flowtopia

Value Stream Reference Architectures

Taking the guesswork out of team organization

1. Introduction

Value Stream Management has existed as a concept in the realms of physical manufacturing and supply chain management since its use with Ford in the early 20th century. Later refined under the Toyota Production System, and then captured by Jeffrey Liker in "The Toyota Way",[1] many of the principles have already been applied and used in technology delivery for agile and DevOps ways of working, such as Kanban, Lean and Value Stream Mapping. More recent changes in the way that software is designed, built, tested and delivered have resulted in improvements of speed and quality, but have, however, exposed new risks in the way technology functions are used to provide business service delivery.

Three key risks have driven the need to apply Value Stream Management techniques into technology — and in particular DevOps — delivery.

1. An extremely high proportion of DevOps adopters are being caught in what is termed "Mid-Tier Stickiness" where high numbers are achieving some level of value add from the transformation, but not realizing the true, full potential, as highlighted in the Puppet State of DevOps Report 2021.[2] This is typically due to silos still not being effectively removed between key areas, for example, between the technology and business layers of an enterprise, meaning Cultural, Lean and Measurement issues across the enterprise.
2. Organizations are still applying the typical technique of defining Systems Architecture and Delivery Methods in advance and then integrating these artifacts into a standard hierarchical model of organization. This typically results in either an architecture that does not achieve the desired requirements of decoupling or being loosely coupled, or an organization that is not able to effectively apply change without large scale system impact. This is caused by the principle of Conway's Law.[3]
3. As a result of the structuring of self-determining teams using agile principles, the number of tools and data repositories being used in business service delivery has grown to levels that impact holistic visibility of systems for governance, management, and support.

Through our research for this paper, we observed the intersection of Value Stream Management, *Team Topologies*, Graph Theory, and the classical sciences (Newton's and Ohm's laws). At this metaphysical intersection we examined how and why software teams work the way they do. We try to explain it, and predict it, using the most parsimonious principles as possible. Our aim is not to provide a prescriptive way to identify accurate measurements, but in line with the general outcome of any science, our goal is to explain the greatest number of observable phenomena with the fewest number of principles, confirm intuitions, and reveal new insights. We believe that this approach allows for a new way to reason about organizational structure of software teams and the impact that flow can have in the delivery of value.

Critical thinkers reading this paper may question whether what is presented here is facts, opinions or claims. They will look for evidence in the research to validate its thesis. The authors of this paper constantly questioned their own reasoning during the research. They welcome further scrutiny from critical thinkers and recommend that the reader pair the logic of this thesis with their own experience to help establish its validity.

1 "The Toyota Way", Jeffrey Liker. en.wikipedia.org/wiki/The_Toyota_Way

2 Puppet State of DevOps Report 2021. puppet.com/resources/report/2021-state-of-devops-report

3 Conway's Law. en.wikipedia.org/wiki/Conway%27s_law

Our motivation in presenting this work is to help others create efficient IT systems, software development, and technological deployments so that organizations can better deliver value at scale. We do this without any expectation of anything in return. The critical thinker should view this research against information from other reliable sources, including other people, to help reason towards better conclusions and decisions.

In this paper, we will demonstrate the need for organizations to adopt a Value Stream Reference Architecture. This will pull from knowledge sources such as the State of Value Stream Management Reports since 2021[4][5][6] as produced by the Value Stream Management Consortium, *Team Topologies*,[7] by Matthew Skelton and Manuel Pais, and the Puppet State of DevOps Report.

A Value Stream Reference Architecture is required to correctly identify value streams within an enterprise, and to implement organization around those value streams using the Inverse Conway Maneuver and Graph Theory. This paper will demonstrate how Graph Theory centralities can be applied to value stream architecture to create a reference model that is optimized for fast flow and quality. We show how Betweenness, and PageRank centralities can be utilized to identify and classify the four different team types that may exist within a team topology that uses the reference value stream architecture. Using this classification model, we can validate that the reference architecture is suitable for team topologies that are focused towards using value streams.

This paper is broken down into a number of sections. We recommend that you read each section in turn but the following is provided as a guide in regard to what is covered in each section.

Section	Description
1	This Introduction.
2	In this section we examine some of the main issues that prevent effective DevOps adoption and look at how organizational structure is a contributing factor that needs close attention.
3	Graph theory and network science is introduced in this section as a way that we can reason about the dependencies that exist within an organization.
4	This section introduces the concept of a Value Stream Reference Architecture (VSRA) and how it relates to *Team Topologies* and Value Stream Management.
5	Here we turn again to graph theory and look at how generalized flow of work can be used to classify the four fundamental team types within a Value Stream Reference Architecture.
6	In this section we look at an example Value Stream Reference Architecture and how VSRA and the inverse Conway maneuver can be used to 'organize' and 'identify'
7	The use of Value Stream Reference Architecture and its application to value stream mapping is examined in more detail in this section. We look at how interactions in the VSRA network of the organization can be used to reason about cognitive effort.
8	Here we introduce a Newtonian approach to value stream thinking. We consider how Ohm's law presents a model of flow that we can adopt when considering a Value Stream Reference Architecture. Here we introduce the four FINE dimensions of VSRA and their associated equations.
9	In this section we examine how the FINE equations and VSRA can be used to explore the concept of flow entropy. We consider the feedback loop that exists when issues are translated into new impediments and how this impacts the resiliency of teams over time.
10	This last section presents some conclusions and recommendations for the adoption of VSRA. We provide details of how to join the ongoing discussion and research.

4 Value Stream Management Consortium's State of Value Stream Management Report 2021. www.vsmconsortium.org/the-state-of-value-stream-management-report-2021

5 Value Stream Management Consortium's State of Value Stream Management Report 2022. www.vsmconsortium.org/the-state-of-value-stream-management-report-2022

6 Value Stream Management Consortium's State of Value Stream Management Report 2023. www.vsmconsortium.org/the-state-of-value-stream-management-report-2023

7 *Team Topologies*. Mathew Skelton & Manuel Pais, IT Revolution, Portland, OR. ISBN: 978-1-942788-81-2. teamtopologies.com/

2. Current issues in DevOps implementations that require a Value Stream Reference Architecture

As discussed briefly in the introduction, three key risks have driven the need to apply Value Stream Management techniques into technology, and in particular DevOps, delivery. Each of these risks will be discussed here and later used to show how a Value Stream Reference Architecture can be used to resolve those and be used in the Value Stream Management Implementation Roadmap as defined by the Value Stream Management Consortium in the State of Value Stream Management Report 2023.

2.1 Mid-tier stickiness

An extremely high proportion of DevOps transformations are being caught in what is termed "Mid-Tier Stickiness" where high numbers are achieving some level of value add from the transformation, but not realizing the true, full potential, as highlighted in the Puppet State of DevOps Report 2021. As highlighted in this report, "Despite all their DevOps talk and funded initiatives, these companies have failed to address or understand the cultural, organizational, and process changes required to adopt a new way of working with technology. They may have invested in automation—67 percent of mid-evolution respondents say their team has automated most repetitive tasks—but as an organization, they haven't addressed the organizational silos and misaligned incentives around deploying software to production that gave rise to the DevOps movement, as evidenced by the fact that a majority 58 percent report multiple handoffs between teams are required for deployment of products and services."

2.2 Conway's Law

In 1967, Melvin Conway submitted a paper called "How Do Committees Invent?"[8] to the Harvard Business Review. The submission of the paper was rejected on the grounds that Conway had not proved his thesis. Conway then submitted his paper to Datamation, a major IT magazine at that time, which published it in April of 1968. One form of the paper's thesis was:

> "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

Fred Brooks later cited the paper and the idea in his publication "The Mythical Man-Month,"[9] calling it "Conway's Law."

Organizations are still applying the typical technique of defining systems architecture in advance and then applying a standard hierarchical model of organization. This is in breach of the principle of Conway's Law. For example, TOGAF (The Open Group Architecture Framework, 10th Edition)[10] defines the Architecture Development Cycle in many phases. Phase A defines the Architecture vision which is followed by the Business Architecture.

A stage in this phase is dedicated to the creation of an Organization Map,[11] but this is differentiated from a traditional organization chart. This means that although TOGAF allows for a more fluidic view of the interactions between teams and individuals, much like team topologies, it does not preclude the existence prior of a traditional top-down hierarchy.

8 How Do Committees Invent?, Melvin E. Conway, 1968. www.melconway.com/Home/Committees_Paper.html

9 The Mythical Man-Month, Fred Brooks, pub. Addison-Wesley 1975, ISBN: 978-0-201-00650-6

10 The TOGAF Standard Architectural Development Method. pubs.opengroup.org/togaf-standard/adm/index.html

11 TOGAF Series Guide, Organization Mapping. pubs.opengroup.org/togaf-standard/business-architecture/organization-mapping.html

In typical businesses, technology teams are organized in isolation from business teams as a separate function. For example, within the Scaled Agile Framework (SAFe)[12] there are defined "Business Owners" that are intended to collaborate with "Agile Teams" of five to eleven people responsible for the technology definition, build, test and deploy where applicable. The intention is for these agile teams to then coalesce around an Agile Release Train with customer centric design thinking. In addition, the framework sets out the concepts of Lean thinking and people, defining Agile Technical Teams and Agile Business Teams which mature around value streams. These value streams are discussed in this example with SAFe,[13] in two terms:

- Operational Value Streams that are responsible for the delivery of a product or service to a customer
- Development Value Streams that convert a business hypothesis into a technology-enabled solution that deliver customer value

Additionally, as an example, when looking at the Spotify Model,[14] business representation is through the Product Owner and not from the business directly.

The reality of implementation in most organizations is very different from this view and other frameworks also provide alternative views of value streams and how they work within a business or technology. In truth, Agile Technical Teams tend to operate in an 'x-as-a-service' function with Agile Business Teams, where there is business agility. This is a reflection of the issues discussed previously in Section 2.1, "Mid-Tier Stickiness". Furthermore, with a widespread misconception that there should be NO silo walls between any teams, all Agile Technology Teams will generally have a group method of communication, for example, Scrum of Scrums.

So, implementation of value streams like that described within SAFe should not be implemented as written, but with true consolidation of value streams, Conway's Law, *Team Topologies* and the Value Stream Reference Architecture.

Following this through with Conway's Law, the result is typically a lack of synchronicity between business and technology deliverables from a business service. Meanwhile, monolithic architectures that have been broken down into microservices are actually tightly coupled resulting in a monolith of micro-services.

In the past, teams were organized around projects, which meant teams were siloed and short lived. As agile methods became popular, teams were instead organized around products. By following Conway's Law and the principals of "No Silos", this has resulted in tightly coupled architectures, requiring large amounts of impact assessment on change and removing the agility from systems development.

Implementing a Value Stream Management approach that is inclusive of both technology teams and business teams, it is proposed that organizations identify and organize around value streams, shown in Figure 6.1, defined according to Team Topology types and with Team Topology interactions. Value streams can still operate based around small product technology themed teams of five to eleven people but should instead be viewed as an integral part of the overall business service stream that is responsible for the delivery of overall customer value, and are potentially longer lasting than the products themselves.

12 Scaled Agile Framework. www.scaledagileframework.com

13 Scaled Agile Framework, organizational agility. www.scaledagileframework.com/organizational-agility

14 25 Scaling Agile @ Spotify. blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

2.3 Tools and data proliferation

As a result of the structuring of self-determining teams using Agile principles, the numbers of tools and data repositories being used in business-service delivery has grown to levels that impact holistic visibility of systems for governance, management, and support. In the 2022 State of Value Stream Management Report, 29.7% of respondents aggregate their toolchain data from several sources/tools, using methods such as dashboards. This requires an overhead of maintenance as tools change and grow. In some cases, the source information may not be easily accessible in the way required with inconsistencies in the interoperability of that data. A further 23.6% used manual collection from several sources/tools, using methods such as spreadsheets.

In the 2022 GitLab DevSecOps Survey,[15] 69% of developers surveyed said they would like to consolidate their toolchains. In operations, 39% of respondents said the DevOps data they need exists but accessing and managing it is difficult, while 27% went further and acknowledged being "overwhelmed" by the amount and scope of data available.

