22nd INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2020
CAMBRIDGE, MASSACHUSETTS, USA, 13 – 15 October, 2020

Enhancing Visibility in Agile Program Increment DSMs

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: *Program Increments (PIs) are sequences of consecutive sprints during SAFe implementation of agile developments. SAFe work is planned at two levels of granularity: (i) stories, which create tasks (within a sprint team) and account for task interactions at a fine level of granularity; and (ii) features (often decomposed into stories), which account for interactions at a coarser level of granularity. A common practice in PI planning, involving 10-15 teams, is to suppress interactions at the story level and focus on interactions at the feature level instead. We create two DSMs for a PI planning process – one based on story interactions, and another based on feature interactions. We find a nearly 9X increase in interactions at the finer level, i.e. more granular interaction, compared to a DSM based on coarser level interaction data. We discuss theory and practice implications for using more granular DSMs during PI Planning and oversight processes.*

Keywords: Agile, Data Granularity, Interactions, Program Increment, SAFe

1 Introduction

Agile development has been envisioned to create expedient processes for incorporating dynamic customer input into iterative and incremental software development (Agile Manifesto 2001). The very first tenet of this agile manifesto for "uncovering better ways of developing software" is valuing "individuals and interactions over processes and tools." This paper focuses on management of interactions in development projects that follow a widely used Scaled Agile Framework (SAFe) approach to agile work (Leffingwell 2008). Within this framework, interactions are managed through structured processes such as daily scrum meetings, bi-weekly sprints, periodic Program Increment (PI) planning workshops – held typically every 10 weeks, and coordination across PIs through scheduled and nested product planning cycles. Since interactions between tasks create dependencies, and examination of the network of dependencies may yield important insights for better project management, we turn to Dependency Structure Matrix (DSM) analysis. Related research has delved into the question: how can DSM analysis provide insights above and beyond conventional agile development and information processing practices (Srinivasan et al. 2019, Bajpai et al. 2019)? These earlier works showed that DSM analyses can provide insights for alternative organization of dependencies, along with predictions for the impact of these interactions on development performance.

It is common within SAFe implementations to track different types of data at different levels of granularity while managing the scope of work, by putting features and stories into different work containers, as shown in Table 1. Finer granularity in this case is defined as provision of more details in terms of defining the scope of work precisely. This difference in scope offers differing level of granularity in terms of interactions between tasks. For

instance, team tasks, typically associated with daily scrums, are focused on implementing use-case stories into software code. An example of a story during development of a user interface is a "data explorer" that is stored as a table in a data warehouse, along with the specification, "when a table is selected, display the variable names in it". Stories are typically created by decomposing features which customers would use. Building on the "data explorer" example from above, a related feature that uses "data explorer," and some other stories, could be "search a variable name across databases." Related interaction data are tracked in information systems, such as issue-tracking databases. These issues and a backlog of stories and features form the basis of discussions for planning work during daily scrums and during SAFe Program Increment planning meetings every 10 weeks.

Table 2: Definitions of work containers in SAFe

Scope Unit	Description	Time Scale	Resource Scale
Feature	A stakeholder need (specified with coarser level of granularity)	Single PI (~ 10 weeks)	Multiple Teams
Story	Single desired functionality formed by splitting a feature, (specified with a finer level of granularity)	Single iteration over several weeks within a PI	Single team

The goal of these discussions is to create an artifact known as the PI Planning Board. The unit of analysis within a Planning Board is features. A stylized example is shown in Figure 1. Dependencies in this board may be temporal (e.g., start of iteration 1.2 will depend on completion of work in iteration 1.1 for team Dolphins). Dependencies may also be shown across teams as red strings either within a single iteration (e.g., during iteration 1.3, team Dolphins provides a significant input to team Iguanas and thus a dependency is identified) or across iterations. Moreover, teams recognize that these dependencies exist both at the story and at the feature level while planning for their work within a PI. Therefore, teams plan for and track interactions at two levels of granularity: stories with finer level of granularity and features at a coarser level of granularity.

