23rd INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2021
MONTRÉAL, CANADA, 12 – 14 October, 2021

Improving Scaled Agile with Multi-Domain Matrix

Nithin Narayanan[1], Nitin Joglekar [2], Steven Eppinger[1]

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

Abstract: Scaled agile projects require coordination of work dependencies across groups of teams called Agile Release Trains (ARTs). ARTs work on developing and delivering bundles of product features called Capabilities. Coordination can be especially complex when dependencies cut across organizational levels and product domains. We develop a field study involving Multi-Domain Matrix (MDM) mapping of dependencies across multiple ARTs and product capabilities. Our analysis finds that organizational dependencies (within a single team or an ART) associated with features that lead up to a single capability typically get tracked through periodic Program Increment (PI) planning processes. Product capabilities that depend directly on other capabilities are visible through customer delivery processes. However, some dependencies that link organizational and product capability domains may be episodic and may not be addressed through PI planning. MDM analysis can be used to both formalize PI coordination and to create dedicated problem-solving groups to improve such episodic coordination.

Keywords: ART, Capability, Dependencies, Multi-Domain Matrix, Scaled Agile Processes

1 Introduction

"A common challenge with enterprise-scale software development, Agile or not, is establishing and maintaining alignment with the vision and strategy from top to bottom across the organization. Oftentimes, there is a struggle just to align multiple business departments with the same strategy. And even when there is alignment across business departments, there is yet another challenge to ensure that strategy is clearly communicated and delivered upon all the way down at the team levels. Even with alignment, large queues and wishful-thinking may still prevail and need to be addressed." Dean Leffingwell (2016)

As the complexity and number of agile projects in large organizations have grown, coordination of work across multiple levels of development involving multiple teams within such organizations, and across a portfolio of development projects, has become a central problem. Crofoot (2020) reviews alternate mechanisms available for setting up such coordination across teams, e.g., Scaled Agile Framework (SAFe), Scrum @ Scale, Spotify, Large Scale Scrums (LeSS). In this paper, we focus on SAFe, a widely used mechanism for coordinating the development of hardware and software projects. Nearly 70% of Fortune 100 companies have practitioners who have been certified in SAFe processes

(SAFe 2021). SAFe work, as shown in Figure 1, is organized across multiple levels: e.g., portfolio (or top level), program (large solutions, or middle level) and teams (or bottom level). This figure shows that the top level involves epic owners and enterprise architects who manage a portfolio of products or large solutions. The middle level involves product managers, solution architects, solution train engineers (STEs) and release train engineers (RTEs) who manage the programs within a large solution. Bottom level involves developers, scrum masters and product owners who work in scrum teams to solve problems and build features within a program.

Information flows from top to bottom and bottom to top as a part of the SAFe workflow (Thompson 2017). In a large-scale software development organization, the information that flows from top to bottom includes high-level business requirements, priorities, value and schedule. The information that flows from bottom to top include stories and feature tasks selected by scrum teams in autonomous manner, their progress status such as burn rates, and impediments. Information exchanges may occur through in-person meetings, virtual meetings (for distributed teams), through application life cycle data management tools (e.g., JIRA), emails and reports (daily, bi-weekly through sprint completions, or quarterly through Program Increment (PI) planning. Organization of bottom-level work, within a single program increment, has received a fair amount of scrutiny (Thompson 2019), including the use of Dependency Structure Matrix methods involving stories and features (Bajpai et al 2019, Benkhider & Kherbachi 2020). However, as the above quote from Leffingwell indicates, even with great communication, the coordination and oversight of Scaled Agile processes remains a complex challenge which can yield undesirable outcomes such as queues and delays.

Efficient process outcomes at the middle level, e.g., avoidance of queues or delays, within and across program increments, are accomplished through aggregate planning. Program managers (at the middle level) are charged with decomposition of capabilities into features, and further breakdown features to stories to estimate and plan the work. Program managers are also charged with coordination of interfaces across teams. Relevant middle-level SAFe constructs include capabilities and Agile Release Trains (ARTs). Capabilities are bundles of aggregated product features. An ART is a group of individual teams. Each individual team may involve 6-10 members. An ART may involve 8-15 teams, and thus include 50-120 people. A large solution at the middle level may involve 10-15 capabilities in the product domain and 5-10 ARTs (involving up to 500 individuals) in the organizational domain.

We further describe these program-level constructs (Capabilities and Agile Release Trains) in the next section, and then discuss the research potential for improving coordination between the product and organizational domains using the Multi Domain Mapping Matrix (MDM) methodology (Eppinger and Browning, 2012). We have assessed this middle level MDM mapping problem through a field study of a large solution being developed at Swisscom Inc. Based on managerial feedback on the utility of this MDM, we find that the mapping between teams within a single ART and relevant capabilities is relatively easy to establish. However, there may be gaps in the information aggregation and dis-aggregation across the ARTs that may examined, and perhaps improved upon, through MDM analysis. We end the article by discussing research and managerial implications of our work.