These sources point to an over-abundance of tools and data repositories being used in the typical DevOps toolchain, all of which complicates not just the mapping of the value streams, but also their connectivity and the ability to effectively 'Inspect' and 'Adapt,' the two stages that allow for continuous improvement in the value stream implementation roadmap after the 'Connect' step.

15 GitLab DevSecOps Survey 2022, about.gitlab.com/developer-survey/

3. Graph theory and network science

In order to resolve the issues described in the previous section, new approaches are required. Some of these are described elsewhere and referenced in this paper. These new approaches require a new way of thinking and a new holistic view of digital business transformation. This will be demonstrated in this paper using the reference architecture, whilst implementing other methods used elsewhere, i.e. Graph Theory.

The application of Graph Theory within the field of computer science is well established. Researchers have more recently used Graph Theory to explain the relationships that exist within system architecture. This is particularly useful when addressing distributed systems and microservice architectures. Nicki Watt[16] explored how network science and Graph Theory techniques can be applied to help gain insight into and explore questions about microservice architecture. Watt was able to explore microservice architecture leveraging computations for the centralities of degree and cluster coefficient, and through computation of community detection to decide whether the microservice architecture presented tendencies towards being a distributed monolith. Using a data driven approach, this analysis can be particularly useful during the decomposition of a monolithic application into a distributed microservice architecture to ensure that services are independent and loosely coupled.

3.1 Dependency graphs

In mathematics, computer science and digital electronics, a dependency graph is a directed graph representing dependencies of several objects towards each other. It is possible to derive an evaluation order or the absence of an evaluation order that respects the given dependencies from the dependency graph. Dependencies are indicated by the existence of a directed edge that has an orientation from one node (vertex) to the next. The simple dependency graph shown in Figure 4.1 has two nodes and we can state that node 'A' is dependent on node 'B.'

Figure 3.1 — Simple dependency graph where A depends on B.

In cases where a dependency results in something of value being passed between nodes, we can theorize and assert that this flow is in opposition to the directed edge between the two nodes. In the example shown where 'A' is dependent on 'B,' then the flow of work is from 'B' to 'A.'

In Graph Theory and network analysis, indicators of centrality can be used to assign numbers or rankings to nodes within a graph corresponding to their network position. They allow questions to be asked like "what makes a node important or influential?" or "how valuable is a node in connecting its neighbors and other nodes within the graph?"

Centralities for Betweenness and PageRank can be used to analyze dependency graphs to understand which nodes of a system are mostly in the path between connected points, and which nodes are most influential in the graph as a whole. dependent for flow of work, and which nodes are most influential in the generation of value.

The PageRank algorithm measures the importance of each node within the graph, based on the number of incoming relationships and the importance of the corresponding source nodes. The underlying assumption is that a node is only as important as the nodes that it links to.

The algorithm for Betweenness centrality calculates unweighted shortest paths between all pairs of nodes in a graph. Each node receives a score, based on the number of shortest paths that pass through the node. Nodes that more frequently lie on shortest paths between other nodes will have higher Betweenness centrality scores.

We can use these centralities for analysis in the observation of architectural and organizational dependencies that exist in the development of complex software systems where there are multiple nodes with many dependencies.

16 Explore your Microservices Architecture with Graph Theory & Network Science, Nicki Watt, GOTO 2019. http://youtu.be/0G5O1ffYIPI

The link between software architecture and team organization has been well established. As described in Section 2.2, Conway's Law states that "any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure." Communication that happens in the organization will typically facilitate the flow of information and ultimately that of work and value. Therefore, the need to communicate and collaborate brings into focus the presence of dependencies that exist within the organization. Following Conway's Law to its conclusion, this means that dependencies that exist between the individuals, or teams, within an organization will be reflected by similar dependencies that exist between components of the software architecture.

We can see that the types of teams and their responsibilities are important factors when designing software systems. According to Skelton and Pais "Organizational design using Conway's Law becomes a key strategic activity that can greatly accelerate the discovery of effective software designs and help avoid those less effective." Just as Watt was able to apply network science to architectural software design, the application of centralities in Graph Theory seems to be a natural extension to the strategic activity of team organization and Value Stream Management using a data driven approach. Section 8 describes this approach in more detail.

4. Setting the 'vision' of the Value Stream Reference Architecture

In creating a Value Stream Reference Architecture, we must employ the steps of the Value Stream Management Implementation Roadmap as shown in Figure 4.1 and defined in the State of Value Stream Management Report 2023. This paper will support the flow of the roadmap to demonstrate the use of the Value Stream Reference Architecture as part of the implementation of the roadmap.

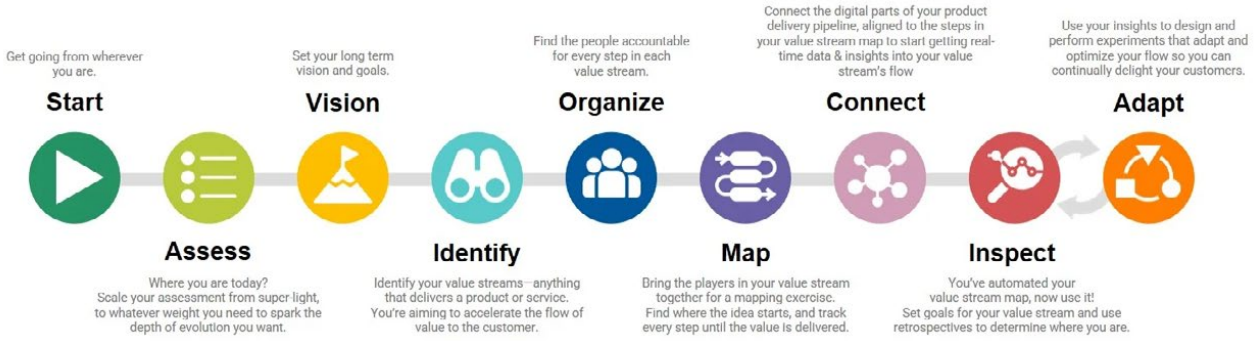


Figure 4.1 — Value Stream Management Implementation Roadmap. source: Flowtopia

The purpose of using the implementation roadmap is not just to define a standard for implementing the use of a reference architecture, but also to apply a continuous improvement and adaptation to the reference architecture. As described in 2.2, architecture and organization are inextricably linked by Conway's Law.

As such, to ensure an equilibrium in the change of both that does not introduce high levels of risk, implementation should be done in an incremental way. Furthermore, as we will discuss in this paper, all systems are eventually subject to change due to "Entropy." The system architecture, and hence the organization, will have to evolve over time as business conditions, technology and customer demand changes. This should be done in a controlled manner using the Inspect and Adapt cycle, as discussed later.

The vision step in the roadmap is defined as the point to "Set your long term vision and goals." This should ideally be in the form of a Value Statement. There are two types of value to consider within Value Stream Management.

The first type is "flow." Flow is the measure of "How" value is delivered, or "efficiency." Flow is the work's journey from idea to realization. It should be lean and continuous with a focus on delivery of value for customers. Examples of flow values are:

- **Cycle/lead/flow time**
- **Flow velocity**
- **Flow efficiency**
- **Flow distribution**

The second type is “realization.” Realization is the measure of “what” is delivered, or “effectiveness.” Realization is the fulfillment of desired outcomes. It is something that is achieved when a customer experiences value. Examples of realization values are:

- **Application/service/feature usage**
- **Customer journey time**
- **Referrals/reviews/retention**
- **Value stream P&L/ROI/PoV**

Both should be potentially considered when defining the value statement. However, Flow values are typically internally facing to improve efficiencies. Realization metrics can often be externally facing and are a true measure of business performance.

One thing that must always be considered in any implementation of a Value Stream Management approach, is to start with consideration of the business outcomes that you wish to achieve. This sets the tone for the Value Stream Reference Architecture, any implementation of it and any future incremental change. Therefore, the vision, or value statement, must consider what you are measuring in relation to the business outcomes you wish to achieve.

5. The application of Graph Theory centralities to 'identify' team topologies

In their book *Team Topologies*, Skelton and Pais propose that team variations can be reduced to four fundamental team topologies that exist within software engineering organizations. These topologies are:

- Stream-aligned teams (SA)
- Enabling teams (EN)
- Complicated sub-system teams (CS)
- Platform teams (PF)

The stream aligned teams are viewed as core, whilst the other types are all supporting. Skelton and Pais state that "when used with care, these are the only four topologies needed to build and run modern software systems." They describe that when combined with effective software boundaries and team interactions, the restriction to these four team types acts as a template for effective organization design. The team topologies act as "magnets" towards all team types where the purpose, role, responsibility, and interaction behavior become better defined to reduce ambiguity within the organization. This means that they can be used as an effective tool to apply a team organization reflective of the architectural design of the organization's systems — the inverse Conway maneuver.

We further propose in this paper that there are fuzzy characteristics in the flow of work that occurs between these team topologies that can be generalized into a more common form. Stream-aligned teams tend to receive value from all the other team types. Whilst it is possible for the other team types to receive value from a stream-aligned team (for example feedback metrics that describe overall system performance), the flow of work starts in the other teams and generally moves towards the stream-aligned teams. The enabling team type on the other hand tends to provide value to all other team types. They primarily provide a facilitation role and help to reduce the cognitive load for all other teams. flow of work is generally from enabling teams to all other teams.

Complicated sub-system teams primarily provide value to stream-aligned teams. Complicated sub-systems can be viewed as any system requiring engineering effort for them to operate; require infrequent, but resource intensive change; do not provide direct value to the end consumer, but otherwise via a stream-aligned team product; and require specialised skills and experience. Like enabling teams, they reduce cognitive load for stream-aligned teams but do this mainly through a collaborating role rather than one of facilitation.

Platform teams also provide value to stream-aligned teams, but they may also provide value to complicated sub-system teams at the same time. They reduce cognitive load for these other two teams through a comprehensive set of platform functions that have well defined bounded contexts and application interface contracts. These interfaces are more well defined (less fuzzy) when presented to stream-aligned teams. The generalized flow of work between the fundamental team types is depicted by the green arrows in Figure 5.1.

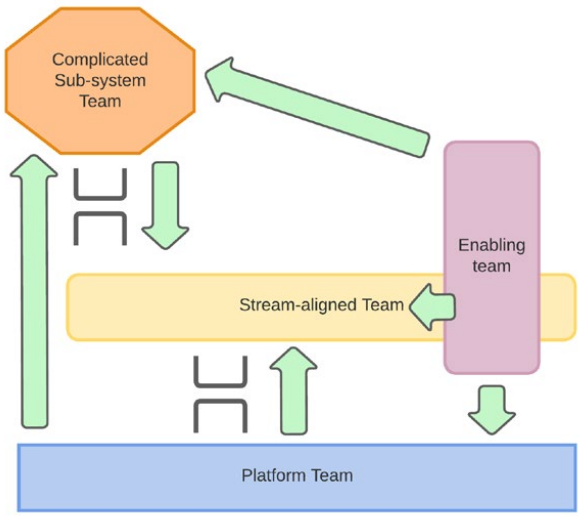


Figure 5.1 — Generalized flow of work between the fundamental team topology types.

Note that in this generalized case, we omit a specialized relationship that could exist where value flows from a platform team to an enabling team; for example, as is the case with GitOps. Given that we understand and accept the generalized flow of work that happens between the four team types, we can also extrapolate the generalized dependencies that may exist between them. As theorized and asserted in the previous section, the flow of work happens in opposition to the dependencies that exist between two connected entities. With this in mind, we can establish a directed graph that depicts the dependencies that may generally exist between the four team topologies. The graph of generalized dependencies is shown in Figure 5.2.

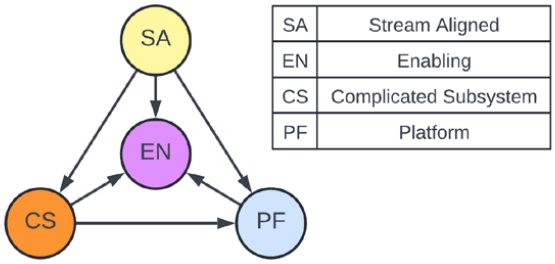


Figure 5.2 — Directed graph of generalized dependencies between team topologies.