Level of granularity has been shown in the design management literature to have bearing on system architecture. For example, Chiriac et al. (2011) find that the degree of modularity can vary for the same system when the system is represented at the two different levels of granularity. They argue that the level of granularity in decomposition can distort the results of architectural analysis and care must be taken in defining the system decomposition for such analysis. Similarly, Maier et al. (2016) discuss the role of abstraction in modeling and the resulting importance of model granularity. Thompson (2019) describes allied ideas such as agile project management with Kanban, requirement definition, and resource planning for development of hardware and software products. However, this literature does not explore project management implications, such as task sequencing decisions, during agile PI planning. This motivates our key research question: *what is the level of information loss associated with PI planning when interaction data are aggregated at the feature-level, rather than at the story-level?*

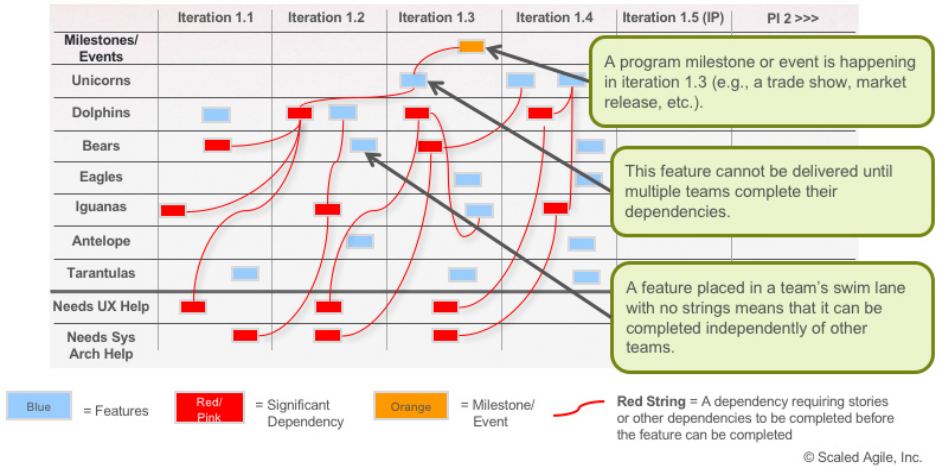

Figure 1: SAFe 5.0 Program Board (source: v46.scaledagileframework.com)

We have designed a field study to collect case evidence on interaction data at two levels of granularity and populated the two separate PI DSMs using these data. This paper reports on our field work and its findings. The rest of this paper is organized as follows: section 2 lays down the rationale for modeling PI DSMs, section 3 describes our field study, section 4 describes the two DSMs and provides a statistical comparison using ANOVA analysis. We find significant differences in the dependency structures across these two DSMs. We conclude the paper by discussing the implications of these comparisons for theories that guide PI Planning rationale. We indicate ways in which DSMs based on granular data may augment and improve conventional PI planning.

2 Modeling Program Increment DSMs

2.1 Program Increment Planning and Program Board

A program increment typically features 5 sprints (i.e., scheduled cycles of iterations). The first four sprints comprise the planned development work, including integration and testing as stories are completed by each of the teams. While the fifth sprint includes release of completed features, it is largely devoted to forward planning for the next PI, including any advance development required before the PI starts. All this work is executed by typically 10+ teams, collectively termed as the Agile Release Train (ART), which may bring in many hundreds of stories and 50-100 features into their PI discussion. The PI Planning event is typically a 2-day face-to-face workshop involving the entire ART and its customer representatives (or "Business Owners"). The goal is to employ the agile principle of "face-to-face conversation" (Schwaber 2009). The teams align on a "Program Vision" and business context, and come up with a shared plan for the program increment. Specifically, self-organizing agile teams select the features they plan to implement within the PI; these teams then coordinate to define a mutually agreed plan communicated to business stakeholders.

