21st INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2019
MONTEREY, CA, USA, 23 – 25 September, 2019

The Structure of Agile Development Under Scaled Planning and Coordination

Siddharth Bajpai [1], Steven D. Eppinger [1], Nitin R. Joglekar [2]

[1] Massachusetts Institute of Technology, Cambridge, MA, USA
[2] Boston University, Boston, MA, USA

Abstract: Agile and Scaled Agile Framework (SAFe) methodologies are increasingly being deployed, both for software development and also by teams in broader work settings. This is a proof-of-concept study to develop a DSM for a SAFe development environment. Key takeaways from this study are: (i) Agile and SAFe DSMs are characterized by a unique set of nested information dependencies that are absent in waterfall development; (ii) these dependencies are not readily evident in the conventional and layered rendering of SAFe process documentation. We discuss the organizational and analytical process improvement opportunities available based on such DSM analysis and conclude by identifying possibilities for follow-on research.

Keywords: Agile, SAFe, Software Development Process, DSM

1 Agile Development Dependencies

Agile development was envisioned to create expedient processes for incorporating dynamic customer input into iterative and incremental software development (Agile Manifesto, 2001). In practice, various implementations of agile processes set up adaptive planning and evolutionary development while encouraging flexible response through rapid product releases to gather frequent customer feedback on desirable product features. An agile team then creates, delivers and updates a prioritized list of features needed by customers. Implementations incorporate concepts such as sprint and scrum. A sprint is a repeated design-code-test-release software development cycle lasting from one week to one month. The term scrum has been drawn from rugby scrummages, because development effort involves a cross-functional team. Scrum teams huddle daily to assign work to individual team members from a prioritized list of features and other tasks known as the backlog (Schwaber, 2009). Teams work to reduce this backlog through sprint cycles, while aiming to release working software, at the completion of each sprint.

In part, the agile revolution in the software industry was a reaction to the (sometimes) excessive planning and control typical of large-scale waterfall software development methods. However, scaling agile methods from a single team to multiple, interacting teams, and from small software applications to larger systems, requires implementation of some degree of planning and coordination. Leffingwell (2008) introduced the Scaled Agile Framework (SAFe) to organize such planning and coordination mechanisms. Agile and SAFe methodologies are now being used, beyond software development, by hardware product developers and by teams in broader work settings (Cooper, 2016; McKinsey Survey, 2017).

Structured as a series of periodic sprints, agile development work is fundamentally iterative in nature. The informal organization of sprint work creates activity dependencies within each sprint team. For instance, daily scrum meetings during a sprint cycle require updates from team members on the status of their assigned work along with emergent issues that might have come up during the previous day's work. In scaled agile, coordination across teams also involves interface definition across sub-systems and coordinated backlog prioritization required to deliver coordinated work results (Thompson, 2019).

This paper seeks to demonstrate how DSM can be applied to agile software development processes and specifically how a SAFe process can be represented using DSM. We also consider how DSM analysis can help to improve an existing agile process. Finally, we ask what we can learn from SAFe for improving other (non-software) projects.

2 Modeling Software Development Processes

Design structure matrix is a network modeling tool for 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.

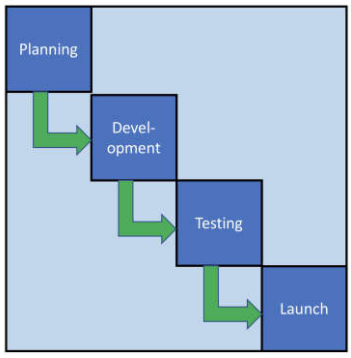

Figure 1. Stylized DSM of a Waterfall Software Ddevelopment Process
(Source: Srinivasan et al., 2019)

The DSM in Figure 1 captures – at an abstract level – a stylized process for conventional (a.k.a. waterfall) software development. This DSM representation can be enriched in many ways, for example, by characterizing interaction types using different marks, 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 planning, process control, and/or process improvement. For a detailed example of mapping such software development, and allied coordination and problem solving tradeoffs, see Gomes and Joglekar (2008).

Mapping of agile tasks into a DSM model is an emergent research theme. Srinivasan et al. (2019) examine the issues associated with modeling a special type of agile operation, termed DevOps, which integrates agile development activties with downstream operations of released software. Such a process is particularly relevant in the Product-Service System (PSS) development context.