From this directed graph of general dependencies, we can see that stream-aligned (SA), complicated sub-system (CS), and platform (PF) teams may all have dependencies on enabling (EN) teams. Stream-aligned teams may also be dependent on complicated sub-system teams and platform teams. Complicated sub-system teams may be dependent upon enabling teams. With this graph constructed we can now perform network analysis to determine centralities of Betweenness and PageRank. The analysis of centralities is shown in Table 5.1.

Team Type	Betweenness Centrality	PageRank Centrality
Stream-Aligned (SA)	0	0.1041
Enabling (EN)	0	0.8718
Complicated Sub-System (CS)	0	0.2115
Platform (PF)	0	0.4294

Table 5.1 — Betweenness and PageRank centralities for generalized dependencies of team topology

Looking first at normalized values for the PageRank centrality in Table 5.1 we can see that stream-aligned teams have the lowest value (0.1041), followed by complicated sub-system teams (0.2115) and platform teams (0.4294), and enabling teams having the highest value (0.8718).

In the graph of generalized dependencies for team topologies we can postulate that this centrality score corresponds to how important a team is in the realization of value when viewed through the lens of dependencies. An alternate way of thinking about this is to consider the impact of the team if it were removed or if it hindered the flow of work. Thus, the value for PageRank is an indicator of the team's potential to impede flow.

Teams that have a high value for PageRank must reduce friction for all other teams to lessen the impedance that they will have in allowing value to be realized. Enabling teams have the highest value of PageRank because they provide value to all other team types, so it is imperative that they do this efficiently with frictionless flow. Platform teams and complicated sub-system teams have mid-section values for PageRank when compared in distribution across all teams. Platform teams have higher values compared to complicated sub-system teams. These teams must also appreciate the role they play in providing value to stream-aligned teams and should do so with as little friction as possible. Stream-aligned teams are at the heart of delivering value and are represented with the lowest score for PageRank. They are least likely to hinder the flow of work and so can be uniquely positioned to amplify its realization.

Looking next at the Betweenness centrality score for all team topologies in this generalized case, we can see that all teams have a value of zero. This means that in the graph of generalized dependencies for team topologies there are no paths that exist where a dependency is transitive — each team topology is connected directly by at least one dependency to all other team types.

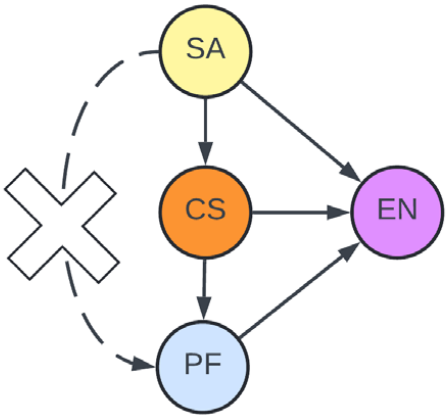


Figure 5.3 — Directed graph of non-generalized case.

In real organizations, the generalized case of dependencies may not hold true for all situations. Consider the directed graph in Figure 5.3 that shows a non-generalized case where the stream-aligned team is allowed to collaborate with the complicated sub-system team but does not collaborate directly with the platform team. The value derived from the platform team may only be obtained through collaboration with the complicated sub-system team. There is a transitive dependency between the stream-aligned team and the platform team in this non-generalized case. This alters the centrality scores that we see compared to the generalized case. Table 5.2 shows the modified scores for the non-generalized case.

Team Type	Betweenness Centrality	PageRank Centrality
Stream-Aligned (SA)	0	0.1106
Enabling (EN)	0	0.8938
Complicated Sub-System (CS)	0.1667	0.2326
Platform (PF)	0	0.3671

Table 5.2 — Betweenness and PageRank centralities for a non-generalized case

In this modified case, we can see that the normalized Betweenness centrality score for the complicated sub-system team rises to a value of 0.1667 and remains at zero for all other teams. To derive value from the platform, the stream-aligned team must collaborate with the complicated sub-system team. There is a single path for this value to flow through the complicated sub-system team such that its Betweenness centrality score rises. Stated another way, value that originates further away from where it is realized will cause the Betweenness centrality scores to rise for those teams that are "on the path to value."

We could postulate that for real world organizations, platform teams and complicated sub system teams will have a higher value of Betweenness centrality when considered within the context of the whole organization. This is because they are typically further away from, or on the path to, where value is finally realized. Because of this, the anti-pattern of frequency of new value is likely to be lower for teams who have a higher centrality score for Betweenness as they culminate changes into less frequent but larger scale risk updates. They will internally serve new items of value less often, in hopes of providing higher stability for the teams that they support. Instead, teams with higher Betweenness centrality, should consider that value will be pulled less frequently and that change must not be forced upon their consumers.

Good API version control and placing new value behind feature flags can improve frequency for these teams. On the other hand, stream-aligned teams and enabling teams are likely to have lower scores for Betweenness centrality since they are often closer to where value is ultimately presented. They will externally serve new items of value more often to keep up with customer demand.

Another interesting consequence of the non-generalized case, is that scores for PageRank centrality (as observed in Table 5.2) have risen for the enabling team and for the complicated sub-system team, but the score has fallen for the platform team. The complicated sub-system team and the enabling team have become more influential in the organizational network meaning that their potential to introduce friction and impede flow has grown. This makes sense since the value that the stream-aligned team leverages from the platform team in the generalized case must now be obtained through these other two teams in the non-generalized case. The platform team, on the other hand, is now less influential and so has less potential to create friction. It services only one consumer so has better focus on work that it needs to get done for the organization.

Considering the graph of centralities together, we can see that teams who are nearer to where value is realized, or deliver that value more continuously with higher frequency, will tend to have a lower score of Betweenness. Those teams that deliver more sporadically with less frequency, or are further away on the path to value, will have a higher (non-zero) Betweenness score. Orthogonally, those teams who have a lower potential for friction and are more unlikely to impede the flow of work will have a lower score for PageRank. Those teams who have a higher potential for friction, where the impediment of flow is more possible, will have a higher score for PageRank. This creates a simple relationship model when evaluating team types.

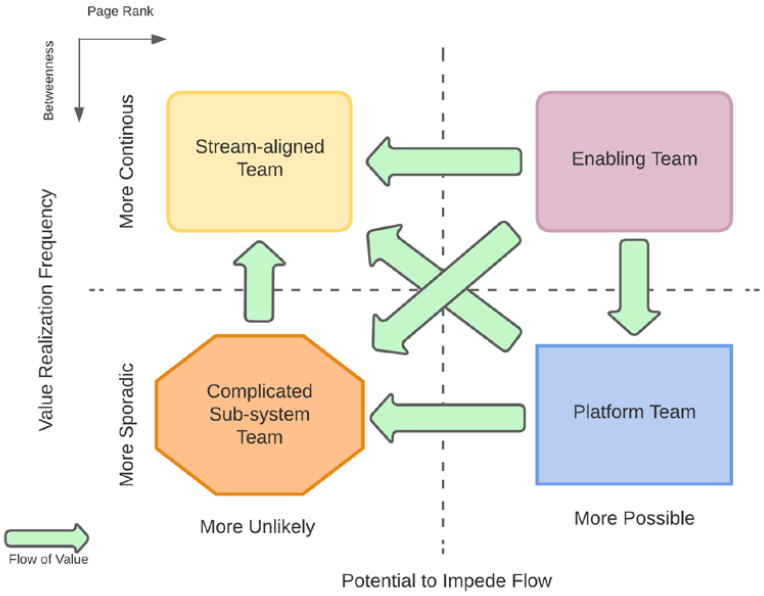


Figure 5.4 — Team type relationship between value realization frequency, and the potential to impede flow.

The diagram in Figure 5.4 shows how these relationships can be viewed in the generalized case with respect to frequency of value realization, and the potential to impede flow. Shown using the team topology graphical scheme, the generalized flow of work is shown by green arrows. Each team type lies in its own quadrant depicting its unique combination of value realization frequency and potential to impede flow.

The relationship between the likely team type and scores for the two centralities of Betweenness and PageRank combined is shown in Table 5.3. This relationship model allows for the creation of a classification process that we can use in the identification of team type when they are not well understood.

Betweenness	PageRank	Likely Team Topology
Low	Low	Stream-Aligned Team
Low	High	Enabling Team
High	Low	Complicated Sub-System Team
High	High	Platform Team

Table 5.3 — Relationship model between centrality and team type

There is further opportunity at this point to apply Wardley Mapping to the team organization, identified with team types, team interactions and potential opportunities for modeling better behavior. By applying your team structure into a graph and highlighting the nodes into the determined team types and tagging edges with the specified interaction style, it may be possible to transpose the Graph onto a Wardley map, with the 'y-axis' identifying the end user exposure and the 'x-axis' identifying the nature of the product.

That is, the higher the team on the 'y-axis', the more exposed they are to the end user, the lower, the more back-end the team is and with little to no exposure to the end user. For the 'x-axis', this would mean that the further to the left the team, the more generative the product being created, to the right, the more commoditized nature of the product. Additional methods of using Wardley Maps is explained later in Section 8.

6. Using the inverse Conway maneuver to 'organize' and 'identify'

The inverse Conway maneuver is a method of applying architectural change to your system by first addressing the organization's communication structure. Basically, in order to achieve the architecture you want, use Conway's Law by defining an organizational structure and communication pathways that reflect that architecture. This is in complete accordance with the Value Stream Management Implementation Roadmap as defined by the Value Management Consortium in the State of Value Stream Management Report 2023, which requires the identification and organization of Value Streams before any technical implementation.

Therefore, in defining our organization there are certain considerations that must be followed with the team types as discussed in the previous section for identification. These are displayed in Figure 6.1 below with particular elements of concern described below.

6.1 The Stream-aligned team

A stream-aligned team by its very nature is purely a team for the delivery of value. Its primary concern is value realization to the end customer. Given that, within any organization the method for delivery of value to a customer is a 'business service,' it makes sense that this should be the primary form of 'stream-aligned team' for value realization.

As identified in *Team Topologies*, there are two principal stream-aligned team types to consider. A business service stream and a more technical DevOps service stream. However, it could be argued that for more seamless delivery between business and technical functions, these could reside within the same Stream-aligned team. This means inclusion of business functions into the stream aligned teams. For example, the production of marketing and legal assets in alignment with technical product delivery. This reduces any potential cadence issues in delivery between technical and business capabilities — possibly to zero.

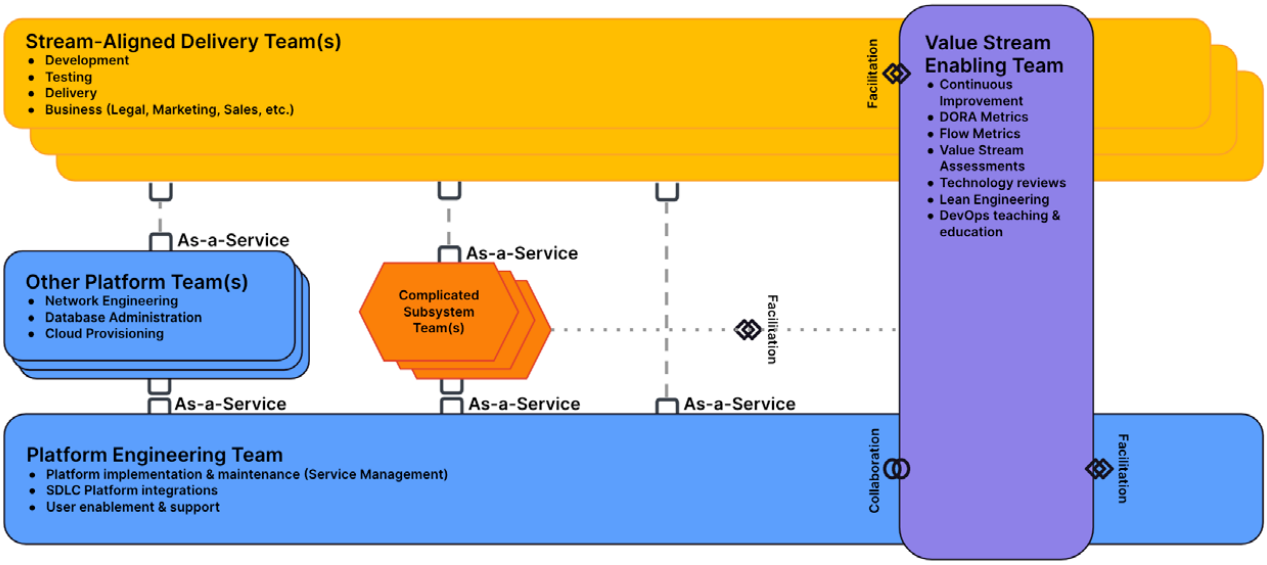


Figure 6.1 Example Value Stream Reference Architecture

An important factor to consider is that this team's focus is purely upon the delivery of value to the end user. One or more of these product teams will exist within an organization. If a Stream-aligned team were to provide value from its product to another value stream, it should no longer be considered a stream-aligned team, but should be seen as a platform or Complicated Subsystem team providing a shared service. This is as per the generalized flow of work between the fundamental team types discussed in the previous section.