A key part of teams committing to a common plan is identification and visualization of dependencies between the work of the various teams, and from features to deliverable milestones. The tool used for this is known as a Program Board. The board lays out team names in rows, with each team's row forming a "swimlane" of work over the course of the PI. Each team's features are represented within the swimlane for the team, with the five sprints laid out in columns. Teams write every feature on a card and place them in their swimlane within the iteration in which they plan to complete each feature. Dependencies between teams or milestones are identified by tying cards together with a red string, as illustrated in the stylized Program Board of Figure 1.

2.2 Modeling Considerations

To facilitate meaningful face-to-face conversation, the representation of coordination requirements on the Program Board is intentionally limited. Guiding theory behind such information hiding during development is to improve clarity of message for individual teams and reduce coordination burden (Yassine et al. 2003, Gomes and Joglekar 2008, Ebert and Paasivaara 2017, Thompson 2019). Typically, only directly relevant features are represented within a team's swim lane on a Program Board. Likewise, significant dependencies are represented by a string connecting either two features, a feature and a milestone, or a feature with significant input from another team. This should serve to clarify what dependencies are, without modifying the SAFe PI Board diagram in Fig 1,

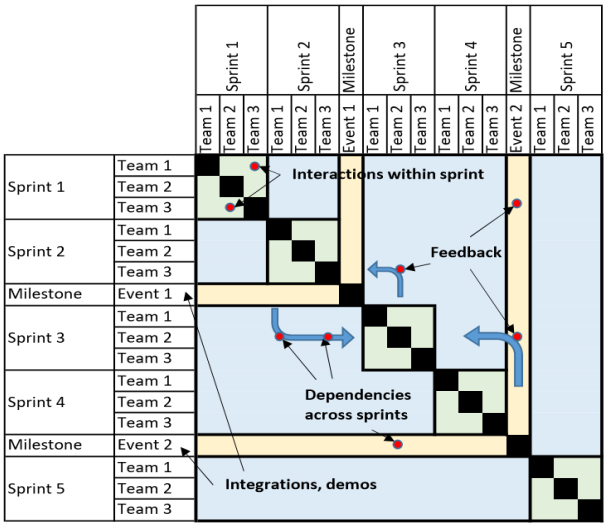

Figure 2. Stylized PI DSM

Teams may have story-level dependencies that are nevertheless important constraints on the plan. Also, the dependency represented as a string may be unidirectional, bidirectional or could even create a loop of dependencies. There may be additional risk factors, affecting the expected success of the PI Plan, not captured on the Program Board. Within these visual limitations, the ability of individual teams or a system-level owner such as the Release Train Engineer (who leads the ART) to suggest improvements or management

interventions is also limited. Key milestones are also represented on the Program Board. Examples of these include system releases (which occur at a fixed schedule, e.g. the first Monday of every month), major external releases (e.g. a product launch date), or other fixed externalities (e.g. vendor system updates, regulatory deadlines etc.). Milestones interact with the plan both as constraints to schedule and sometimes as sources of new information or change (e.g. feedback after a monthly release).

The type of DSM model we use to represent the structure of a single PI is shown in Figure 2. The DSM includes a sequence of five sprints over the PI, with the team interactions laid out for each iteration. Interspersed between the sprints are the milestones. This matrix does not specify features or stories as entities; i.e. it only reveals task interactions between teams. Additional details for allied DSM modeling choices are described in Bajpai (2020).

3 Field Study

3.1 Site

Swisscom AG is a leading telecommunications provider in Switzerland. Swisscom holds large market shares of mobile, broadband internet, and TV telecommunication in its domestic residential and commercial markets. Swisscom is known for its premium quality offerings, which command a premium price. We studied the PI Planning process for the Agile Release Train "Data Lake", which is a part of the Large Solution "Data, Analytics & AI" (DNA) program. In addition to critical business analytics services, DNA also provides storage, computation and access infrastructure and services to other Swisscom analysts and engineers leveraging Swisscom's data. The Data Lake ART, together with five other ARTs comprise the DNA Large Solution. Four of these five ARTs are analytics-focused, and one is focused on developing applications for business users. All six ARTs do their PI Planning together in a single event.