We present a stylized DSM diagram, shown in Figure 2, depicting the work of a single agile team engaged in repeated sprints. This DSM captures – at an abstract level– the process of product planning following by an iterative sprint block within which scrum activities include daily meetings, development, and testing. Software is released at the end of the sprint and customer feedback is delivered for ongoing planning of future releases.

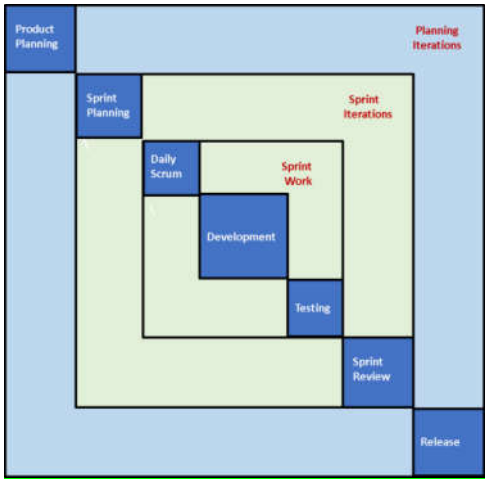

Figure 2. Stylized DSM of Agile Software Development for a Single Team

Modern software development following an agile approach, even with moderate complexity, involves multiple sprint teams working in parallel to address a portfolio of customer needs. Alternative frameworks (e.g., SAFe, SOS – scrum of scrums, DAD – disciplined agile delivery and Nexus) may be used for coordination across multiple teams. Ebert and Paasivaara (2017) review five frameworks and offer a comparison matrix.

SAFe is arguably the most prevalent framework for coordination of complex software development. It calls for scaling the agile development process with multiple layers (e.g., portfolio, program, and team layers), as shown in Figure 3. At the portfolio level, work is described in terms of themes and epics (a.k.a. major milestones). At the program level, work is planned as a sequence of Product Increments (PIs) developed by agile release trains (ARTs). PIs are time-boxes during which a specified set of product improvements are released to achieve an expected incremental value to the product. An ART is a self-

organizing, virtual organization comprised of 5 to 12 teams that plans, commits, and executes these PIs.

Another key construct at the SAFe program level is the "Innovation & Planning" sprint (I&P), which serves multiple functions including preparing for the next PI Planning event, local innovation, and Inspect and Adapt workshops where recently completed work is showcased to business teams. At the individual team level of SAFe, work is characterized by sprints and scrums. The aim behind Figure 3 is to depict the layered structure of SAFe.

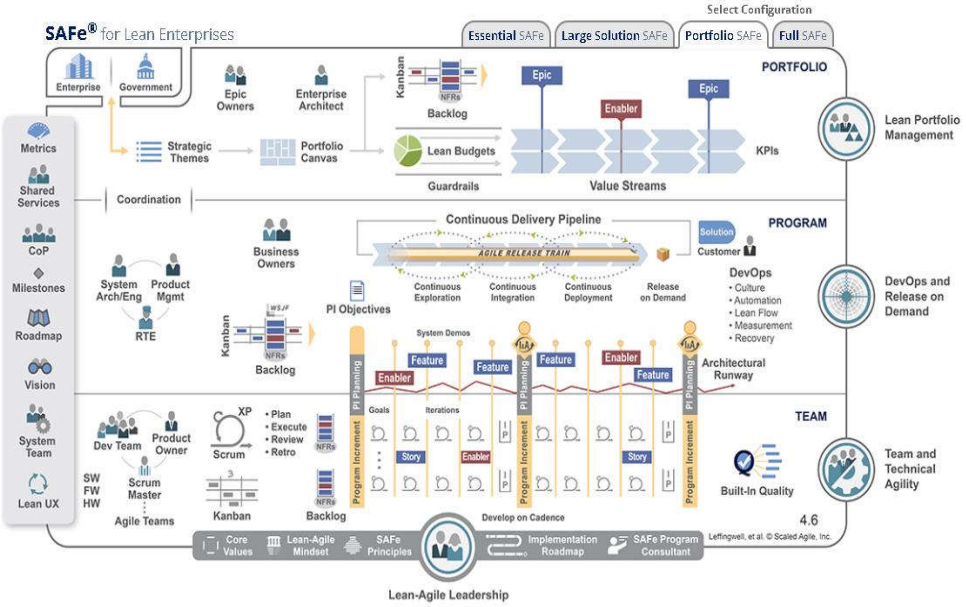

Figure 3: A Portfolio SAFe Development Process (Source: Scaled Agile, 2019)