The Stream-aligned team should consider not just the use of a DevOps pipeline as part of its value stream for delivery flow, but also for feedback flow, for example, Defect Management, as defined in The Three Ways.[17] As part of the organizational structure, and considering the inverse Conway maneuver, careful consideration should be given to the communications pathways between technical teams, as defined in *Team Topologies*. Collaboration between teams can promote rapid innovation, but increases cognitive load and risks creating tightly coupled integrations, so collaboration should be encouraged, but limited. Ensuring well documented and available APIs should ensure continued integrated working, with high value delivery cadence, providing a loosely coupled and decoupled operation when implementing fracture planes.

An often overused consideration in DevOps is the removal of silos. Whilst this is considered good practice, there are times when implementing silo walls is required. As part of the digital transformation, organizations have moved from monolithic architectures, to ones defined by microservices. Whilst this has introduced easier management of small incremental changes into parts of a digital system, or products, the removal of all silo walls has led to a larger scale of tightly coupled integration between those microservices. This is as a result of Conway's Law and the interaction types identified by Skelton and Pais.

For example, where two microservice product teams regularly communicate and collaborate, they typically create tightly coupled integrations and dependencies; where restrictions are applied between teams in how they communicate, i.e. x-as a-service, integrations and dependencies are restricted to APIs and documentation. Therefore, if all silo walls are removed and all teams actively collaborate in a two-way manner, sharing code and access to code, then all microservices are highly likely to be created with tightly coupled integrations, resulting in a 'monolith of microservices' or a 'distributed monolith,' as discussed in Section 2 previously.

The result is that any change in one product by one agile team will require impact assessment and possible further development in any of the other products, each of which could create further change requirements in other products resulting in an ever increasing amount of change and, therefore, risk.

By carefully considering which products are required to be tightly coupled, preferably as few as possible, even nil, we should establish fracture planes where we require products to be decoupled. Therefore, interaction between stream-aligned teams should be limited to APIs, appropriate documentation and Requirements for Change.

6.2 The platform engineering team for value stream management

As defined by Skelton and Pais in *Team Topologies*, "The purpose of the platform team is to enable stream-aligned teams to deliver work with substantial autonomy." In other words, it provides value, but only internally to the business services and DevOps streams rather than to the end user or customer. This flow of work was discussed earlier in Section 5 and shown in Figure 5.1.

When a platform team is identified, it is important to ensure that the correct communication structure is set up between that team and any stream-aligned teams. As highlighted in Figure 5.1, platform teams have a high potential to impede the flow of work, so ensuring that they operate untethered from the stream-aligned team is important. However, the importance of the internal value they provide cannot be underestimated.

In terms of how Value Stream Management is applied for stream-aligned teams, the roles of the platform team are then clear. There is a distinct value in the provision of a platform service that allows the stream-aligned teams to deliver value, unimpeded, with continuous improvement and in such a way as to ensure the reduction of cognitive load on stream-aligned team members.

In the Value Stream Reference Architecture, this is demonstrated by a platform team operating as-a-service. Any changes made to a Value Stream Delivery or Management Platform should be on an as-required basis, non-disruptive to the flow of work through the stream-aligned team, but where value is added to the capabilities of the stream-aligned team. Platform team members are typically administrators of the platform, applying changes, providing support and evaluation of new technologies and tools that can support the stream-aligned teams.

17 The DevOps Handbook, Second Edition. Gene Kim, et. al. IT Revolution, Portland, OR. ISBN: 978-1-942788-00-3.

In turn, a platform team should be considered as a value stream, but one that provides value to an internal customer ONLY. In addition, the services that a platform team provides should be constantly and consistently updated by taking mainstream functions away from stream-aligned teams when they are considered to be adding cognitive load to remove burden but provide additional value add services for all other stream aligned teams.

For example, where an operation for a particular value stream may be unique to that service, it could remain in the Stream-aligned team. However, where delivery may be consistent across all value streams, it may be more useful to consider this the function as a platform team, providing assets and support for all Stream-aligned teams, removing the cognitive burden of considering the change implications from those teams and ensuring consistency of approach across the organization. This is where the Value Stream Reference Architecture model can therefore provide value, by modeling differing construct types of organization to identify the one that allows for faster value delivery, with consistency and reduced cognitive load.

New roles are emerging in the market place to provide this level of operation. Value Stream Leads within a platform team will typically provide guidance and enablement for Value Stream Engineers whose role it is to provide what is being termed an Internal Developer Platform. The Internal Developer Platform is the holistic collection of technology and tools, which will include planning, version control, continuous integration, test automation, release orchestration, continuous deployment (and rollback), monitoring, security testing and analyzing value stream metrics. In terms of the Value Stream Implementation Roadmap, they are principally concerned with the "Connect" in applying the platform as a service for the stream-aligned teams.

In accordance with the issue identified in Section 2.3 on data and tools proliferation, this should ensure the seamless integration of the toolsets, but must also ensure a complete holistic view of the data captured so that a full and all-encompassing view of the performance of the value streams can be determined. These are shown through flow metrics, for example DORA, however, the platform team should only be responsible for the visibility of these metrics via the platform for the "Inspect" stage of the Value Stream Management Implementation Roadmap.

6.3 The enabling team for value stream management

As for the enabling team, Shibata stated, "there should be a way to give feedback to platform developers and how the platform is doing in general. Without this, the platform lives in isolation with the rest of the company. Adoption will be strenuous at best."[18] To avoid simply transferring cognitive load from the stream-aligned teams onto a singular platform team, the requirements for identification, organization, inspection and adaptation should be held within the responsibility of a value stream enabling team.

Unlike the platform team whose interactions with the stream-aligned team are principally as-a-service, the enabling team should be collaborating during the identification and organization stages, but facilitating during inspection and adaptation.

The purpose of the Value Stream Enabling team is to assist with the creation of new value streams in meeting business outcomes and flow metrics expectations, provide continuous improvement for existing stream-aligned teams and to reduce the cognitive load on all teams, but in particular, the Platform Engineering team ensuring provision of the platform as-a-service.

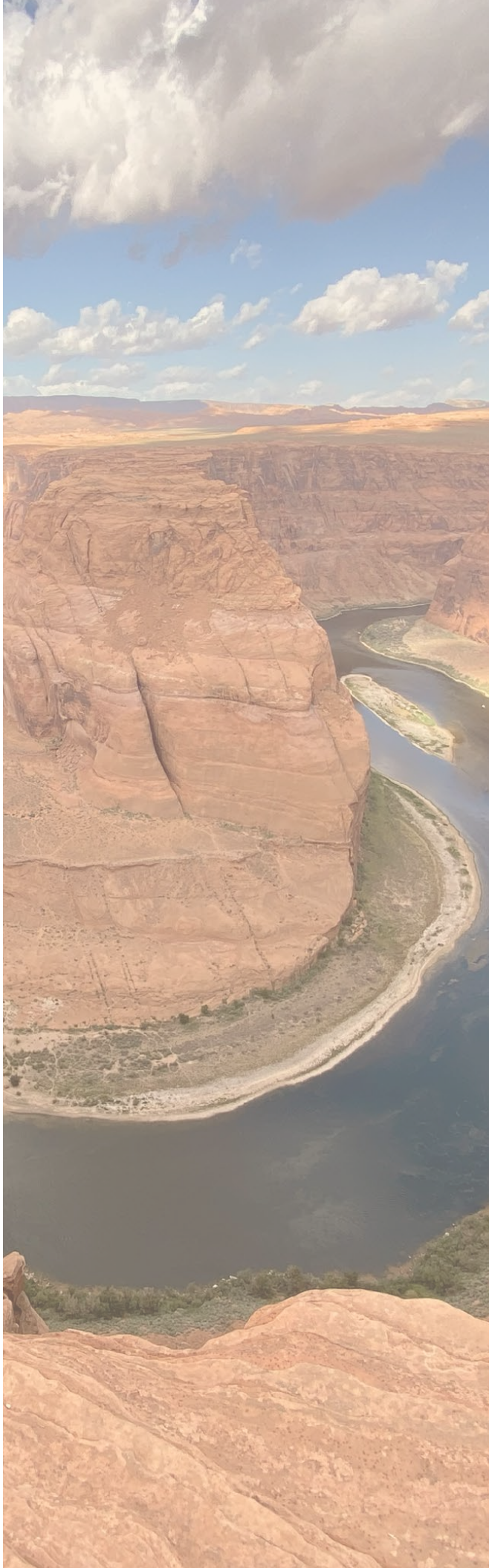
Emerging roles in this area are the Value Stream Architect, who should be reviewing the overall value stream architecture to ensure that both flow and value realization metrics are being met, and the Value Stream Facilitator, whose role it is to oversee Value Stream Mapping exercises, as well as provide insight from inspection and recommendation for adaptation. Engagement is typically with one value stream at a time with insights and benefits shared and made available to other value streams as part of the platform engineering service.

This shows an evolution in the way that architects operate within any organization. In the past, architects have operated at different levels: Enterprise architects, who were often portrayed as sitting in ivory towers determining the technological landscape of the future with little scope of organization and business outcomes. Business architects, who often defined the high-level business requirements for the enterprise, but with little view as to how this could be technically achieved. Solution architects, who gathered the dictates of the previously mentioned roles to define the technical requirements to meet the business need but within the confines of the pre-defined hierarchical structure.

18 Shibata, "How to build a platform team now!" faun.pub/how-to-build-a-platform-team-now-the-secrets-to-successful-engineering-8a9b6a4d2c8

Architects now have to balance all of the aspects of CALMS,[19] which stands for Culture, Automation, Lean, Measurement and Sharing. They must consider the behaviors of teams, meaning their responsibilities and communication pathways. They must ensure and define the automation to be used to deliver the flow. They must ensure that the flow and feedback methods applied are lean and are measured, and that those measurements are used to calculate efficiencies, points of evolutions and, most importantly, value. In other words, the actions of the architects should be determined by the value, or vision statement, defined in Section 4 previously.

Finally, it should also be their responsibility to ensure that this is shared across the organization and for this reason the role of the architect will most likely sit within an enabling team that delivers on Value Stream Management and Delivery. This detail of ownership for the Value Stream Architect role is down to the implementing organization. If the technical risk is a risk to the delivery of value, then the Value Stream Architect should own it, otherwise, it sits with the respective team who can ask for assistance from the Value Stream Architect as an enabling function.



19 CALMS Framework, Damon Edwards, John Willis & Jez Humble. itrevolution.com/articles/devops-culture-part-1/

7. Value stream mapping and the Value Stream Reference Architecture

The actions of mapping a value stream have been long understood within DevOps, however, the remit and approach often differ. In standard manufacturing and supply chain systems, the approach to mapping is simplified to represent a single stream of a flow of materials from conceptualisation to realization. In the world of digital supply this is different.

In supply chains, the design, specification and construction of the assets to be produced is largely pre-defined in components. It is only rarely that this process is changed and usually due to a fault being found within the construction process. This is represented in the Toyota Production System by a pull cord called the Andon Cord. People working on the production line would pull the Andon Cord at any time to stop production should they find an issue.