3.2 Data Collection & Processing Methodology

Data presented in this paper were collected over the course of 4.5 months in 2019 at Swisscom, the organization which serves both as the research subject as well as the sponsor for this project. Data were collected for the model described below using three means: 1) Live observation during a PI Planning session including discussions and agreements between Data Lake development team members and customer teams; 2) The PI Program Board (showing features and hard technical dependencies) and team-level planning boards (showing story-level scheduling) generated during the meeting; 3) Additional interviews with product owners from eight out of twelve development teams and two of the five other ARTs.

Our method for aggregation and processing of dependency data is consistent with established practices for aggregation and comparison of DSM metrices (see Chen and Lin 2003, and DSM literature e.g., Eppinger and Browning 2012). For each individual feature represented on the Program Board, anticipated interactions needed to successfully complete the feature in the PI were recorded. There were two sources for the interactions data: 'tasks' in a tracking system (using a software called Jira), and from field interviews.

Four categories of interactions were recorded: inputs/ enablers**,** feature/technical dependency; coordination and feedback. Team interactions were modeled for each sprint and across sprints. This means that there were interactions represented between 19 teams (12 Data Lake engineering teams + 1 architecture team + 5 DNA ARTs + 1 'team' representing Swisscom-wide interactions) over the course of five sprints in the PI. In addition, three milestone events were also included as entities. This creates a matrix of size 98x98 (i.e. 19*5 + 3). Each cell in the matrix represents an interaction between two teams. Two groups of interactions captured as DSMs (based on story-level and feature-level data) allow us to explore our research question in two steps: (i) We assemble two DSMs and compute a compositive interaction vector for each row of the respective DSM; (ii) We conduct ANOVA tests to explore if differences in these vectors amount to statistically significant information loss across these two groups of interactions. Then, we discuss the implication of this significance on feature-level planning while managing a PI.

