioritization. IT operations processes such as #27 Infrastructure Scaling drive decisions for planning processes such as #6 Prioritization, #7 Selecting Features for PI Backlog, and #8 Scoping Features for Value by answering key questions such as availability of relevant infrastructure to accommodate adoption of a new feature. For example, features such as recorded TV require massive amounts of storage since the Swisscom TV service uses only cloud storage rather than in the set-top box. These feedbacks often constitute passive interactions in which new information can be captured using a robust project management system and the teams need not meet about every issue as it arises. For example, Swisscom TV team goes through a product increment planning meeting once each release. Most of the planning activities happen in one day with all relevant stakeholders in the same room. This could be an opportunity to share all the relevant feedback to the upcoming product increment. At the other extreme, some of the coupled or sequential activities must be linked directly to the subsequent steps of the process, such as #25 Friendly Testing and #28 Phased Rollout. Friendly testing is contingent upon the phased rollout taking place.

5.2 Enablers for adoption of DevOps

The DevOps DSM maps the information process flow and highlights the different speeds of iterations. In order to gain additional insights, we showed the DSM maps to relevant stakeholders, and then asked them about key enablers for adoption and implementation of a DevOps process. We group these enablers into three categories: organizational enablers, product enablers, and process enablers and metrics.

5.2.1 Organizational enablers

It can be seen from the iterations in the launch and operations iterations phases of the DSM that business and IT operations teams share many interactions. These interactions are more pronounced in PSS where operations enable value creation for customers. This is illustrated by activities such as #30 Launch Coordination, wherein the TV team coordinates with marketing and retail stores for communication of the new value proposition. IT operations processes such as #27 IT Infrastructure Scaling and #35 IT Support and Troubleshooting ensure high availability and timely cross-organizational communication of issues to relevant development teams. For instance, each of the coupled activities between launch and test or development phases illustrates the need for establishing organizational ties between development teams and IT and business operations teams. Product managers act as business representatives in a scrum team to facilitate this communication.

Ongoing operations often provide flexibility in deployment strategies and the ability of a scrum team to carry out phased rollouts. Activity #25 Friendly Testing uses company employees as a test bed to identify issues with usage in real-world environments. This feeds data towards a phased rollout for real customers. It often takes several weeks before a feature is fully deployed to all users. This multi-step workflow must not only account for organizational boundaries, but also coordinate varying team priorities since different teams conduct each level of testing (unit, regression, and integration, and friendly testing).

In DevOps teams, developers also run a portion of operations. While the IT operations and infrastructure teams focus on issues at a macro level, such as scaling, developers are expected to write code that is configuration agnostic and has established interfaces to other modules in the system. Advancements in cloud computing and allied infrastructure (e.g., containers and orchestration systems

https://doi.org/10.1017/dsi.2019.318 Published online by Cambridge University Press

such as Docker and Kubernetes) accelerate this trend further. That is, updating processes to take advantage of infrastructure helps teams in cutting down cycle time. This requires that developers work closely with the IT operations and business operations teams to understand their priorities while setting up time-boxing deadlines.

5.2.2 Product enablers

An incremental product deployment strategy plays well with the plan-do-check-act (PDCA) learning cycle embedded in scaled agile (Leffingwell *et al*., 2013). It helps companies to build a product feature quickly, test it in the market, assess customer response, and continue iterating further if it gathers interest or pivot to a different feature if it doesn't. Moreover, a team's focus on building a testable PSS at the end of each sprint gives the team a bias for action. Both of these factors must be considered while planning and scoping features.

Some product enablers, typically those associated with customer response, involve the ability to thoroughly measure and understand customers' interests and pain points. Swisscom TV has processes to objectively analyze customer support requests. It also has data collection instruments and analytics built into the product. For instance, some features are designed to understand system performance and crash reports. Such product enablers are accompanied by periodic net-promoter score (NPS) surveys and subjective surveys for direct customer feedback. Trends are analyzed over time to isolate patterns and correlate them with events to quantify drivers of customer satisfaction.

5.2.3 Process enablers and metrics

DevOps relies upon development teams working closely with many other teams. Nevertheless, for various reasons such as talent availability and localized needs for end customers, it may not be possible for all teams to be co-located. Consequently, slow response times may be experienced for both planned iterations and handling of emerging concerns. One way to address the resulting bottleneck is to design a process with planning meetings where all relevant stakeholders are present. Swisscom TV team holds a product increment planning meeting for every new product iteration wherein all details of each feature to be developed are fleshed out. This reduces the scope of some iterations, especially late in the product increment cycle. Such meetings are particularly useful in bringing the operations teams' feedback directly into the overall planning cycle. Necessary decision makers are also present in this meeting and a clear consensus is reached on the commitment for the upcoming product increment.

With a large product development process involving multiple stakeholders, establishing responsible process owners, deliverables, and metrics for all processes can be helpful. Table 1 offers examples of possible metrics-based management strategies at each cycle speed.