.

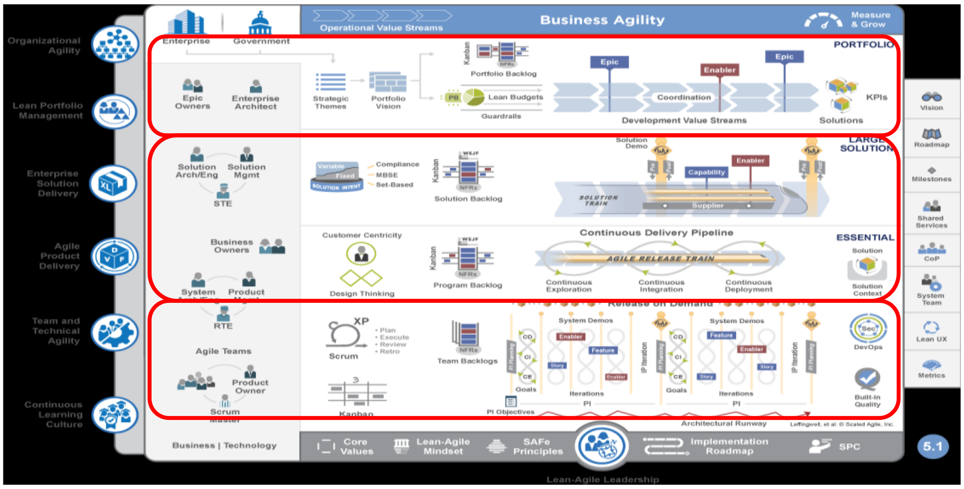
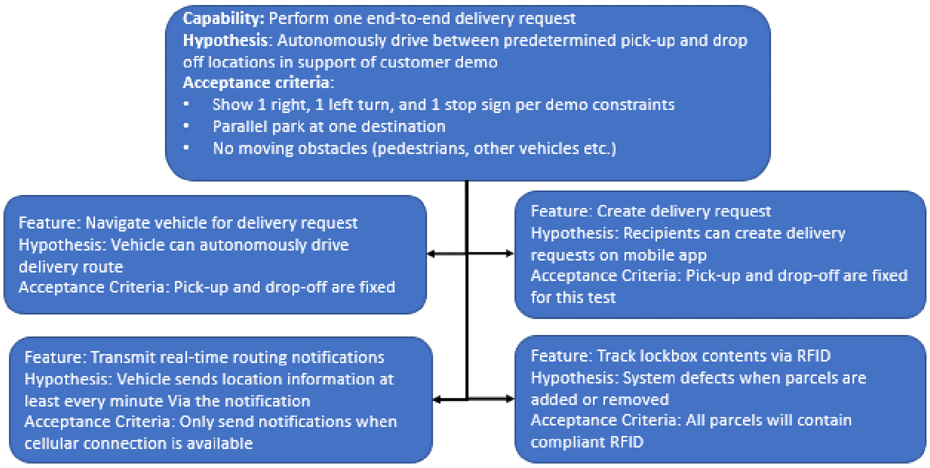


Figure 1: Multiple Levels of SAFE 5.0 Processes (Source: SAFe 2021)



Capability: Perform one end-to-end delivery request
Hypothesis: Autonomously drive between predetermined pick-up and drop off locations in support of customer demo
Acceptance criteria:
- Show 1 right, 1 left turn, and 1 stop sign per demo constraints
- Parallel park at one destination
- No moving obstacles (pedestrians, other vehicles etc.)

Feature: Navigate vehicle for delivery request
Hypothesis: Vehicle can autonomously drive delivery route
Acceptance Criteria: Pick-up and drop-off are fixed

Feature: Create delivery request
Hypothesis: Recipients can create delivery requests on mobile app
Acceptance Criteria: Pick-up and drop-off are fixed for this test

Feature: Transmit real-time routing notifications
Hypothesis: Vehicle sends location information at least every minute Via the notification
Acceptance Criteria: Only send notifications when cellular connection is available

Feature: Track lockbox contents via RFID
Hypothesis: System defects when parcels are added or removed
Acceptance Criteria: All parcels will contain compliant RFID

Figure 2: An Example of a Capability along with its Decomposition into Features (Source: SAFe 2021)

2 Literature

2.1 Key Constructs

Capability: A Capability is a high-level solution behavior that typically spans multiple ARTs. Capabilities are sized and split into multiple features to facilitate their implementation in a single program increment. An example of capability, shown in figure 2, is an end-to-end delivery request for an autonomous automotive vehicle. In order to deliver on this request, a software platform will have to build multiple features: creation of delivery request, tracking the routing notification in real time, tracking lockbox for access, and navigating the vehicle.