An additional conceptual difference is how value streams are visualized. In manufacturing, most of the process control is during production and this is typically stable with infrequent reasons to change other than the Andon Cord or a change in business outcomes. Given the nature of digital business systems, the frequency of change is much higher and the number of products that combine to create a business service can vary extremely. As such, when mapping a value stream for a team producing a digital product, this is typically done within the isolation of one value stream. The reality, however, is that business services, and the digital products within them, typically interact on a frequent basis. So one value stream can impact another based on the type of interaction. There may be cause at times for a more holistic view of the mapping process for digital systems.

7.1 Team interactions and value stream mapping

For many mapping exercises, the concentration is on the flow of work. For digital deliveries, we must consider more than just flow, we must consider feedback. However, we must also consider Conway's Law as discussed in Section 2.2. The principal interactions mapped in the flow of a digital delivery are represented in the CI/CD pipeline and are often resolved by the implementation of an automated orchestration tool. This has been proven to be extremely effective at speeding up the time to delivery and improving quality through repeatability, but it is also known to have increased the cognitive load on development teams as they are required to maintain these pipelines as well as their expected code output.

Much of this cognitive load can be removed through the use of enabling, platform and complicated sub-system teams. Platforms teams can remove the need for stream-aligned teams to provide continual maintenance, support and future analysis of the platform tools required for stream-aligned teams to deliver value in an automated way. Likewise, they can ensure data interoperability between each of these tools to provide holistic value analysis. Enabling teams can then provide the facilitation required from time to time to identify, organize, map, inspect and adapt teams. There may also be an interaction between stream-aligned teams and complicated sub-system teams where there is functional usage needed. Careful balance is required when deciding team types. Platform teams, for example, can greatly help magnify value but they can also be a force majeure in impeding flow.

> *"The danger of platforms is premature 'platformization' of aspects deemed standard or with high cohesion but turn out to be the opposite in reality. Platform teams struggle to support very different use cases for what looks like similar functionality while stream-aligned teams struggle to move fast because their dependency on the platform means they can only go as fast as the latter.*
>
> *This is a case where high value variation is blocked by platform services trying to avoid what has prematurely decided as low value variation. Especially when we're talking about 'core functionality' for a set of products/services. The worst is that these decisions are rarely questioned over time, even when teams are suffering from them."*

– Manuel Pais, Team Topologies. Kindly reproduced with permission.

As noted by Skelton and Pais in *Team Topologies*, there is a 'typical' interaction between these teams which must be considered when determining the flow of work. The interaction modes are noted as 'collaboration,' 'x-as-a-service' and 'facilitating.' The typical interaction between teams is driven in a business-as-usual operation, where the day-to-day workings of the stream-aligned team is in delivering value.

The mapping stage for Value Stream Management is a different matter. During the Value Stream Mapping workshop, non-typical interactions, noted in *Team Topologies* as 'Occasional' would be expected to be the standard. In order to map the 'typical' interaction of the flow of work between teams, all teams must potentially come together with "High Interaction and Mutual Respect." In order to ensure the limited time of this interaction between teams, and to maintain Conway's Law, the Reference Architecture in Figure 6.1 determines certain interactions during the mapping period.

7.1.1 Stream-aligned team with enabling team during mapping

The stream-aligned team must operate in a "collaborative" way with the enabling team for a very limited period. In this case, the enabling team is providing expertise and guidance in Value Stream Management and Delivery, providing external perspectives and alternative views to value delivery to the stream-aligned team, which must remain open minded. The stream-aligned team is providing the perspective for the value that they must deliver and the requirements that they are needed to meet. For the rest of the time, outside of a mapping workshop, the interaction should be considered as "facilitating." At the end of the mapping stage, after a mapping workshop, there will also be some "collaborative" interaction between the stream-aligned team and the enabling team as the value stream flow implementation is validated.

In Table 5.3, we can see that an enabling team has low betweenness, but a high PageRank. This is reflected in the type of interaction we should expect to ensure the optimal flow of work. Low Betweenness means that this team, typically, does not lie on the critical path to value, however, its high PageRank is indicative of its importance to the delivery of that value.

7.1.2 Complicated sub-system team with enabling team during mapping

The complicated sub-system team must act like a stream-aligned team where it is delivering value to the specific stream-aligned team being mapped. It must therefore act likewise with the enabling team, as noted above in 7.1.1, for the duration of a mapping workshop. However, interaction between these team types is going to be of less a priority and likelihood.

7.1.3 Enabling team with platform team during mapping

Outside of the Value Stream Mapping workshop, there is a brief period of "collaborative" interaction required between the enabling team and the platform team, where the platform team is providing the Value Stream Delivery and/or Management Platform. Where there is a high level of confidence in the value stream understanding of the platform team, this could be done in a "facilitating" interaction.

Both teams will have a high PageRank as they are key to the delivery of value by the value stream-aligned team, however, a platform team has a high Betweenness score and is therefore an influencer on the critical path to value. This means that the team has the risk potential for a negative impact on value delivery which must be mitigated. As such, any period of collaboration that can increase cognitive load must be extremely short lived and infrequent, but such periods are important to providing high innovation in a high performing team, whilst maintaining a high value flow between such periods.

7.2 Team interactions post-mapping

There must be a very clear understanding between all of these teams that whilst these interactions occur briefly for the mapping exercise, the target goal of the mapping exercise is to achieve the "typical" interactions between the teams, i.e. "x-as-a-service" to "Emphasize the User Experience" (In the case, the member of the stream-aligned team) and "Facilitating" to provide service management support for the stream-aligned Team.

Therefore, for the mapping period during a value stream implementation or change, the cognitive load on the impacted teams is adjusted as shown through Graph Theory in Section 7.3.

7.3 Cognitive load shown by Graph Theory during mapping

The relationship between cognitive load and team interactions is not immediately obvious, but we can establish a "fuzzy" relationship that acknowledges how much effort is required by a team when it takes on more or less cognitive load.

- Teams that work in isolation and do not utilize services, or take value from other teams, can be considered to be taking on the entire cognitive load to perform their work, but no additional cognitive load from other teams. These teams will need to use high levels of effort and can be considered to be "value hives."
- Teams that provide things of value as a service (x-as-as-service) take on considerable cognitive load in order to minimize the effort necessary in the consumers of their work. These teams are a high source of value second only to value hives.
- Teams that collaborate share the cognitive load. This sharing of load may be equal or in some proportion. Effort required by these teams is lower than that needed for value hives or x-as-a-service teams since it is shared across the team boundary.
- The remaining interaction style for teams is one of facilitation. In this type of interaction, the team sets guidance or provides tools for doing work but takes on less of the effort needed in the actual production of value. Cognitive load for these teams is still very apparent but the translation of that load is in enablement of the other teams to perform work.

We can represent this fuzzy relation of interaction style to cognitive load in the form of a cognitive slope as shown in Figure 7.1.

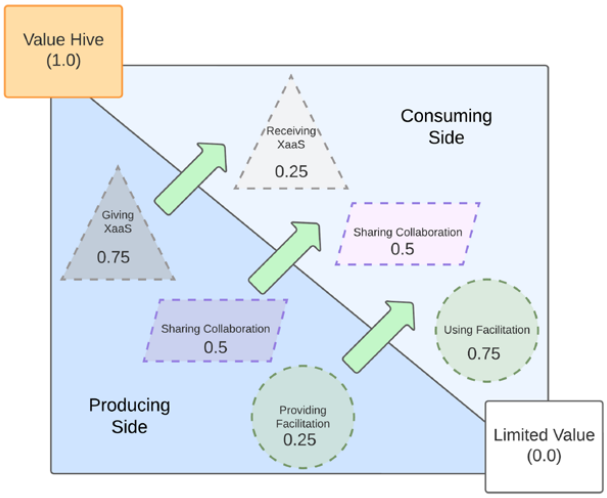


Figure 7.1 — Dual sided cognitive slope

It can be seen from the diagram for cognitive slope that we assign scores between zero and one. Higher scores are given for interaction styles that take on more of the cognitive load. Value hives score a unity value, whereas teams that offer limited or no value score zero. The other three interaction styles are evenly separated across the slope.

Using the cognitive slope relationship to team interaction style we can compute overall values for cognitive load in team organizations using a weighted graph. The edges between two nodes of the graph represent the interaction style that exists between two teams in the organization. These edges can be weighted according to the values assigned in the cognitive slope. The total cognitive load (a value of 1.0) between two teams is shared according to this slope and for each interaction style. For example, if a team provides facilitation then the cognitive load it takes on is 0.25. Conversely, this means that the team receiving the facilitation is taking on the remainder of the total load for that interaction. A team that receives help through facilitation will take on 0.75 of the cognitive load. This "two-sided" relationship of the cognitive slope needs to be taken into account when computing cognitive load across the organization and is represented by the complete cognitive load graph with edges between nodes that are in both directions.

In converting the cognitive load graph to an adjacency matrix we can compute average and total values for cognitive load in each team. In this matrix, teams are also assigned a value of 1 for internal interaction. This ensures that no team scores zero for average or summed cognitive load. The average value is computed using only non-zero values so that we only account for interactions (edges of the graph) that actually exist in either direction. The cognitive load scores for the general case of interactions between the four team types is shown in Table 7.1.

Team Topology	Average Cognitive Load	Summoned Cognitive Load
Stream-Aligned Team	0.625	2.5
Enabling Team	0.4375	1.75
Complicated Sub-System	0.625	2.5
Platform	0.8125	3.25

Table 7.1 — Cognitive load for the team topologies general case

It can be seen that in the general case of interaction between the four team types, platform teams typically have the highest cognitive load. This is not too surprising considering that they typically use the x-as-a-service interaction style that scores higher on the cognitive slope. Complicated sub-system teams and stream-aligned teams share the next highest score for cognitive load. The collaborative nature between these two teams is aligned with their equal score. There is a good balance between these two team types. Enabling teams take on the least amount of cognitive load in the general case. This does not mean that enabling teams are less busy than other team types, nor that they have less knowledge or skill. It speaks more to the efficient nature of this team type in their ability to provide enablement and facilitation to the other teams.

Later in this paper we will examine the relationship between flow, cognitive load, potential to impede flow, and the requirements that are placed on teams.

8. Value stream thinking: a Newtonian approach

The State of DevOps report helped to establish a correlation between practices and capabilities that drive high performance in technology delivery and organizational outcomes. Value stream thinking is becoming an important part of the strategy that teams are using to shift their practices and capabilities towards those established by the State of DevOps report. It is widely understood that correlation and causation are two different concepts that can exist at the same time, although one does not necessarily imply the other. Causation occurs when one variable affects another, while correlation simply implies a relationship between the two variables. We can move beyond correlation by applying scientific principles that hold fast and demonstrate causation in the face of correlated results. Value stream thinking may be enhanced by taking a Newtonian approach. We can postulate that there are three Newtonian-style laws that hold true for value stream thinking.

The three Newtonian laws of value stream thinking can be presented as:

- Value remains at rest or in uniform flow unless acted upon by a need
- The rate of change in momentum of flow is proportional to the need
- When a need is enacted on a body (team or system), then other needs will act simultaneously that are in opposition to the flow

These three laws can be explained as follows:

The needs that we place on the teams, often in the form of system requirements and user features, are the driving force that creates flow. The difference between a system that implements a particular requirement and one that does not establish the needs that will be placed on the team. This is pressure that we apply to the team to get work done. Flow will not happen, or will not change, unless there is a change of need.

The rate at which flow is measured is in units of value over time. There is a momentum of flow that is proportional to the need that is exerted on the team. A change in need will result in a change to the momentum of flow. The "Value Flywheel Effect" is a good analog to this second law. In the book of the same name, David Anderson, et.al.[20] explains that "As the flywheel starts to turn, each small win builds momentum. This continuous momentum builds situational awareness, breaks new boundaries, and catapults companies to sustainable, long-term success instead of short-term gains."

The third law is a little harder to visualize but can be explained in the following way: flow that happens across an organization or system does not happen without expending effort. This is because there are impediments to the flow that push back in the opposite direction. Teams need to overcome the constraints of flow to make progress. The effort they exert comes in the form of cognitive load. Impediments may come in the form of dependencies where value flows in opposition to those dependencies. Other impediments may also be apparent such as lack of resources or knowledge to perform a task. The list of impediments may not be finite. These impediments all represent other needs that act in opposition to the desired flow. If there were absolutely no impediments, then flow would be infinite. Since infinite flow is not attainable in the real world, then the logical conclusion is that impediments must always be present and always push back against the flow.