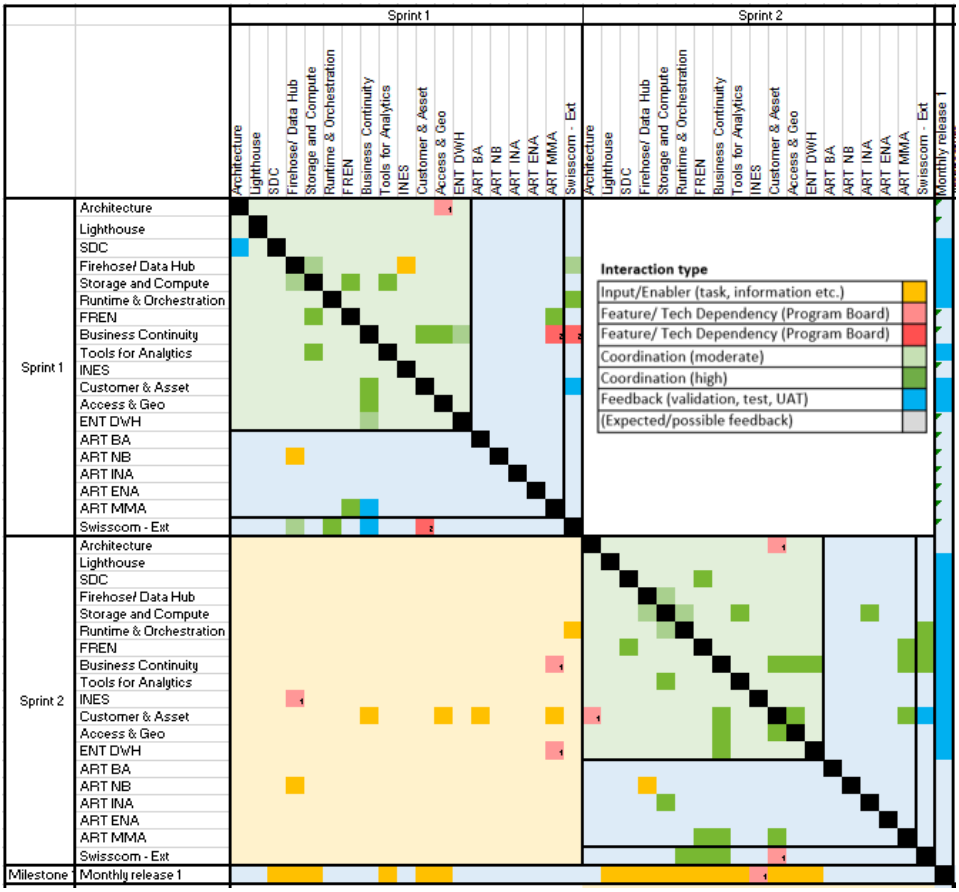

Figure 3. Two Sprints of PI DSM Based on Story-Level Data
(Number of Interactions for Full PI DSM Using Story-Level Data = 304)

4 Results

For ease of depiction, we present the first two sprints and one key milestone of the PI in a 39X39 DSM in Figures 3 and 4 based on story- and feature-level data, respectively. Categories of interactions are shown in different colors as indicated by a legend in Figure 3. A full DSM based on story-level data is shown as Figure 5. Owing to page size constraints, some details (e.g., row names) are suppressed while patterns of changes in interactions across sprints can be ascertained.

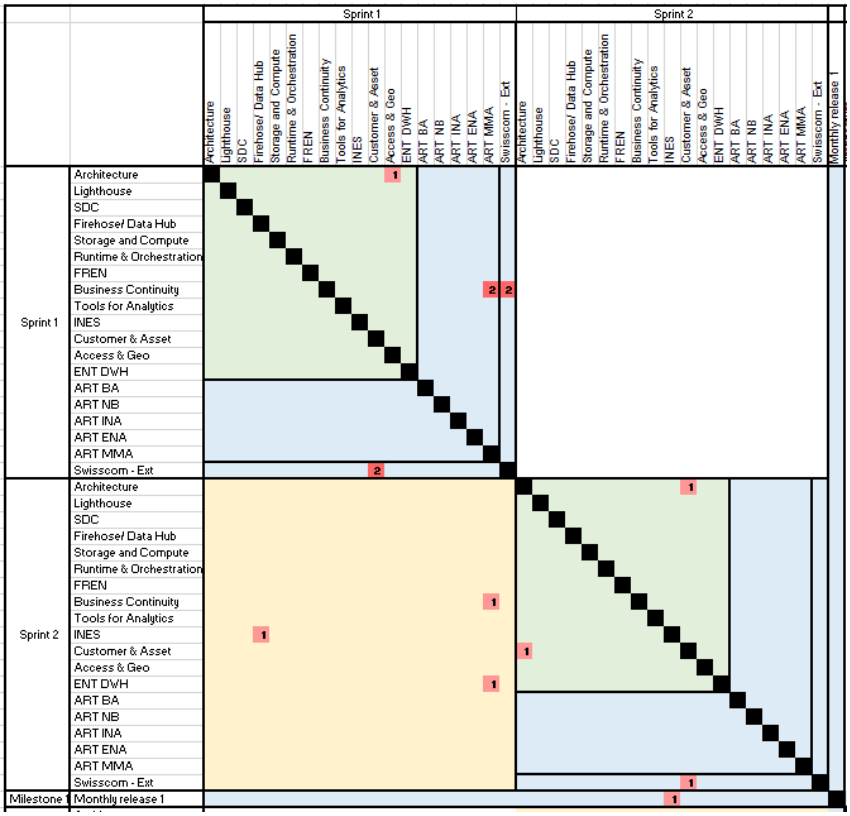

Figure 4. Two Sprints of PI DSM Based on Feature-Level Data
(Number of Interactions for Full PI DSM Using Feature-Level Data = 32)