ART: The Agile Release Train (ART) is a long-lived group of Agile teams, which, along with other stakeholders, incrementally develops, delivers, and where applicable operates, one or more solutions in a value stream. The term long-lived refers to the goal of complete involvement (e.g., tasks needed to define, implement, test, deploy, release, and where applicable, operate solutions) across successive program increments. Figure 3 shows an example of an ART needed for delivering the autonomous vehicle capability across multiple Program Increments, with each increment lasting ~10 Weeks. It shows nine teams within an ART. Each team is responsible for one type of task ranging from business to security.

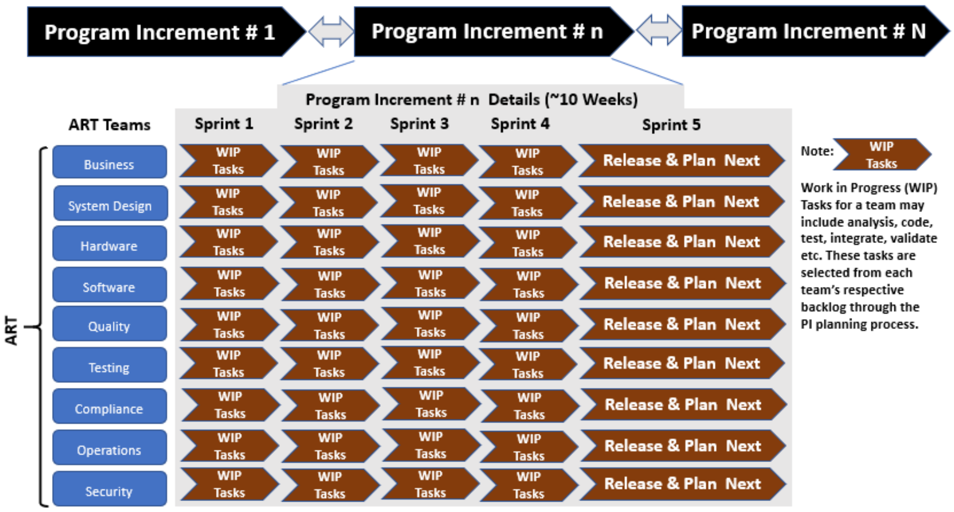

Program Increment # 1
Program Increment # n
Program Increment # N
Program Increment # n Details (~10 Weeks)
ART Teams
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5
ART
Business
System Design
Hardware
Software
Quality
Testing
Compliance
Operations
Security
WIP Tasks
Release & Plan Next
Note:
Work in Progress (WIP) Tasks for a team may include analysis, code, test, integrate, validate etc. These tasks are selected from each team's respective backlog through the PI planning process.

Figure 3: Example of an ART and its Work Progress through Long-Lived Constituent Teams

2.2 ART and Capability Coordination

Coordination tasks aimed at delivering multiple capabilities (each involving multiple features) across multiple ARTs (each with its set of teams) are complex endeavors. Thomson (1967) indicates that coordination can occur in different areas, such as vertical and horizontal areas of coordination. In the Capability – ART coordination context, vertical coordination is aimed at improving the workflow for an entire ART, while its constituent teams work on the features within the same capability: selecting stories from the backlogs, decomposing them, solving problems by planning and dividing work and responsibilities between lower level teams within this ART. Vertical coordination is typically supported by PI planning processes. Horizontal coordination is set up across ART teams working on different capabilities that require mutual adjustments on an *ad hoc* basis, by solving problems as they occur. Gustavson (2019) provides several case examples of roles and mechanisms that have been created to facilitate such coordination (e.g., Area Product owners for vertical coordination at Nokia and a cloud-based system for tracking of horizontal requirements at Ericsson).

Crofoot (2020) points out that "synchronicity of PIs is an important concept in SAFe ("develop on cadence") and it enables teams to integrate the system routinely." In this process, every team member is responsible for inter-team coordination. Additionally, SAFe defines a specific System Architect/Engineer role and a systems team. Systems design team may either be one of the teams in an ART or it may be set up as service team that supports many ARTs during each PI planning ceremony. Systems team may define non-functional requirements for capabilities, which are often cross-cutting across teams, both vertically and horizontally, requiring cooperation and buy-in from multiple teams. However, such requirements may not be able to specify all the dependencies. It is hard for individual teams, within an ART, to keep up with all these dependencies, because they draw upon a complex set of features distributed within and across ARTs. According to a RTE Engineer interviewed by Crofoot, "It was happening several times that some team decided to push something, doing it later and the other team also working on it did not notice it or [noticed it] too late." There also arises a natural tension between top-down and bottom-up work. In some cases, agile teams have removed the middle layer to have direct communication between top and bottom level. However, this is harder to do in larger and complex programs.