Since the goal of this study is to examine SAFe using the lens of DSM analysis, we lay out the SAFe process as a stylized DSM in Figure 4. This process begins with a block of Portfolio Planning activities, followed by a block of Release Train Planning, followed by a block of Innovation of Planning (I&P) activities, and further by another set of blocks for Sprint Planning, Scrum, Development, Testing, and Release activities. It is worth noting that in this depiction, there are typically several teams running sprints concurrently, but only one instance is shown here for ease of depiction. It is also normal in the SAFe environment to model the ongoing activities (termed as DevOps in Figure 3). We depict ongoing operations (Ops) as a separate block at the bottom of this stylized DSM. Also represented are several possible sets of dependencies across activities that are organied either sequentiually, and/or within concurrent activity blocks.. In Figure 4, we highlight these blocks and feedback dependencies across these blocks. Moreover, it is also possible, indeed quite normal, for these activity blocks to be run at different speeds. (Srinivasan et al., 2019)

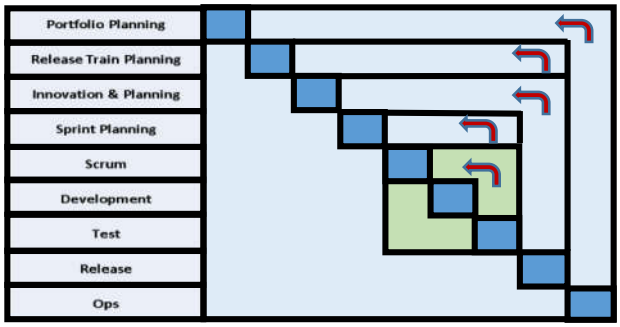

Figure 4. Stylized DSM of a SAFe Software Development Process

Ebert and Paasivaara (2017) have considered how SAFe can improve visibility and alignment across the organization. They point out that (p. 102), while software engineering literature recognizes the need to manage such complexity, issues such as dependencies and their optimal coordination have not been well addressed. We have designed a case based field study to address this gap in the literature. In this study, we map the overall SAFe process and individual sprints using DSM. We aim to assess whether analysis of such maps can provide agile teams with unique insights to improve teamwork and SAFe coordination decisions.

3 Field Study

We studied the SAFe process for development and support of the Swisscom Big Data Platform (SBD for short). This is a data storage and computation platform which is used by several internal software applications and technology or business teams to build data and analytics solutions. SBD's agile development process is part of the Data Lake Agile Release Train, within Swisscom's Data, Analytics and AI (DNA for short) Large Solution.

3.1 Swisscom Big Data as an Internal Technology Product-Platform

Swisscom AG is a leading telecommunications provider in Switzerland. Swisscom holds a market share of 59% for mobile, 68% 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's Data Lake system is a large, centralized, unstructured data repository for all of Swisscom's data. Swisscom Big Data Platform (SBD) serves as technical infrastructure used by several other engineering teams within Swisscom to build data visualization, analytics, and AI solutions for internal customers. The SBD platform is specialized for ingestion of real-time data streams, and provides a source-agnostic way for data to be passed into the system and stored long-term in an unstructured form. These data are made accessible through an application that makes it structured and queryable. Analytics teams at Swisscom interested in accessing any data available in SBD develop and set up their own pipeline using the tools available through the SBD stack. SBD then maintains this pipeline over time, while evolving the system to make it capable to serve the anticipated future business needs.

3.2 Data Collection and DSM Modeling

We conducted the field study through multiple visits to Swisscom in Bern. Data collection consisted of 12 interviews with the SBD team, generally lasting 40 to 60 minutes each. Subjects included product owners, development team managers, software architects, and senior leadership including IT Architect, Vice President of Engineering for the DNA Large Solution, and Principal Product Owner for Swisscom Big Data (SBD). Interviews covered SBD's adoption of the SAFe development methodology, current engineering and operational processes, and key technical and management pain points in these processes. 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. Using the notation established in Srinivasan et al., (2019), we present a summary of our results using the DSM interface labeling convention shown in terms of Sequential, Strongly Coupled, Weakly Coupled, and Feedback. In this DSM, shown in Figure 5, we also identify the different periods over which these activities take place: Ongoing, PI-ly and Sprintly, along the top of this DSM.