4.1 Aggregate Statistics

We have conducted comparisons for the story- and feature-level DSMs using the full (98x98) matrices. We find that the story-level DSM captures 304 interactions, whereas the feature-level DSM captures only 32 interactions. In order to assess the statistical significance of the differences between these two DSMs, we have computed the total number of interactions in each row (termed as interaction density). Density data are then normalized between 0 and 1 to create vectors. These normalized vectors are used to compare the distribution of interactions for two DSMs. Aggregate statistics and ANOVA

results are shown in Table 2. They indicate that differences in the distribution of normalized interaction vectors for the two DSMs (termed as groups in Table 2) are statistically significant. In this analysis we have weighted all the categories equally. We have also conducted robustness checks for category weights (e.g., by adjusting relative weights on inputs versus technical dependencies). Category adjusted results (not shown here for brevity) are materially consistent with the finding presented in Table 2.

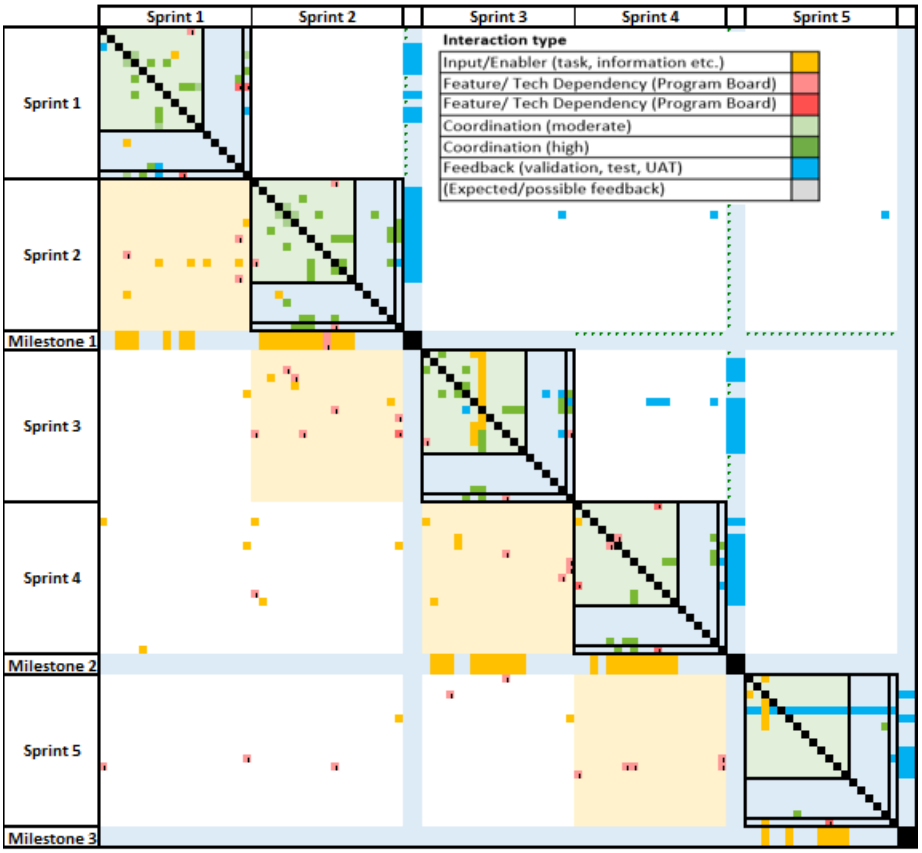

Figure 5. Full PI DSM (98x98) Based on Story-Level Data

5 Implications

This study examines the interaction structure of a PI using DSMs based on different levels of data granularity. We have identified gaps in literature, in the introduction section, that draw on intersection of system engineering, SAFe, and agile project management. Our case work and data analysis bring up follow on opportunities. For instance, research issues mentioned in section 5.1 may engender new type of theories, extending the literature on granularity in design (Maier et al. 2016) based on data with finer granularity.