Some emergent research has taken an information mapping approach to characterize the agile coordination issues at the bottom level of SAFe. It addresses improvement opportunities within a single ART, for a particular program increment (e.g., Bajpai et al 2019, Benkhider & Kherbachi 2020). However, we are yet to find such data-driven characterization of dependencies issues at the program or large solution level of SAFe across product and organization domains. This leads to the following research question: **Will gaps in the coordination processes be evident through multi-domain matrix (MDM) methods, for early identification and resolution of dependency issues?**

2.3 MDM-Based Research Approach

The resolution of our research question must address uncertainties around dependencies in the product and the organizational domains. Hence, our research question has been addressed using a Multi Domain Matrix (MDM) methodology (Eppinger and Browning 2012). A key step in in our approach is the stylized MDM shown in Figure 4. This matrix represents the ART data at two levels of granularity to keep our data collection and analysis manageable:

(i) For one ART (shown as ART1 in Figure 4) in a large solution, we capture feature-level dependencies for each constituent team.
(ii) For the rest of ARTs in this solution (shown as ART2, ART3, ART4 and ART5 in Figure 4), we capture feature-level dependencies at the ART level.

We split organizational domain results in two DSMs: DSM1 for teams within ART1 is shown in green, and DSM2 (shown in dark blue) for rest of the ARTs. Both these DSM capture organizational dependencies across their constituent teams. We also introduce a third DSM (DSM3, shown in yellow) for capturing the dependencies across relevant capabilities.

Domain mapping matrices (DMM) lay out the relation between DSMs across domains. We split the DMM into two parts: brown (to depict the mapping between DSM1 and DSM3) and pink (to depict the mapping between DSM2 and DSM3).

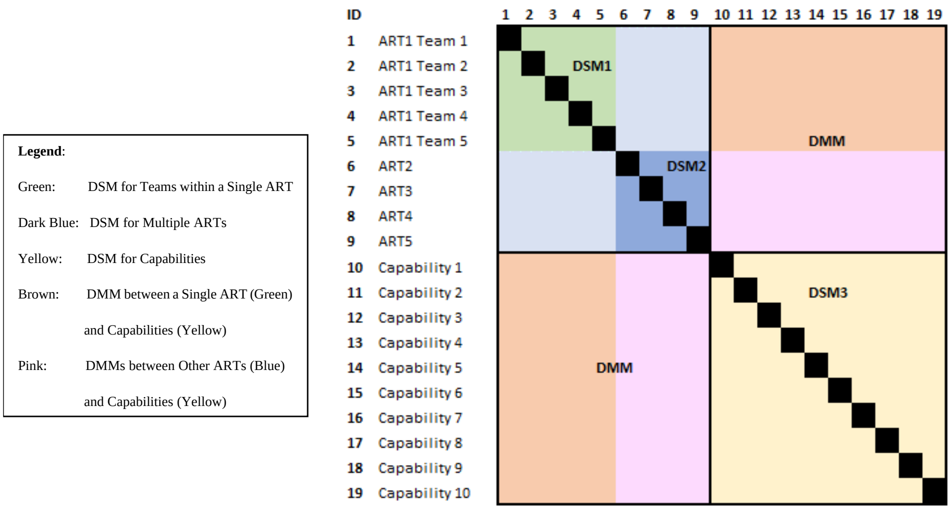


Figure 4: Stylized Structure of the MDM and its constituent DSMs and DMMs

3 Field Study at Swisscom

The implementation of our approach involves a field study to populate the stylized MDM with dependency data. We then seek feedback on the MDM from our field study principals (program managers) around insights that can shed light on our research question.

3.1 Research Context

We have collaborated with Swisscom (Switzerland) Ltd. to collect data, set up the MDM analysis and to get feedback on the findings. Swisscom is a leading telecommunications provider in Switzerland. It holds a market share of 59% for mobile, 53% for broadband internet, and 36% for TV telecommunication in its domestic residential and commercial markets (Swisscom 2021). Swisscom is known for its premium quality offerings.

We studied the Capability-ART planning process for the Agile Release Train (ART) "Data Lake", and its sister ART teams which are a part of the Large Solution called "Data, Analytics & AI" (DNA). Swisscom's Data Lake is a large, centralized data repository for structured and unstructured data from a variety of source systems across Swisscom. It also provides storage, computation, and access infrastructure and services to leverage these data. The DNA large solution contains five other ARTs focused on developing analytics and applications for business users, which depend on the Data Lake ART for this infrastructure. All the ARTs in the DNA Large Solution do their PI Planning together in a coordination event held every 10 weeks.

In our overall research context around the gaps in the ART-Capability nexus, a key challenge faced by DNA teams was managing external dependencies with other organization within Swisscom and third-party vendors. Since the external teams may have different timelines for their deliverables, it requires careful coordination with these teams for DNA teams to deliver features that require their inputs. A small miss can lead to major delays. Hence, early identification and communication are required with these external teams.

Another challenge is to understand the ART-Capability dependencies that teams within Data Lake have with other ARTs within DNA. All teams manage their dependencies in different styles decided by the team and scrum masters, and release and sprint planning occur in silos within each ART. Hence, there is a chance that miscommunications may occur while discussing dependencies and they may cause re-planning during the PI execution causing added stress to the teams.