4 Description of SAFe DSM

We have organized this section in terms of a description of concurrent activity blocks, followed by a discussion of nested iterations across these blocks within Figure 5. These concurrent blocks were initially identified by examining a preliminary DSM. The block membership and boundaries were then reviewed by the Swisscom team. Suitable adjustments were made to arrive at the final DSM shown in Figure 5.

4.1 Concurrent Activity Blocks

An ongoing set of activities occur in the Large Solution Planning (LSP) block involving activities such as #1 (Requirement Generation), #2 (Opportunity Assessment) and #10 (Requirement Engineering).

The first seven activities of LSP are deemed strategic and result in the creation of epics (i.e., setting up overall time line) along with the organization of work among ARTs. Activities in this block also provide an assessment of the impact of the architecture and prioritization at the solution level. The LSP block overlaps with the Agile Release Train Planning (ARTP) block. ARTP involves activities such as #11 (Drilldown on Program Capabilities), #12 (Business Value Estimation) and #13 (Breakdown of Capabilities into Features). Armed with a feature list generated (from activity #13) upon conclusion of ARTP, each sprint team in the ART (including those comprising the SBD team) can begin working on a set of preparatory activities leading up to the next Program Increment (PI) planning session, conducted during the upstream I&P sprint of the previous program increment, activities #14 (Feature Prioritization) to #18 (Effort Estimation). Upstream I&P sets up the activity sequence that is followed by four activities in a block called PI & Sprint Planning (PSP Block). Both Upstream I&P and PSP blocks occur at the frequency of once per PI.