Table 2: ANOVA Single Factor Analysis of Normalized Interaction Vectors

Groups	*Count*	*Sum*	*Average*	*Variance*		
DSM Based on Story Level Data	98	41.08	0.42	0.11		
DSM Based on Feature Level Data	98	24.10	0.25	0.13		
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.47	1	1.47	12.10	0.0006	3.8898
Within Groups	23.59	194	0.12			
Total	25.06	195				

5.1 Research Issues

1) Biases in Decision Making: Suppressing within-team interactions in order improve cross team communication through PI Planning Board is a recommended best practice (Scaled Agile, 2019). Our analysis shows that this practice results in a systematic underrepresentation of interactions (32 in the PI Planning Board instead of 304 that emerged from the more granular data analysis). Detailed analysis of such a sizable underestimation could enhance our understanding of planning issues: does this underestimation stem from interactions with teams that are in close physical proximity such that coordination could managed informally? What is the dynamic impact of initial underrepresentation of interactions (in sprint 1) on later sprints? And, *given this underrepresentation, is feature-level planning appropriate for managing a PI?*

2) Early Information and Limited Look Ahead: The DSM in Figure 3 shows two square sub-matrices corresponding with the teamwork in sprint 1 and sprint 2. It also shows another square sub-matrix for sprint 2, below the sprint 1 sub-matrix. We term this sub-matrix the early information submatrix. It ought to be possible to improve the predictions for sprint 2 based on the early information submatrix data from sprint 1 (Bajpai 2020). Figure 5 ascertains that this pattern is repeated in follow on sprints. That is, inter-sprint decisions before any sprint can be improved by limited look ahead analysis.

3) Adaptive Organization of PI Teams: Agile work and PI planning are set up to enable adaptive development. Composition of individual teams and interactions between teams are managed through PI planning, scrums, and scrum-of-scrums meetings. The square matrices for each sprint shown in Figure 5 could be put through sequencing and clustering analyses (Eppinger and Browning 2012) to provide guidance for sequencing tasks, and for adjusting the organizational interfaces between teams suitably. Bajpai (2020) has provided examples for such analysis.

5.2 Implementation Opportunities

Aside from the research issues listed above, fieldwork suggests it may be the possible to automate the generation of PI DSMs in order to support PI planning. For instance, data collected for the story-level DSM in Figure 3 came from a combination of field interviews and by querying the information captured by software tool (Jira). It ought to possible to automate the generation of such granular DSMs.

6 Conclusion

An overarching theme behind our research is to assess if DSM capabilities can improve the performance of agile work by enhancing current best practices such as implementation of Scaled Agile work. We have developed a new type of DSM to account for interactions during PI planning and oversight. Our analysis shows that it is possible to populate such a PI DSM with interactions based on (lower) story-level data. This generates a DSM of interactions that is statistically different from conventional PI Planning Board. Preliminary findings on analysis of granular interactions also indicates that use of finer granularity (story-level) data is a fruitful line of enquiry for research on improving the PI processes.

References

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

Chiriac, N., Hölttä-Otto, K., Lysy, D., & Suk Suh, E. (2011). Level of Modularity and Different Levels of System Granularity. *Journal of Mechanical Design*, *133*(10).

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

Bajpai, S. 2020. Planning Large-Scale Agile Development Using a Dependency Structure Mapping Model, Masters Thesis, MIT, Cambridge, MA.

Bajpai, S., Eppinger, S.D., Joglekar, N.R. 2019. The Structure of Agile Development Under Scaled Planning and Coordination. In *DS 97: Proceedings of the 21st International DSM Conference (DSM 2019)*.

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

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

Maier, J.F.; Eckert, C.M. and Clarkson, P.J. 2016. Model Granularity and Related Concepts. In: Proceedings of the DESIGN 2016, 14th International Design Conference, pp. 1327–1336.

Chen, S., Lin, L. 2003. Decomposition of interdependent task group for concurrent engineering *Computers & Industrial Engineering*, 44(3), 435-459.

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): Agile Project, Program, and Portfolio Management for Development of Hardware and Software Products.* 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: Prof. Steven Eppinger, Massachusetts Institute of Technology, eppinger@mit.edu