As dependencies evolve during the PI processes, it is imperative that they are tracked and addressed. Such tracking requires a tool to identify these dependencies in a near-real-time fashion. Hence, the third objective was to figure out if it is possible to automate the dependency visualization.

3.2 Research Design and Data Collection

Research was conducted over a period of five months between February and June 2020. Data collected consisted of over 200 features from DNA Data Lake ART. The features were extracted from JIRA, the Application Lifecycle Management tool which Swisscom uses. The data collection process involved over 15 interviews with a Product Manager on DNA Data Lake team to understand adoption of the SAFe development methodology, current engineering and operational processes, and key technical and management pain points in these processes. Dependency data were analyzed and used to construct an MDM to gain insights. These insights were presented to the Swisscom leadership.

Data were collected by following means:

- Data export from JIRA - Program Increment #8 data were downloaded from JIRA on an Excel worksheet that contained information at feature level. The key elements included were:

  - Feature ID
  - Linked Issues (contains information regarding other features linked to the primary feature)
  - Labels (contains information regarding other teams that are participating in the delivery of the feature)
  - Agile teams (team that is primarily responsible for delivering the feature)
- Interviews with Product Manager of DNA Data Lake ART
- Inputs from MIT's previous engagement with Swisscom during 2019 (Bajpai et al. 2019.)

Vertical and horizontal data flows in a typical scaled agile development organization are illustrated in Figure 5. Vertical lines in the product domain indicate information exchanges (either upward or downward). These are tracked in the JIRA system. The vertical lines in the organization domain indicate mechanisms for communication across individual teams and ARTs (e.g., scrum of scrum meetings). The horizontal arrows indicate information dependencies within each level of the SAFe hierarchy (and are also reported in the JIRA system). The focus of our MDM analysis is the structure of feature-level dependencies across capabilities (product DSM) and ARTs (organization DSM), and the mapping of coordination needs across these domains (DMM).

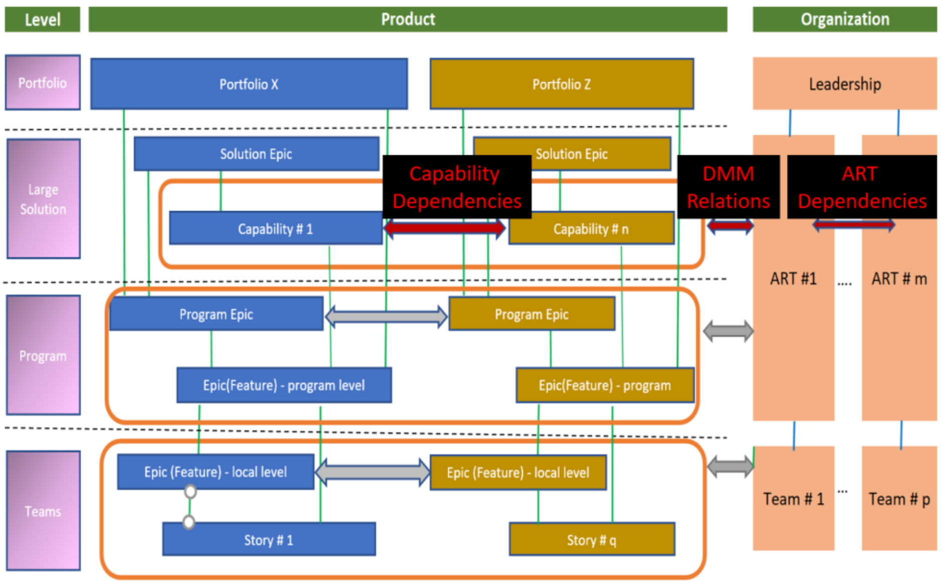


Figure 5: Multiple Types and Levels of Information Exchanges

We note that the dataset we obtained contains the features belonging to one ART – DNA Data Lake, and the information presented in the DSM is from the point of view of Data Lake team. Features belonging to other ARTs may or may not be a part of PI 8, and we did not have their feature list to validate the relevant dependency information.

3.3 Construction of Multi Domain Matrices

For each unique feature, the dependencies at the feature level and team level were identified from *linked issues* and *labels* reported in the JIRA database respectively. These dependencies are then represented in the MDM by capturing their relevant interactions:

Intra-ART dependencies: If a feature requires input from a team within the Data Lake ART, it is considered as an intra-team dependency. For instance, (i) if a *linked issue* has another team within DNADL mentioned in its corresponding *agile teams* field in JIRA, or (ii) more than one *agile team* is mentioned in the feature's *agile teams* field in JIRA, these links were reported as a dependency required by teams to deliver a capability epic.