Newton's Laws are closely aligned with Ohm's Laws which model the flow of current in electrical circuits. Jeff Yee[21] established the relationship between these laws and showed through mathematical proof that despite the inability to measure each and every electron, Ohm's Law and Newton's Laws are related. We can expand on this hypothesis to create equations that explain flow in value stream thinking. Ohm's Law can therefore be considered as a good pattern for reasoning about flow.

20 The Value Flywheel Effect, David Anderson, Mark McCann & Michael O'Reilly. IT Revolution, Portland, OR. ISBN: 978-1-950508-57-0.

21 The Relation of Ohm's Law to Newton's 2nd Law, Jeff Lee, 2019. www.researchgate.net/publication/330639107_The_Relation_of_Ohm%27s_Law_to_Newton%27s_2nd_Law

Ohm's law states that there is a simple relationship between voltage, current and resistance:

Voltage (V) = Current (I) x Resistance (R)

$$V = I.R \quad [1.0]$$

We can establish a similar relationship for value streams:

Needs (N) = Flow (F) x Impediments (I)

$$N = F.I \quad [1.1]$$

Which we can also rearrange as:

Flow (F) = Needs (N) ÷ Impediments (I)

$$F = \frac{N}{I} \quad [1.2]$$

This means that flow (units of value over time) is proportional to the needs (size of the requirements or user features) and the impediments that oppose the flow. This is consistent with the three laws.

Ohm's Law also states a relationship between power, voltage and current:

Power (P) = Voltage (V) x Current (I)

$$P = V.I \quad [1.3]$$

Again, we can establish a similar relationship for value streams:

Effort (E) = Needs (N) x Flow (F)

$$E = N.F \quad [1.4]$$

Which we can rearrange as:

Flow (F) = Effort (E) ÷ Needs (N)

$$F = \frac{E}{N} \quad [1.5]$$

This also means that flow (units of value over time) is proportional to the effort used (cognitive load) and the needs (size of requirements or user features). This is also consistent with the three laws. The mathematical relationships above form the foundation of what we can call the "FINE Flow Equations."

As described in the introduction to this paper, our aim is not to provide a prescriptive way to identify accurate measurements, but in line with the general outcome of any science, our goal is to explain the greatest number of observable phenomena with the fewest number of principles. We use mathematical equations purely as a modeling tool to help understand how value streams and team organizations interact. For this reason we do not prescribe any standard units for flow, impediments, needs or effort, but instead show that calculations performed through these equations can be used to explain and reason about the relationship that exists between each of the four variables.

We recognize that software development, and working in software teams, is complex and that these complexities can make mathematical models impractical except in very narrowly defined contexts. Using the presented mathematical models of this paper requires a pragmatic engineering approach for effective decision making. Our aim is to provide context where the mathematical model can assist in the engineering process and in the organization of software teams.

The variables of the FINE model can be considered as the four degrees of freedom, or dimensions, that exist within a Value Stream Reference Architecture. These four dimensions are shown in Figure 8.1 and we examine each in more detail on the following pages.

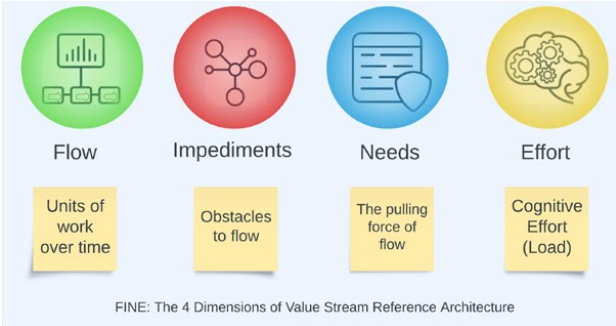


Figure 8.1 - The four dimensions of VSRA.

8.1 Flow rate (F)

As mentioned earlier, the rate at which flow is measured is in units of value over time. A unit of value is not prescriptive in this sense. A 'unit' may be a feature, a code commit, an added test, a user story, or even a defect fix. The definition of a unit is not necessary to understand the relationships. The important aspect to recognize is that these units are products of work that take some period of time to manufacture or complete. Flow is the only dimension within the equations that has, or is, a function of time. The unit of time in this function is also not prescriptive. It may be in seconds, minutes, hours, days, weeks or months. Like all of the variables of the FINE flow equations, flow rate is instantaneous. Only a change in one or more of the variables can cause a change to flow. It is intrinsically linked to all of the other degrees of freedom in the equations.

Understanding how flow is impacted by changes to the other variables of the equations allows us to reason about those other degrees of freedom and perspectives in terms of flow. Later in this paper we show how flow can be thought of as valuable and non-valuable units of work; good and bad flow. We will show the relationship between bad flow and the introduction of new impediments that create entropy in the system of flow.

8.2 Impediments (I)

Obstacles and impediments in flow are what slow the production of value down. These impediments can be presented to teams in many forms. They may be due to lack of resources, lack of knowledge, the ability to make (or take) decisions and risk, or the cognitive capacity to find a solution to a problem. One of the main impediments that teams can often face is when they need something from another team or have complex dependencies on other parts of the software system. Teams that become important to many other parts of the organization will have a much greater potential to impede the flow of work done by other teams.

Earlier in this paper we have shown how PageRank can be used as a way to measure the importance of a team within the connected graph of the organization. The value of PageRank can therefore be used as an indicator of impedance that may be presented by the team. In the FINE flow equations, increased impedance has a tendency to reduce flow, or otherwise require more needs and effort to restore flow status quo. Impedance in the FINE flow equations is non-prescriptive in terms of its unit of measure and is not a function of time.

8.3 Needs (N)

The needs given to a team represent the requirements and outcome of value that is expected from that team in cooperation with other teams. These needs may be presented as specifications and converted into acceptance tests that will validate when the outcome is achieved. Needs are the pressure, or pulling force, that causes the flow of work towards the end user. They are the "promises" that are made to the end user and pull the flow of work in the user's direction. In software systems we can postulate that these needs are the difference between an imperfect system and a perfect system. This means that needs are always present and constantly changing, and that perfection can never actually be obtained. Software teams are striving to make this difference as small as possible and it is the force of needs distributed across the organizational network that causes flow of work to pass through the teams as it heads towards its final destination. Teams that have a higher potential to impede flow are likely to see a higher differential of need at their own locality. This is because they are more important to the system as a whole so will find themselves being placed under greater need and pressure. Needs in the FINE flow equations is non-prescriptive in terms of its unit of measure and is not a function of time.

8.4 Effort/Cognitive Load (E)

Teams need to apply effort in order to get work done. This effort is the power they need to create inertia that makes flow happen. When effort is used for any period of time it requires cognitive energy. This energy can not be created nor destroyed but can only be converted from one form to another. A team of a particular size will have a maximum capacity for effort that is at the limits of their cognitive energy. In line with Fred Brooks's "Mythical Man-Month," adding more people to a team does not imply that this capacity can be increased. Indeed, adding more people to a software project may only delay it further.

In the physical world, potential energy is a type of energy that an object possesses due to its position or configuration. It is the energy that an object has as a result of its location within a field or system, such as the gravitational field or an electric field. For example, when an object is lifted off the ground, it gains potential energy due to its position in the earth's gravitational field. The higher the object is lifted, the more potential energy it tends to have. This potential energy only exists when the object is within the earth's gravitational field such that the pulling force between it and the object is apparent. The pulling force weakens as the object gets further away from the earth. Rocket scientists use escape velocity calculations to determine whether an object will remain within the gravitational sphere of influence of a given body. They understand that overcoming enough of the potential energy of a space object is important in establishing an orbit.

In terms of the human capacity for cognition, potential energy might be thought of as the energy that is stored within a person's mind due to their experiences, knowledge, and understanding. This mental potential energy can be transformed into action or thought, just as physical potential energy can be transformed into motion or work. A person, or team, that has spent a lot of time learning about a particular subject has a higher potential energy of knowledge and understanding in that area. We can state that their potential for cognitive effort is also higher. Just like in the physical world, this potential for energy is finite. There is a maximum cognitive effort that individuals or teams can exert for a certain period of time. The maximum cognitive effort that a team can handle is therefore constrained. In this sense, effort is the only dimension of the FINE equations that has a very real limit. Teams that can not overcome the pulling force of need using their own potential energy will find themselves not being able to escape the work. Overcoming the need with just enough of their energy is necessary to place the work "into orbit" outside of the organization.

A team that does not use all of its potential energy has capacity to take on more need or deal with additional impedance. A team where the required energy level exceeds the potential energy, i.e. cognitive load needed is greater than the cognitive capacity, will be producing toxicity and lower flow. They are likely to come "crashing back to earth". More importantly, the load that is placed on a team is determined by the parts of the system that they have responsibility for, and the interactions that they have to have with other teams. Teams that have high needs placed upon them, and are required to produce rapid amounts of work can expect to use a higher level of effort and cognitive load. Earlier in this paper we showed how cognitive load could be determined using Graph Theory by looking at the interactions between teams as defined by the cognitive slope. Cognitive load computed this way is a good approximation, and proxy, for effort that we may expect to see locally in the team. Effort in the FINE flow equations is non-prescriptive in terms of its unit of measure and is not a function of time.

8.5 The FINE flow equations

In common with Ohm's law, when looking at the FINE flow equations, we can show that there are two relationships and twelve derivations of these relationships in total. Taking the two flow relationships described previously, we can establish a complete set of derived equations that can be used to compute flow, impediments, needs, and effort. These 12 equations are given below.

F	I	N	E
Flow	*Impediments*	*Needs*	*Effort*
$F = \frac{E}{N}$	$I = \frac{N}{F}$	$N = F \times I$	$E = F \times N$
$F = \frac{N}{I}$	$I = \frac{N^2}{E}$	$N = \frac{E}{F}$	$E = \frac{N^2}{I}$
$F = \sqrt{\frac{E}{I}}$	$I = \frac{E}{F^2}$	$N = \sqrt{E \times I}$	$E = F^2 \times I$

Figure 8.2 — FINE flow equations

The FINE flow equations can be used to compute any unknown value from any other two known values. For example, if we were able to measure flow perhaps by the number of software deployments made by a team per week, and we knew the potential for impediments that a team has through its computed PageRank, then we could determine relative values for effort and needs using the equations:

Effort (E) = Flow (F) x Flow (F) x Impediments (I)

$$E = F^2 . I$$

And

Need (N) = Flow (F) x Impediments (I)

$$N = F . I$$

The Graph Theory presented in the previous sections of this paper demonstrated how values for cognitive load and potential to impede flow could be established for teams within a particular team topology. These values can be considered as equivalents for effort and impedance in the general system of flow. We can therefore use these values to compute flow and needs from the following equations:

Flow (F) = SQR[Effort (E) ÷ Impediments (I)]

$$F = \sqrt{\frac{E}{I}}$$

And

Needs (N) = SQR[Effort (E) x Impediments (I)]

$$N = \sqrt{E . I}$$

Using these equations we can compute flow values for the general interaction that occurs between the four team types. The results of these computations are shown in Table 8.1.

Team Topology	Flow	Impediments	Needs	Effort
Stream-Aligned	2.4498	0.1041	0.2551	0.625
Enabling	0.7084	0.8718	0.6176	0.4375
Complicated Sub-System	1.7192	0.2115	0.3635	0.625
Platform	1.3756	0.4294	0.5906	0.8125

Table 8.1 — FINE flow values for the generalized interactions in Team Topologies.

Some observations we can make from this analysis are that stream-aligned teams have the most flow, enabling teams will have the most need exerted upon them, platform teams will be required to use the most effort, and complicated sub-system teams find themselves evenly placed when it comes to impeding flow. Other observations are left to the astute reader to contemplate and discover for themselves.

8.6 Wardley Maps and the FINE flow equations

As discussed at the end of Section 5, the outputs from performing Graph Theory on an organizational structure could be transposed onto a Wardley Map to help identify where teams need to become more User delivery focussed or moved from a generative product to commoditised. There is also potential to use the same mapping technique to determine Continuous Improvement efforts, such as reduction of Effort on some teams.