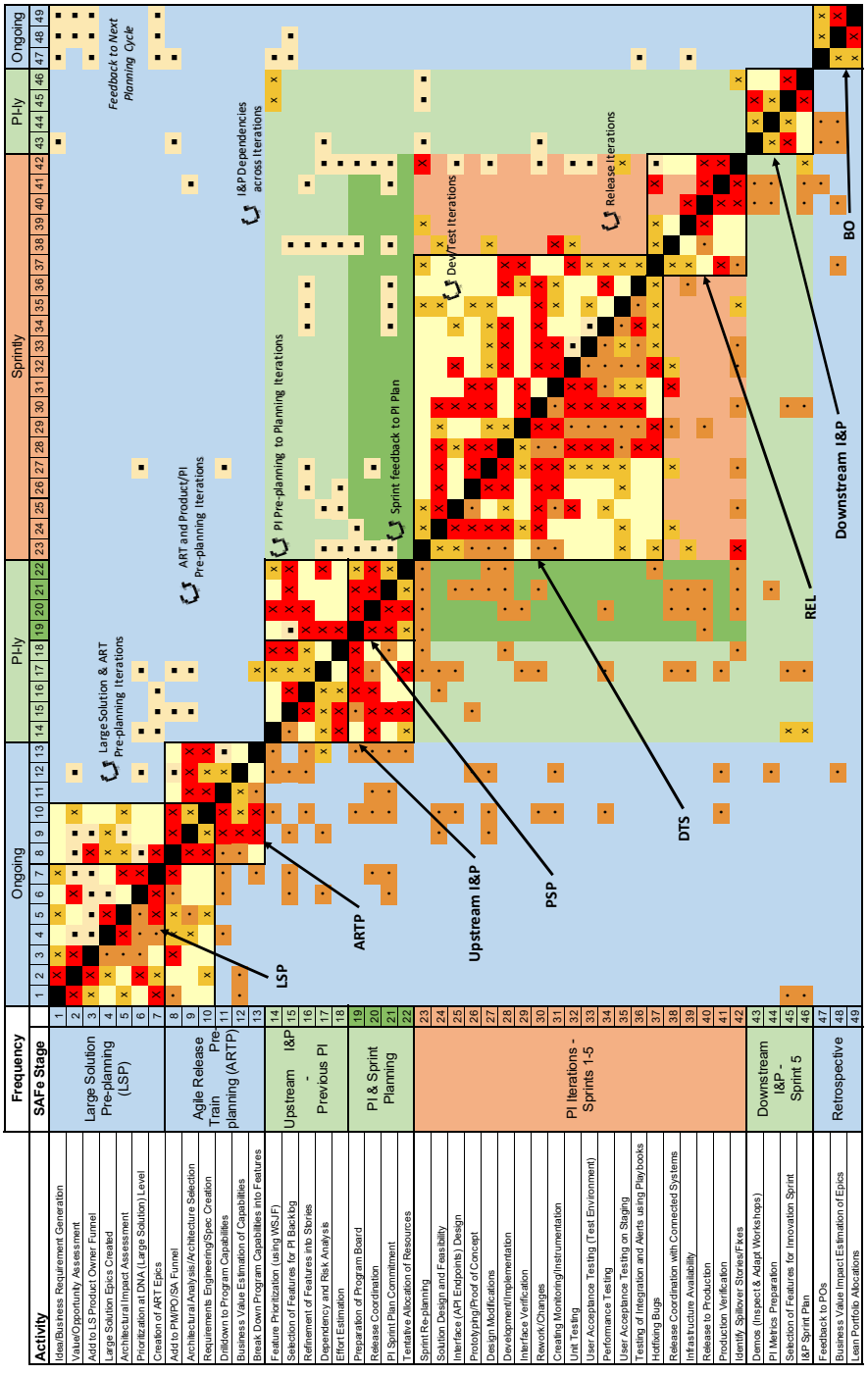

Figure 5. DSM of the Observed SAFe Development Process

Upon conclusion of the PSP, each sprint team can begin Dev/Test iterations in the PI Sprints stage of activities. These iterations involve concurrent Development, Test and Staging (DTS Block) represented in activities #23 to #37, followed by Release (REL Block) represented in activities #37 to #42. These blocks (i.e. DTS and REL) run on the Sprintly time scale. It is not necessary for the full set of activities in these two blocks to take place each sprint, and sometimes iterations can happen within sprints or across sprints. However, all iterations of these activities conclude within Sprints 1 through 5 of the Program Increment (PI), and it is occasionally true that all activities pertaining to a single feature are completed within a sprint.

Release block is followed by Downstream I&P activities (#43 to #48) which round out the current PI, including activities such as #44 (PI Metrics Preparation) and #45 (Selection of Features for Innovation Sprint). These activities occur concurrently during the final Sprint of the PI and thus couple with some activities in the DTS block. The Downstream I&P set of activities occur PI-ly, and are concurrent with the Upstream I&P block for the next PI iteration. Each epic concludes when released work becomes operating software features. Based on customer input to business operations, assessment of the business value impact of the epic and portfolio allocations (activity #47 to #49) can be carried out within the business operations (BO) Block.

4.2 Nested Loops across Concurrent Blocks

Perhaps the most striking feature of the SAFe process we observed is the prevalence of nested loops across the concurrent activity blocks. While there certainly is a nominal flow of work from high-level planning through release, there are multiple paths for planned feedback to update plans at each development pace – both within sprints and PIs, and for planning subsequent ones. Figure 6 summarizes these nested, planned iterations.

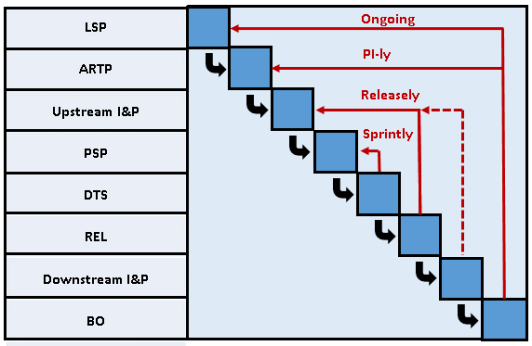

Figure 6. Nested Structure of SAFe Dependencies

5 Discussion

5.1 Organizational Opportunities for Improving Agile and SAFe Processes

The SAFe DSM provides a map for each activity in terms of where input information comes from, which activities use the output, and which other activities create tight

coupling. When laid out in a two-dimensional matrix, such mapping may be easy to depict, but its usefulness in terms of organization of groups of activities, and assigning roles, may not be trivial. For example, in Figure 3, which provides a layered view of the SAFe process as a whole, these dependencies have been suppressed. In the absence of a DSM map, it will be virtually impossible, based on Figure 3, for the planners to tell exactly where the points of communication may lie. It will also be difficult to ascertain how frequently these communications take place and which channels of information exchanges face organizational mismatches (because these exchanges take place at different speeds).

Even though the blocks with tight coupling are easy to observe in a two-dimensional map, it is not clear if the teams within any block would recognize this structure and organize around it, because there is no comparable guidance either in the SAFe or within Agile/Scrum training literature to address these issues. The extent of guidance in this literature is limited to recognizing that dependencies exist, and the scrum masters (responsible for the ARTs) and architects (who coordinate the SAFe) should address them. Arguably, some scrum masters and architects may create resource buffers, and foster additional mismatches, around their own team in the face of informational uncertainty, and thereby compound the coordination problems.