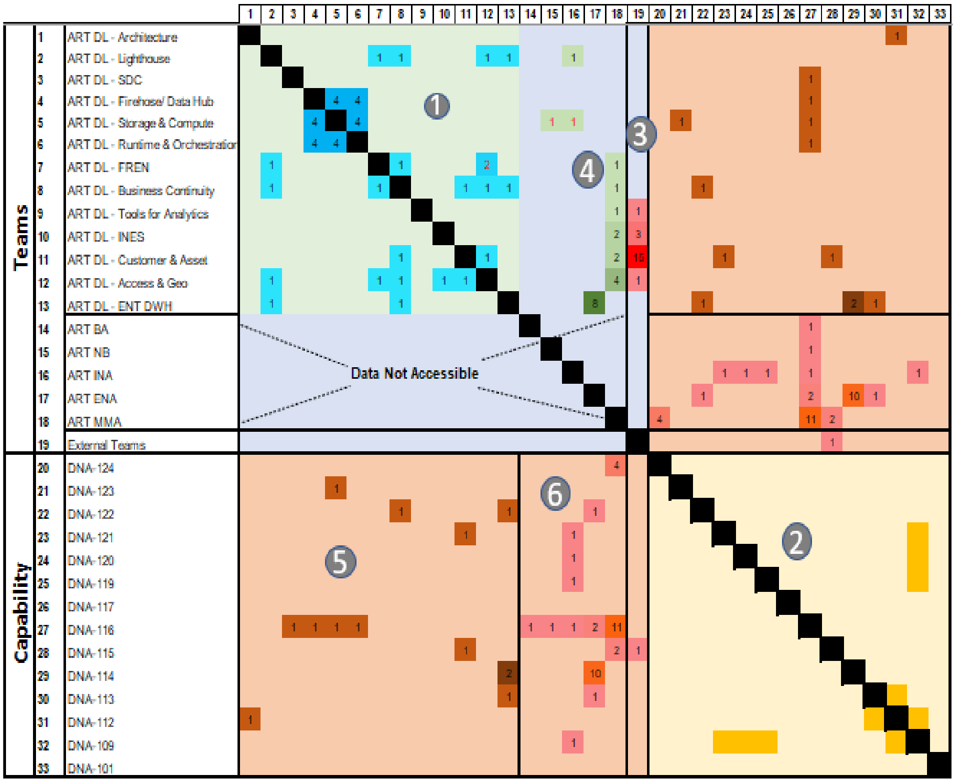
Inter-ART Dependencies: If a feature requires input from a team outside of Data Lake but is still within the DNA large solution, it is considered as an inter-ART dependency. The *linked issues* field in JIRA has features with prefixes belonging to other ARTs (e.g., DNAMMA, DNAIA etc).

External Dependencies: If a feature requires inputs from a team outside of DNA, it is considered as an external dependency. In JIRA, the feature has ExtDep listed in *labels*. The external team may be within the Swisscom organization or an external organization such as third-party vendors.

Capability – Team/ART DMMs: If the *linked issues* have features with prefixes belonging to other ARTs (DNAMMA, DNAIA etc.) their dependencies are marked accordingly.

4 Data

The resulting dataset summarized in Figure 6 has been captured into a MDM involving 13 teams (12 Data Lake engineering teams plus 1 architecture team, 5 DNA ARTs and another team representing Swisscom-wide interactions with teams outside of the DNA organization) over the course of program increment #8. This resulted in the two DSMs (organization/team and product/capability) and a DMM which maps the relationships between these two domains. The Data Lake ART has been disaggregated at the team level while the other ARTs have not been disaggregated, because disaggregated data for the other 5 ARTs were not accessible. Thus, the blue portion of the DSM representing cross-ART team dependencies (in rows 14 through 18) is shown to be empty.



Legend:

🟩	Inter ART feature dependencies
🟦	Intra DL feature dependencies
🟥	External dependencies
🟫	Capability level interaction within DL
🟧	Capability level participation by other ART
🟨	Capability level dependencies

Figure 6: Representation of Swisscom PI8 Data in MDM

4.1 Organization DSM: Team-Level Dependencies with Data Lake ART

The upper DSM represents dependencies between Data Lake ARTs. Each populated cell in the matrix represents one of dependency mentioned below and has been color coded. The number within the cell represents the number of unique features the given dependency involves. Since a cell can represent more than one feature worth of interaction, each cell contains a count of the number of features for which the teams are interacting.

4.2 Product DSM: Capability-Level Dependencies

The lower DSM represents dependencies between capabilities. Though capabilities are large pieces of work which are preferably independent, there are indeed some dependencies among capabilities. PI 8 consisted of 13 epics and the cells highlighted in dark yellow represents dependency.

4.3 Organization-Product DMM: Team-to-Capability Mapping

The DMM represents the degree of coordination required between teams to complete the capability epics. The number within the cell represents the number of unique features the given dependency involves. The marks within the DMM represent some of the coordination needed between teams that is in addition to the allied coordination effort is also be represented in the organization DSM. That is, when a dependency feature is already included either in the organizational or capability DSM, it is not tracked in the DMM.