For example, to map the 'x-axis' against the cognitive effort of each team, we can identify potential areas of psychological risk and determine the nature of what is required to change to reduce the amount of cognitive load. Likewise, a similar comparison could be used to look at the amount of Impediments, in particular to the relationship of Flow entropy discussed in the next section.

9. Flow entropy and the inspection and adaptation of value streams

An important aspect of any transformation is to understand what has worked better, what has degraded and where there are next opportunities to improve. As Plato once observed,

"Heraclitus, I believe, says that all things pass and nothing stays, and comparing existing things to the flow of a river, he says you could not step twice into the same river."

Alternatively, as many have observed, "the only constant in the universe is change." So then, we too must change with them or get left behind. The Value Stream Management implementation roadmap provides us with the capability to constantly inspect our position and adapt according to needs.

Likewise, *Team Topologies* identifies key points at which teams must evolve. These are identified as triggers. Firstly though, we should consider that there are two types of flow metrics we should consider: lagging and leading flow metrics.

9.1 Lagging flow metrics

DORA,[22] for example, are metrics that allow us to look at historical past performance, and through trend analysis identify potential areas for improvement. Continuing with the DORA example, these metrics are:

- Deployment Frequency
- Lead Time for Changes
- Change Failure Rate
- Time to Restore Service

However, there are other metrics related to flow to be considered:

- Cycle Time
- Throughput
- Work In Progress
- Code Quality
- Flow Efficiency
- Defect Distribution
- Flow Distribution

Many of these are defined in the State of Value Stream Management Report 2021. In all cases, these are extremely useful in showing how an organization has performed in the delivery of value. These are value measures on the efficiency of the delivery process. However, these measures are typically of the past and not an indicator of the performance to come.

9.2 Leading flow metrics

Using FINE, defined in Section 8, and applying the concept of Flow Entropy, we can start to create leading trend analysis charts to view the value generation of our value streams, in particular, the expected effort, or cognitive load, the flow and the need, as a hypothesis.

A leading flow metric that is of particular interest in VSRA research provides us with a measure of "stream alignedness" for a particular reference architecture. This metric is a simple ratio of the number of stream-aligned teams in an organization when compared to the total number of teams. We call this metric the stream-align-ratio, or "SAR" for short. For example, in an organization that has a total of 15 teams where 10 of those teams are stream-aligned, then a value for SAR would be 0.66 or 66%. We hypothesize that good values of SAR lie in the range of 0.5 to 0.8. That is to say that organizations where the numbers of stream-aligned teams are somewhere between half of the organization and 80% will be correlated with better performance. The authors of this paper consider this an active area of research that requires more formal validation.

22 DORA Metrics, cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devopsperformance

9.3 Flow entropy

In a system of flow, entropy is a measure of the randomness or unpredictability of that flow. In the physical world of fluid dynamics it can be applied to various types of flows, including those of gasses and liquids. It is the measure of the amount of thermal energy being transferred to or from a flow, and is related to the thermodynamic entropy of the system. It is typically calculated using the flow velocity, temperature, and other properties of the fluid. Flow entropy is an important concept in many fields that deal with the movement and transfer of energy, matter, or information. It can be used to understand and predict the behavior of various types of flows, and to design systems and processes that are more efficient and effective. In the world of information systems, we can use flow entropy to describe the randomness or unpredictability of the information in this context. Information system flow entropy is a measure of the amount of uncertainty or disorder present in the flow of information.

For software development teams, we can consider the flow entropy as being made up of the randomness of valuable or not valuable things; good or bad flow. There can be uncertainty in whether the flow will be good or bad. Good flow contains items of completed work that when presented to the end user will be considered as useful and valuable. Conversely, bad flow contains items of completed work that are not useful and not valuable to the end user. These bad flow items could come in the form of software defects, production incidents, technical debt, out-of-scope features, features that do not perform as expected, features that are not secure, or any other type of work that the end user finds unacceptable or unwanted. When items of bad flow make it to the production system we call them "escapes." These escapes in the bad flow are a major factor in causing friction for teams that ultimately slow them down.

We can state that there is a ratio that exists between bad flow rate and good flow rate. We can calculate this ratio simply by dividing the bad flow rate by the good flow rate thus:

Flow Ratio (f_r) = Bad Flow (f') ÷ Good Flow (f)

$$f_r = \frac{f'}{f}$$

where:

- **f'** is bad flow rate
- **f** is good flow rate

and:

- $\mathbf{f_r}$ is the flow ratio

The total amount of flow rate (F) is thus also given by:

Total Flow (F) = Bad Flow (f') + Good Flow (f)

$$F = f' + f$$

The curious and astute reader may also be wondering if flow ratio is the same as change failure rate (CFR) as used by the DORA metrics. Whilst these two measures are similar in their ability to define good versus bad flow, they are not quite the same. Flow ratio is computed using rates of flow (units of good and bad flow over time). Whereas change failure rate uses units that are not a function of time. Change failure rate is the amount of errors (defects) divided by the total number of changes (commits) in which the errors were caused. This means that whereas flow ratio can have any value between zero and infinity, change failure rate is confined to only values between zero and one. Change failure rate is the probability of errors occuring with change. Flow ratio is the odds of bad flow versus good flow. As we will soon see, this difference is important when considering how flow ratio is used with regards to the FINE flow equations and its ability to create impediments through flow entropy.

We can assert that the relationships between flow, impediments, needs, and effort are maintained when dealing with good and bad flow. The probability of bad flow reaching the end user is a function of how much quality that we build into the system of flow. Quality is a need. In the FINE flow equations we can consider that quality is synonymous with tests, which in turn is synonymous with specifications, requirements and need. They are fundamentally all the same thing.

This means that to lower the probability of bad flow, we have to place higher quality demands on the teams in the form of increased needs. The reason for this is because the bad flow ultimately results in extra rework that will slow the teams down. This slow down is caused by an increase in impediments. The new impediments are converted from energy of the increased effort needed to handle the bad flow that is now getting in the way of creating new flow (good or bad). The team is using some of its new energy and cognitive load to get past the impediment and fix the bad flow.

This is demonstrated within the FINE flow equations by:

Flow (F) = Needs (N) ÷ Impediments (I)

$$F = \frac{N}{I}$$

As Impediments increase, so the effort expended increases also. However, as the Flow of Work (F) increases, so the effort increases exponentially. In situations where production quality is deemed an issue, additional needs and expectations are set on teams to meet consumer needs. The net result is typically an increase in both the Flow of Work (F), to meet higher demand, AND an increase in Impediments (I) as quality controls are circumvented or risk assessed in order to meet the demands of Time to Value. Increases in both result in a huge increase on the Effort (E) required by teams and results in higher burn out rates and a turn over of key people.

We could dub this equation the "DevEx Equation" as it can be used as the greatest indicator of the factors that cause poor experiences for developers, a lack of psychological safety and high churn rates. In other words, high flow entropy equals high churn of valuable people.

Teams that try to avoid, or worse ignore, these new impediments create technical debt that over time will halt flow completely. This is like not repaying a monetary debt until you eventually default on the loan. Possibly one of the greatest examples of impediments to flow are production incidents, especially as they gain higher severity. These issues feed backwards into the development process requiring a higher level of immediacy and effort to resolve.

The diagram in Figure 9.1 illustrates how bad flow is converted and fed back into the development phase to impede new flow.

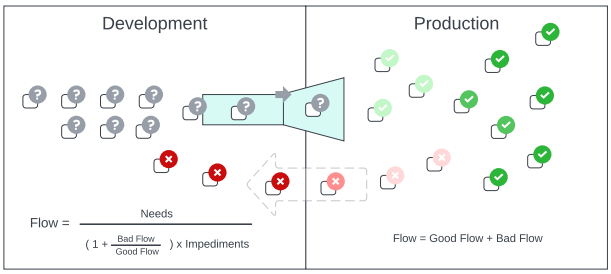


Figure 9.1 — The feedback cycle of flow entropy.

In her DevOps Enterprise Summit 2015 presentation "It's all about feedback,"[23] Elizabeth Hendrickson explains that placing quality late in the development process means that we speculate about whether or not the software is correct. She explains that speculation over time equals risk, and that the longer we speculate the more that risk grows. In systems where validation is left to the very last moment, or indeed where no quality is performed at all, then we have what Elizabeth calls a "Schrödinger's cat release." A thought experiment between physicist Erwin Schrödinger in 1935,[24] in a discussion with Albert Einstein, Schrödinger used a simple hypothesis involving a cat to illustrate what he saw as the problems of the Copenhagen interpretation of quantum mechanics. The main hypothesis was that when there is an equal probability of an event happening or not (the cat being killed or surviving), if you cannot observe any effect of the event, then we can state that the event both happened and did not happen concurrently. It is not until you can observe the effects of the event and so measure the result, that the probability wave collapses and you can determine if the event happened or if it did not.

23 DOES15 – Elisabeth Hendrickson – Its All About Feedback, DevOps Enterprise Summit, 2015. youtu.be/r2BFTXBundQ

24 Schrödinger, Erwin (November 1935). "The present situation in quantum mechanics". Naturwissenschaften. 23 (48): 807–812. en.wikipedia.org/wiki/Schr%C3%B6dinger%27s_cat

We can see that this same hypothesis can be considered in the context of the FINE flow equations. If good and bad flow have an equal probability of occurring, then the flow ratio; bad flow divided by good flow, will be equal to one. This means that the impediments that a team sees will double at each and every new instant that new flow is generated. If the probability wave collapses only when software is released to the end user then the dramatic climb in impediments will quickly bring the software development team to a grinding halt. Therefore, flow can not happen if there is no quality in place. Simply stated by the equations, this means that zero needs (no quality checks in the form of specifications or tests) results in zero flow. This can be easily explained by the formulae:

Flow (F) = Needs (N) ÷ Impediments (I)

$$F = \frac{N}{I}$$

Therefore:

Zero Flow (F_0) = Zero Needs (N_0) ÷ Impediments (I)

$$F_0 = \frac{N_0}{I}$$

The net effect of flow tending towards zero when no quality is applied, will always happen regardless of the impediments that are present. Even if there are zero impediments, without quality in the form of needs, flow is prevented. This would assume that code is being created without any thought for correctness, or test, even by the developer that is writing the code. Yet, in spite of this remarkable admonishment for the use of early quality, many teams still leave testing to the last possible moment, risking a Schrödinger's cat release, the opportunity to halt flow completely and the loss of valuable minds in the development process. This is why teams should "shift-left" in how they handle quality.

Another valuable technique that teams can use to control flow entropy is to use feature flags. These can be used as a "valve" that can throttle the amount of bad flow that is feeding back into the development process. Whilst this kind of "flow entropy valve" does not prevent the bad flow or lower flow ratio, it can by a very effective way to control the amount of bad flow that teams have to deal with at any instant. This technique lowers the severity of incidents caused by bad flow because service outages can be quickly dealt with. The effort (cognitive load) needed to deal with bad flow is lowered when feature flags are employed which in turn leads to a reduction in fear of bad flow. The increase in psychological safety introduced by feature flags means that more energy can be spent by the team in creating new flow. Flow rates are increased and impediments are better managed when feature flags are employed.

Other techniques to be considered for reducing flow entropy as a result of incidents is the use of Site Reliability Engineering and rapid reactive techniques such as observability and incident response.

As hinted in the previous diagram of Figure 9.2, we can model the effect of flow entropy by focusing on the change of impediments that happens from one instant of flow to the next using the following equation:

New Impediments (I_{t+1}) = 1 + Flow Ratio (f_r) x Impediments Now (I_t)

$$I_{t+1} = (1 + f_r) . I_t \quad [3.4]$$

where:

- $\mathbf{f_r}$ is the flow ratio (bad flow rate divided by good flow rate)
- It is the impedance at the current instance (t) of flow.

and:

- $\mathbf{I_{t+1}}$ is the impedance at the next instance (t+1) of flow.