Table 1. Establishing process ownership and deliverables

Cycle Speed	Process	Owner	Deliverable	Metrics
Sprintly	Web development	Web product owner	Captured in Jira	Story points
Releasely	Launch coordination	Product manager	Training materials	Outlets reached
Ongoing	Opportunity analysis	Business development manager	Opportunity presentation	Potential revenue

6 CONCLUSION

We have mapped DevOps activities using DSM in order to identify unique features of the process in one implementation at Swisscom. Key takeaways from this study include:

1. We illustrate how to map a multi-speed DevOps process using a DSM representation by introducing a novel iteration block identification and labeling scheme.
2. We identify unique features of a case study process, including iterations occurring at multiple speeds, time boxing of both iterative cycles and ongoing activities, and multiple levels of feedback enabling teams to learn from development sprints and ongoing operations.
3. Analysis of this PSS development and support process has allowed us to identify a number of enablers for adoption of a DevOps process.

While some of the learning from Swisscom's TV team might be translatable to other software systems that have a need to change the product offering on the fly, this study also raises a host of questions that motivate future research. For instance, will this process architecture work in other cloud-based PSS systems, such as banking or episodic services such as emergency response support systems? What kind of development and test cycle speeds, metrics and time boxing would be needed in such settings? Could these approaches be used for developing more conventional PSS such as electric utilities or vehicle leasing services? Finally, there ought to be careful empirical examinations of the potential productivity gains and adoption enablers we have described.

REFERENCES

Ahmed A., Ahmad S., Ehsan N., Mizra E. and Sarwar S.Z. (2010), "Agile Software Development: Impact on Productivity and Quality", In: *IEEE International Conference on Management of Innovation & Technology*, Singapore. https://dx.doi/org/10.1109/ICMIT.2010.5492703

Agile Alliance (2001), *What is Agile Software Development* [online], www.agilealliance.org/agile101 (Nov 2018).

Agile Manifesto (2001), *Manifesto for Agile Software Development* [online], www.agilealliance.org/agile101/the-agile-manifesto (Nov 2018).

Allspaw, J. and Hammond, P. (2009), *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr*", O'Reilly Velocity Conference, San Jose.

Anderson, E., Lim, S.Y. and Joglekar, N. (2017), "Are More Frequent Releases Always Better? Dynamics of Pivoting, Scaling, and the Minimum Viable Product", In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. https://dx.doi/org/10.24251/HICSS.2017.705

Böhmer, A.I., Hostettler, R., Richter, C., Lindemann, U., Conradt, J. and Knoll, A. (2017), "Towards Agile Product Development: The Role of Prototyping", In: *Proceedings of the 21st International Conference on Engineering Design (ICED17)*, Vancouver.

Cooper, R.G. (1990), "Stage-gate systems: A new tool for managing new products", *Business Horizons*, Vol. 33 No. 3, pp. 44–54.

Cooper, R.G. and Sommer, A.F. (2016), "From experience: The agile–stage-gate hybrid model: A promising new approach and a new research opportunity", *Journal of Product Innovation Management*, Vol. 33 No. 5, pp. 513–526.

Eppinger, S.D. and Browning, T.R. (2012), *Design Structure Matrix Methods and Applications*, MIT Press, Cambridge.

Feldhusen, J., Löwer, M. and Bungert, F. (2009), "Agile Methods for Design to Customer", In: *Proceedings of the 17th International Conference on Engineering Design*, Palo Alto.

Grashiller, M., Luedeke, T. and Vielhaber, M. (2017), "Integrated Approach to the Agile Development with Design Thinking in an Industrial Environment", In: *Proceedings of the 21st International Conference on Engineering Design (ICED 17)*, Vancouver.

Leffingwell, D., Yakyma, A., Jemilo, D. and Knaster, R. (2013), *Scaled Agile Framework* [online], scaledagileframework.com (Nov 2018).

Puppet Labs. (2015), *State of DevOps Report*, Puppet Labs [online], puppet.com/resources/whitepaper/2015-state-devops-report (Nov 2018).

Sakao, T. and Lindahl, M. (Eds.) (2009), *Introduction to Product/Service-System Design*, Springer Science & Business Media.

Schwaber, K. and Sutherland, J. (2017), *The Scrum Guide: The Definitive Guide to Scrum - The Rules of the Game*, Scrum Inc. [online], scrumguides.org (Nov 2018).

Smith, R.P. and Eppinger, S.D. (1997), "A predictive model of sequential iteration in engineering design", *Management Science*, Vol. 43 No. 8, pp. 1104–1120.

Ulrich, K.T., Eppinger, S.D. and Yang, M.C. (2019), *Product Design and Development*, 7th edition, McGraw-Hill, New York.

Unger, D.W. and Eppinger, S.D. (2009), "Comparing product development processes and managing risk", *International Journal of Product Development*, Vol. 8 No. 4, pp. 382–402.

Viszlai, L. (2015), *Introducing Kanban to IT Operations*, VMware Blogs [online], blogs.vmware.com/accelerate/2015/11/kanban-it-operations.html (Nov 2018).