5 Discussion

Swisscom leadership had expressed interest in establishing mechanisms for early identification and coordination of the development processes at the capability level. In our review, Swisscom recognized several benefits when we presented our findings to the DNA leadership responsible for delivering the set of capabilities shown in rows 20 to 33 in the MDM of Figure 6.

Capabilities deliver large pieces of work that involve multiple ARTs. The multi domain matrix shown in Figure 6 identifies the level (in terms of number of features) of participation required to complete a capability epic. This provides relevant teams with ability to better plan their PIs. Such planning may include creation of dedicated problem-solving and coordination (DPSC) groups based on the level of feature dependency across teams. These DPSC groups may be within an ART or span across ARTs depending on the level of participation required. Using the capability domain, work of these DPSC groups can be prioritized, for early identification of issues, based on the value of the capability being developed. We divide our discussion of results into six types of dependencies identified with circular label 1 through 6 in Figure 6.

1. DSM within the Data Lake Team (blue marks): Level of interactions required between teams may vary according to their dependencies. For example, three teams (Firehose/Data Hub, Storage and Compute, and Runtime & Orchestration) form a cyclic dependency and hence would require increased coordination within the Data Lake ART. Some of these teams may be co-located. These teams meet during the PI planning session and have additional coordination mechanisms through working sessions and frequent team meetings where the developers discuss their ideas and impediments. The mapping in the DSM for a given milestone (PI or release) can help the program leadership in improving the program's efficiency by setting up proper coordination mechanisms for the teams that have such tight and well specified dependencies. When these teams are collocated, they are more likely to be aware of these key dependencies, even without the benefit of a MDM. They may be less likely to have this understanding if they are not collocated.

2. Capability DSM (yellow marks): Capabilities involve long-lived and large pieces of work. In general, the capability matrix is decoupled (sparse). That is, to the extent possible, each capability is being developed for delivery to the end customers in a standalone manner. However, capabilities # 109, 112 and 113 have bidirectional dependencies with each other. Similarly, capability #109 has bidirectional dependencies with capabilities # 119, 120 and 121. These capabilities must be delivered as bundles because they are coupled with each other. Moreover, their development, and progress status assessment may have to draw upon multiple teams. Such dependencies are visible through the DMM view.

3. External Organizational Interactions (red marks): Early identification of external dependencies helps relevant teams in better coordinating with those teams that are outside the organization. This is a major concern for Swisscom program managers, when it comes to managing dependencies. External teams follow their own release cycles and DNA team have limited influence on their priorities, hence identifying the dependencies early in the cycle helps them coordinate. According to the DNA product manager, a missed dependency with an external team could delay a feature's release by months.

For example, Customer & Asset team requires information from external teams for 15 of their features in PI 8. These interactions do not fall within the purview of the DSM interactions described above as these teams belong to other organizations. Since other organizations follow different delivery timelines, it is imperative that the dependencies are identified early, and coordination is conducted efficiently in order to avoid delivery delays. Since, external teams may not participate in periodic PI planning events, having a map of their dependencies is a useful mechanism for improving coordination.

4. Organizational Level DSM Interactions across ARTs and the Data Lake Teams (green marks): An assessment of green marks, indicates that the JIRA database for the Data Lake ART did identify the ARTs that must communicate with the Data Lake teams frequently. For instance, column 18 shows that the MMA ART must coordinate with six different data lake teams to deliver on 11 features. Improved planning for these dependencies could be accomplished if formal coordination is planned.

5. Capability Level DMM Interactions within Data Lake Teams (brown marks): An assessment of the DMMs highlights dependencies associated with the delivery of several capabilities that require participation from multiple teams with the Data Lake ART. For example, DNA 116 (row 27) in Figure 6 requires participation from SDC, Firehose/ Data Hub, Storage and Compute, Runtime & Orchestration. These dependencies are tracked during PI Planning meeting, and it may be useful to have the DMM matrix as a checklist during such planning meetings.

6. Capability Level DMM Interactions across ARTs (pink marks): An assessment of the DMMs, by inspecting the pink marks, shows that the delivery of a few capabilities require participation from multiple ARTs. For example, DNA 116 (row 27) needs input from 6 ARTs: ART BA, ART NB, ART INA, ART ENA, and ART MMA. The capability level coordination occurs at the large solution level. Such coordination activities may be episodic and less frequent than PI planning meetings. It is not clear, based on the data collected, if there is a formal mechanism (beyond the PI planning meeting for each of these ARTs), for coordinating such dependencies. The DMM could serve as a map for guiding such episodic planning.