It is important to mention again that only flow is a function of time; meaning that it is measured in units of value (good or bad) over time. This should not be confused with the ability to measure any of the other FINE flow dimensions at a single instance or point in time. The flow entropy equation given above demonstrates a value for impediments at time t and time t+1. This simply means that the time they are measured is not the same, and that the latter (I_{t+1}) is measured after the former (It). There is also no implication of what the time period between t and t+1 is. Effort is relative to the energy used when considered over a set period of time. More simply stated; energy can be measured as the effort used for a period of time.

With this model for changing impediments driven by flow ratio we can observe the effect of flow entropy for teams with different starting values of impediments and level of effort. The values for flow and needs can be calculated at each instant of time from impediments and effort using the FINE flow equations. For example, in the general case of interactions for team topologies, a stream-aligned team has an impediment value of 0.1041 and an effort value of 0.625. We can explore how flow ratio has an impact over a number of time instants (cycles). Table 9.1 shows what happens to the FINE flow dimensions over 20 consecutive cycles when there is a flow ratio of 10% bad flow to good flow (0.1). In this example we assume that energy (effort for the individual time period) remains constant from each instant to the next.

t	Flow	Impediments	Needs	Effort
1.	2.450273956	0.1041	0.255073519	0.625
2.	2.33624455	0.11451	0.267523363	0.625
3.	2.227521778	0.125961	0.280580871	0.625
4.	2.123858682	0.1385571	0.2942757	0.625
5.	2.025019798	0.15241281	0.308638958	0.625
6.	1.93078062	0.167654091	0.32370327	0.625
7.	1.840927089	0.1844195	0.339502854	0.625
8.	1.755255109	0.20286145	0.356073597	0.625
9.	1.673570081	0.223147595	0.373453139	0.625
10.	1.595686463	0.245462355	0.391680956	0.625
11.	1.521427347	0.27000859	0.410798453	0.625
12.	1.450624057	0.297009449	0.430849052	0.625
13.	1.38311577	0.326710394	0.451878298	0.625
14.	1.318749143	0.359381433	0.473933957	0.625
15.	1.257377972	0.395319577	0.497066128	0.625
16.	1.198862857	0.434851534	0.521327353	0.625
17.	1.143070884	0.478336688	0.546772741	0.625
18.	1.089875325	0.526170357	0.573460088	0.625
19	1.039155349	0.578787392	0.601450015	0.625
20.	0.99079575	0.636666132	0.630806097	0.625

Table 9.1 — Flow calculations for a stream-aligned team with a flow ratio of 0.1.

Charting the values from Table 9.1 provides a very clear and visual representation of the flow entropy that occurs when there are fixed levels of effort in the team. The energy they consume over the period is limited. We can see how flow is reduced over 20 cycles by approximately 60%. The impediments and needs of the team rise dramatically as this flow drops. Impedance is increased by 511% and needs are increased by 147%. This is shown in the graph of Figure 9.2.

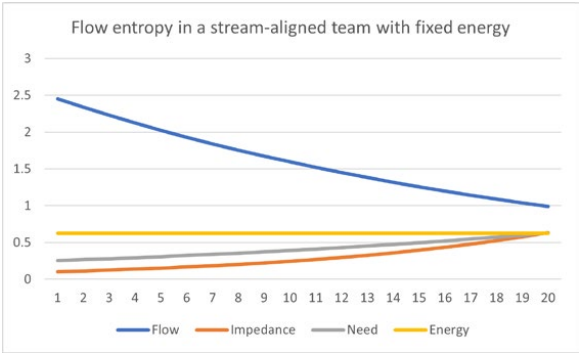


Figure 9.2 — Flow entropy with fixed energy.

We can of course assume that the cognitive load of the team is not constrained to a fixed value, and instead we can simply ask the team to "increase their brain power" at each instant. This can indeed cause flow to become constant but results in a dramatic load on the team as shown in the graph of Figure 9.3. In this case, both cognitive load and needs are now increased by 511%.

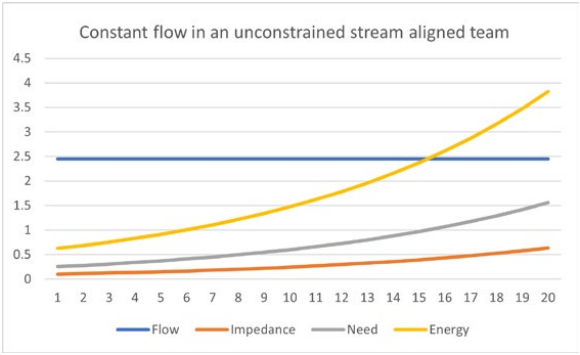


Figure 9.3 — Constant flow with unconstrained energy.

It is clearly infeasible to require teams to keep increasing their cognitive load in an attempt to maintain constant flow. There will come a point where the team reaches its maximum cognitive capacity and has no more energy to give. As stated earlier in this paper, adding more brain power to the team in the form of more people is also unlikely to raise the cognitive capacity to the extent needed. Indeed, adding more people creates additional burden that may slow the team down even further. When more people are added, new impediments arise such as lack of knowledge and the necessity to coordinate more work.

In our earlier assessment of cognitive load using Graph Theory, we asserted that the value for any single team could at most be a value of one. This is because the cognitive slope states that teams must be "value hives" once they reach this level. In this state, it supposes that teams take on all of the cognitive load without any outside assistance. The team has reached its maximum capacity for load on the cognitive slope. The graph of Figure 9.4 shows how flow entropy is re-introduced when a team reaches its maximum cognitive capacity. As with the previous example, this again uses a flow ratio of 10%.

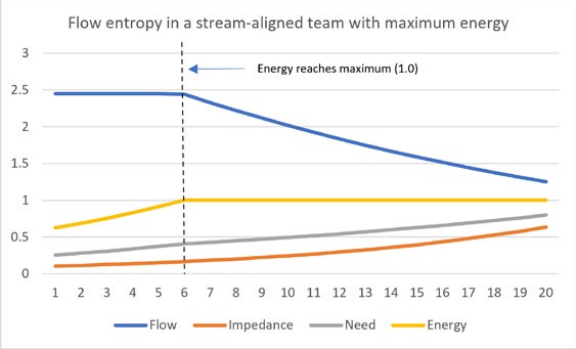


Figure 9.4 — Flow entropy after maximum cognitive capacity (energy for the period) is reached.

We can see from this graph that flow rates start to drop once a team reaches its cognitive capacity. In this example, flow is maintained until the sixth instant but then is quickly reduced by nearly half after 20 cycles. Impedance continues to rise as it did before reaching a 511% increase, but needs are slowed this time and in this example rise by 212%; less than half what they did when effort levels had no maximum. This lesser rise in needs is more reflective of what the team can actually achieve when there is a finite capacity for cognitive load.

9.4 Other formula of flow

It should be noted that we do not intend to imply in this paper that the FINE flow equations are the only way to reason about the flow of work within software teams. Many other applications of mathematical formulae have been proposed by other researchers to reason about such flow. Little's Law[25] for example, provides formulae that is useful for inspecting the average time work stays in a queue or system. Other theorems exist for computation of optimum batch size, throughput, probability of failure, cost optimization, and several others. Many of these techniques focus on the idea of queues and batches of work.

The FINE flow equations take a different approach where flow is viewed more as a continuum without finite items of work or effort. This allows a more fundamental view of flow that is not constrained by particular units of trailing metrics. We also believe that it presents a more parsimonious approach to reason about the dimensions that control flow, and provides more opportunity for the prediction of these dimensions that can be useful in decision making. We hope that other researchers will expand upon the FINE flow equations to discover other new insights.

9.5 Inspection and adaptation of value realization

The ultimate purpose of value streams and value stream management is business outcomes. As stated in Section 4, the purpose of any business service operating as a stream-aligned team should be ideally expressed in the form of a value statement and these value statements are typically based around the value you are required to realize. It is therefore imperative that those metrics are correctly measured and given the appropriate focus. It is not for this paper to define those metrics that are well covered in other pieces such as the State of Value Stream Management Report 2023. However, it is important to state that the correlation between flow metrics and value realization metrics should not be read as definitive. For example, the value statement "to increase customer retention by 20%" could be influenced by improving the change failure rate that in turn could improve customer perceptions of the business around quality and stability. However, this will not be the only factor impacting customer retention, and for every individual enterprise, the level of influence of this flow value to the realization value will differ widely. It is therefore the responsibility of each organization independently to determine how much flow metrics feed into or influence value realization metrics.

25 Little, J. D. C.; Graves, S. C. (2008). "Little's Law". Building Intuition. International Series in Operations Research & Management Science. Vol. 115. p. 81. doi:10.1007/978-0-387-73699-0_5. ISBN 978-0-387-73

10. Conclusions and recommendations

The implementation of a Value Stream Reference Architecture ensures that the primary requirements for a high maturity of DevOps transformation are met. The key requirement for Value Stream Management is business outcomes which are achieved by high flow of work from ideation to implementation through a stream-aligned team. These stream-aligned teams do not operate in isolation, typically being made up of multiple product aligned streams forming a business aligned group that incorporates wider enterprise functions beyond those of traditionally understood information technology. In turn, the high values of flow for stream-aligned teams are achieved by lowering impediments, maintaining levels of need with balanced cognitive effort, typically through supporting team types of enabling, platform and complicated sub-system teams.

Given the holistic nature for implementing the reference architecture, the inverse Conway maneuver must be applied to ensure optimized flow with "just enough" collaboration to meet the architecture required for the systems. Interactions and flows, therefore, should be simplified as much as possible between teams to match the required architecture types and to limit the problems of toolchain sprawl.

By using analytical methods such as graph theory, it is possible to map, model and measure those interactions, classify team types, and to identify other potential modes of operation using the FINE equations to optimize teams for flow, impediments, needs and effort. This also includes identifying peak moments for improvement of systems due to entropy in the form of defects, technical debt, incidents, legacy technology and other potential impediments that can be created over time to reduce the flow of work.

Using the most parsimonious principles as possible, the FINE flow equations help to explain how and why teams work the way they do. We try to do this using the greatest number of observable phenomena with the fewest number of principles, confirming intuitions, and revealing new insights. It has been shown that FINE equations allow for a new way to reason about organizational structure of teams and the impact that flow can have in the delivery of value.

It is the recommendation of this paper that by mapping the reference architecture and then applying the techniques presented herein with the FINE flow equations, it is possible to explore using leading metrics to identify potential continuous improvement opportunities ahead of any impact that could occur due to flow entropy. We welcome and encourage further research that builds upon the concepts presented in this paper to further the ideals and the promise of value stream architecture.

Want to learn more? Since you are reading this paper, you may already be aware of the Value Stream Reference Architectures community of practice. We encourage you to join the community (for free!) to engage in new research and discussion of VSRA. A proud member of the Value Stream Management Consortium, this community aims to establish a gathering space for practitioners of value stream management to share knowledge and experience that they have gained in the research and delivery of VSRA.

Find out more at: vsra-community.org

See you there and "Keep Flowing!"

About the Authors

Stephen Walters

EMEA Field CTO, GitLab

Stephen Walters has been in the IT industry for over 30 years and is an extensively experienced Subject Matter Expert in Value Stream Management, DevSecOps, DevOps, ALM, SDLC and IT4IT, with management & consultancy experience across end-to-end IT disciplines. Currently also operating as an Ambassador for the Value Stream Management Consortium & the DevOps Institute, he has an interest in all things DevOps. Certified in Value Stream Management, DevOps, SAFe, CMMI, ITIL, TOGAF and Prince2, Stephen is currently implementing leading edge thinking into Value Stream Management at GitLab to enhance the complete DevOps experience.

linkedin.com/in/1stephenwalters

Dr. Craig Statham

Chief Software Architect,
SAS Customer Intelligence Solutions

Dr. Craig Statham is the Chief Software Architect in the Customer Intelligence division at SAS Institute. He holds a Ph.D. from the UK's Liverpool John Moore's University and has been involved in software development for over 35 years. The majority of his career has been spent in senior management roles helping organizations and teams to develop cutting edge solutions to some of the most analytically demanding IT projects. He has worked across industry verticals including manufacturing and data science. A keen advocate for education, Dr. Statham has also served in an advisory capacity to National Academy Foundation accredited schools in helping educators bring forth the next generation of IT professionals.

linkedin.com/in/craig-statham

flowtopia