It ought to possible to set up an intervention study by constructing such maps, and then use these maps to inform architects and scrum masters regarding the desirable structures for their blocks, along with plans for formalizing the needed communication practices. Such a study can document the organizational and performance consequences in matched settings: one sprint team with and the other without an intervention.

5.2 Analytical Opportunities for Improving Agile and SAFe Processes

In theory, it is also possible to improve the DSM structure through analytics (Eppinger and Browning, 2012). For instance, in order to improve the within-team information exchanges, one may advocate the use of sequencing algorithms within each block (in particular for LSP and ARTP because of their strategic importance, and also for the Development, Test and Staging (DTS) block because it is exercised repeatedly. This would amount to a local optimization within key blocks.

Global optimization, based on analytical rationalization of the nested dependency structure shown in Figure 6, which operates at multiple speeds, is a difficult problem to solve. While there has been some research on optimizing a DSM with a single periodic update (Yassine et al., 2003), the multi-speed nested problem is yet to be studied. We identify such analytics as opportunities for follow-on research.

5.3 Generalizability

A major limitation of the current study is the need to generalize the lessons from a single case study into a broadly applicable methodology, both for mapping the SAFe DSM and for drawing inferences. In a related study, Srinivasan et al. (2019) have mapped and analyzed a DevOps DSM. Their DSM has major structural differences when compared with Figure 4, because their ART has integrated ops activities. Our cuurent study may be a maiden attempt at mapping a SAFe DSM. Even a cursrory comparison of Figures 4 and 5 shows that Swisscom has a unique way of settting up some of their blocks (for instance the I&P blocks occur both upstream and downstream). Other SAFe implementations may

not follow this pattern. The generalizabity of such a DSM structure in other scaled agile settings, such as large-scale scrum (LeSS), disciplined agile delivery (DAD), and Nexus, is yet to be tested. A related issue is what can be learnt from SAFe DSMs for improving the coordination of other (non-software) projects. We identify such assessments as fruitful avenues for follow-on work.

6 Conclusion

This is a proof-of-concept paper. The goal for the underlying research was to explore if DSM mapping can be used to illustrate the dependency structures for agile and SAFe development environments. Key takeaways from this case study, and allied data analysis are: (i) agile processes are characterized by a unique set of nested information dependencies that are absent in waterfall development; (ii) these dependencies are not readily evident in the conventional and layered rendering of the activities in a SAFe process documentation. We discuss the organizational and analytical process improvement opportunities available based on such a DSM analysis and conclude by identifying opportunities for follow on research.

References

Agile Manifesto, 2001. *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Cooper, R.G., 2016. Agile–Stage-Gate Hybrids: The Next Stage for Product Development Blending Agile and Stage-Gate Methods Can Provide Flexibility, Speed, and Improved Communication in New-Product Development. *Research-Technology Management*, 59(1), 21-29.

Ebert, C., Paasivaara, M., 2017. Scaling Agile. *IEEE Software*, 34(6), 98-103.

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

Gomes, P.J., Joglekar, N.R., 2008. Linking Modularity with Problem Solving and Coordination Efforts. *Managerial and Decision Economics*, 29(5), 443-457.

Leffingwell, D., 2007. *Scaling Software Agility: Best Practices for Large SAFe Enterprises*. Addison-Wesley. ISBN 978-0321458193.

McKinsey Survey, 2017. *How to Create an Agile Organization*. https://www.mckinsey.com/business-functions/organization/our-insights/how-to-create-an-agile-organization

Schwaber, K., 2009. *Agile Project Management with Scrum.* O'Reilly Media, Inc. ISBN 9780735637900.

Scaled Agile, 2019. https://www.scaledagileframework.com

Srinivasan, R., Eppinger, S.D., Joglekar N.R., 2019. The Structure of DevOps in Product-Service System Development. *Proceedings of the International Conference on Engineering Design* (ICED19), Delft, The Netherlands.

Thompson, K.W., 2019. *Solutions for Agile Governance in the Enterprise (Sage).* Sophont Press.

Yassine, A., Joglekar, N., Braha, D., Eppinger, S., Whitney, D., 2003. Information Hiding in Product Development: The Design Churn Effect. *Research in Engineering Design*, 14(3), 145-161.

Contact: Steven D. Eppinger, Massachusetts Institute of Technology, eppinger@mit.edu.