6 Conclusions

The key contribution of this paper is to explore how MDM data can be collected across ARTs and Capabilities at the large solution level in a SAFe program, and how visualization can improve the ability of ARTs to plan sprints. Through the MDM mapping, our visualization illustrates dependencies that are represented across multiple domains to improve the coordination during PI planning and execution. In the context of our MDM coordination map, the team level DSM (blue marks) will likely be routinely examined as a part of program increment planning processes. Capability dependencies (yellow marks) are visible, in the sense that these capabilities must be delivered to end customers as functioning and well-integrated bundles. Interpretation and tracking of the DMM (brown and pink) marks is trickier, because these dependencies take place across teams and capabilities, and some of them may potentially not be tracked and managed within regular Program Increment planning processes. We have recommended creation of dedicated problem-solving and coordination groups based on Intra-ART interactions, Inter-ART interactions, and some episodic planning within ARTs based on capability-level DMM interactions.

Such effort would improve coordination through: (i) early identification and tracking of external dependencies, (ii) planning and prioritization of capability epics based on team participation, and (iii) planning of distributed delivery of features based on levels of dependencies. However, we consider this work to be an early study related to the management of large solutions/ programs under SAFe. Our work brings up several opportunities for follow-on research.

1. We have manually extracted data (from an existing JIRA database), that usually would not be deployed to assess the multi-domain dependencies identified in this study. Such data extraction can be automated, if necessary, to maintain a real-time representation of these dependencies. This will give teams the ability to understand how coordination needs evolve when requirements change. For example, if a new feature is added that introduces a dependency between two teams, the DSM may help the relevant teams in understanding how to prioritize and coordinate the work. A critical aspect of such data extraction and usage is the ability and willingness of individual teams to keep the database up to date in terms of dependencies.

2. This research has focused on team and program dependencies for the organization at feature and capability level from a product standpoint. However, the MDM may be used to represent other multi-dimensional constructs such as (a) dependencies at story levels (see Bajpai et al 2020, for gains to be made by using such disaggregated data), (b) prioritization of coordination needs based on defects, integration testing needs that require

cross-team participation, and (c) prioritization of coordination needs based on other business relevant parameters such as business value, priority and capacity.

3. Follow-on research opportunities may involve analytics associated with MDMs. For instance, Hamraz et al (2012) illustrate a method to assess change propagation through the analysis of multidomain engineering interactions. Their analysis combines change proposition with function-behavior-structure analytics (Gero & Kannengiesser 2004) that is aimed at supporting uncertainty reduction and risk management in design process. This type of assessment may be particularly useful where the architecture is more complex and when there is a lot of coupling across ARTs/sprints.

It is also possible to extend the MDM analysis across multiple levels of assessment (Eppinger et al 2014). Such multi-level assessment of MDM filters test-coverage data by levels while using maximum and minimum function queries to isolate all the interfaces that are associated with either early or late revelations of integration risks based on the planned suite of SE-V integration tests. In an analogous manner, it ought to be possible to couple the MDM developed in this paper with more detailed interactions at the level of sprints and stories (Bajpai et al 2020) to enhance visibility. Analysis can then document the impact of such visibility, either at the sprint level or at the aggregate planning level, within a large solution that follows SAFe processes.

Overall, we hope to see agile programs take on more data-driven approaches using MDMs to manage dependencies.

References

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, Monterey, California.

Bajpai, S., Joglekar, N., & Eppinger, S., (2020). Enhancing Visibility in Agile Program Increment DSMs. In DS 103: Proceedings of the 22nd International DSM Conference, MIT, Cambridge, Massachusetts.

Benkhider, N., & Kherbachi, S. (2020). Modeling Agile Organization Under Scrum Approach and Coordination. In DS 103: Proceedings of the 22nd International DSM Conference, MIT, Cambridge, Massachusetts.

Crofoot, L. (2020). Management of Cross-Team Interfaces in Large-Scale Agile Development, Master’s Thesis, MIT.

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

Eppinger, S.D., Joglekar, N.R., Olechowski, A., & Teo, T., 2014. Improving the Systems Engineering Process with Multilevel Analysis of Interactions. *AI EDAM* 28(4): 323-337.

Hamraz, B., Caldwell, N.H., & Clarkson, J.P. (2012). A multidomain engineering change propagation model to support uncertainty reduction and risk management in design. *Journal of Mechanical Design* 134: 1-14.

Gero, J. S., & Kannengiesser, U. (2004). The Situated Function-Behavior Structure Framework. Design Studies 25: 373–391.

Gustavsson, T. (2017). "Assigned roles for Inter-team coordination in Large-Scale Agile Development: a literature review." Proceedings of the XP2017 Scientific Workshops. 2017.

Leffingwell, D. (2016). https://www.solutionsiq.com/resource/blog-post/5-reasons-to-consider-safe/

SAFe (2021). https://www.scaledagile.com/enterprise-solutions/what-is-safe/

Swisscom (2021). “Swisscom Company Profile.” https://www.swisscom.ch/en/about/company/portrait/profil.html.

Thompson, J. D. (1967). Organizations in action: Social science bases of administrative theory. New Brunswick, New Jersey: Transaction Publishers.

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

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