

The Momentum Guide

The guide to managing complex software delivery through collaboration



Draft – 1

Compiled by Vasanthakumar

April 2024

| | |
|--------------------------------------|----|
| 1. Introduction..... | 3 |
| 1.1. Why Momentum?..... | 4 |
| 1.2. Culture..... | 10 |
| 2. Characteristics..... | 11 |
| 2.1. Complexity..... | 12 |
| 2.2. Job Shaping..... | 13 |
| 2.3. Project Life Cycle..... | 17 |
| 2.4. Uninterrupted Time..... | 18 |
| 2.5. Theory O Governance..... | 20 |
| 2.6. Individual Personality..... | 21 |
| 2.7. Individual Psychology..... | 27 |
| 2.8. Motivation..... | 29 |
| 2.9. Principles..... | 30 |
| 2.10. Law of Mobility..... | 31 |
| 3. Scaling Setup..... | 34 |
| 3.1. Team..... | 34 |
| 3.2. Tracks..... | 35 |
| 3.3. Roles..... | 35 |
| 3.4. Configuration..... | 39 |
| 3.6. Meetings..... | 42 |
| 3.7. Collaboration..... | 45 |
| 3.8. Communities..... | 46 |
| 3.9. Team Formation..... | 47 |
| 3.10. Scenarios..... | 48 |
| 3.11. Implementation Guidelines..... | 49 |
| 4. Performance Management..... | 50 |
| 4.1. Collaboration Hub..... | 50 |
| 4.2. Community Hub..... | 50 |
| 4.3. Retrospective Hub..... | 51 |
| 4.4. Jobs Hub..... | 51 |
| 4.5. Product Value Hub..... | 52 |
| 5. References..... | 53 |

1. Introduction

Momentum is a software development methodology for creating and sustaining scalable software delivery initiatives. Momentum methodology emphasizes cultivating a supportive culture and features flexible communication and coordination policies to accommodate projects of varying sizes and complexity. Scalability and flexibility are inherent to the Momentum methodology.

In terms of principles, Momentum is closely aligned with the Crystal Method, while in terms of practices, it is closely aligned with Shape Up and Fluid Scaling Technology (FaST). Furthermore, Momentum adopts practices from Scrum, Kanban, Nexus, LeSS, as well as practices from a variety of methodologies. Despite altering many of its practices, Momentum preserves the core values of its adopted methodologies. It is a synthesis of principles and best practices from a variety of methodologies.

Rather than advocating for a one-size-fits-all approach, Momentum encourages a blend of methodologies tailored to suit the unique requirements of each project. If Momentum has seen further than others, it is by standing upon giants' shoulders. It is at Momentum that Scrum is applied in the right context.

In the current volatile, uncertain, complex, and ambiguous (VUCA) landscape, managing teams has become increasingly challenging due to rapid technological advancements, evolving employee expectations, transitioning work models, and shifting market dynamics. Momentum is designed to address these challenges head-on.

This guide contains the culture, characteristics, and scaling setup that support the Momentum methodology and it offers both prescriptive and descriptive guidelines.

It's important to note that while Momentum draws from various methodologies, it adapts and modifies them to suit its specific context. For instance, references to Scrum within Momentum are tailored to a semi-Scrum approach rather than strictly adhering to the traditional Scrum framework outlined in the Scrum Guide. Similar adaptations apply to other methodologies incorporated within Momentum.

Scaling with Momentum can introduce significant complexity. It's recommended to fully utilize Momentum when dealing with products requiring over 100 developers, intricate software patterns such as microservices, and complex delivery schedules. Otherwise, for projects of varying needs, it's advisable to selectively apply components of Momentum.

Instead of monitoring developers' work hours, user story size, or team velocity, the momentum methodology concentrates on creating an environment that supports developers in delivering outcomes.

1.1. Why Momentum?

Momentum is a versatile methodology designed to facilitate scaling, whether at the team level or product level. The following table illustrates where Momentum aligns within existing methodologies.

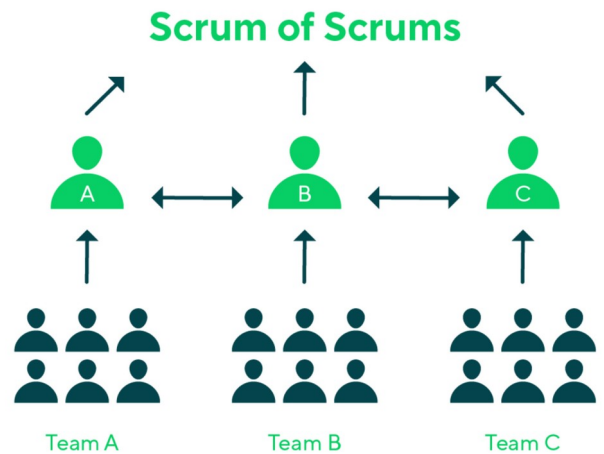
| Team Based | Product Based | Organization Based |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XP Scrum Kanban Scrumban Shape Up X-Team, DevOps Design Thinking Design Sprint Lean Startup | Scrum of Scrum Nexus Scrum@Scale Large Scale Scrum (LeSS) Spotify Model Kanban Scaling Scaled Agile & Lean Develop Fluid Scaling Tech. (FaST) Momentum | Management 3.0 Agile SHIFT Agenda Shift Agile Fluency Model OpenSpace Agility Agility Scales Humanocracy Sociocracy, Holacracy xScale Alliance Lean Software Development Enterprise Scrum Heart of Agile Brightline Transf. Compass Business Agility Crystal Method Agile Business Consortium FLEX- FLOW for Enterprise Goal Driven Agile (GDA) Modern Agile, Open Allocation Org Mindset, Helix Model Agile Leadership Toolkit GEAR Framework |
| Project Based | Engineering Based | |
| AgilePM Prince2 Agile PMI-ACP Project Half Double Dynamic Software Dev Adaptive Software Dev AgileDS PMP Prince2 Agile Unified Process Rapid Application Dev Critical Chain Project Mgt Agile-Waterfall Develop Hybrid Model | TDD, BDD, ATDD FDD, EDD Clean Code, Refactoring UX Design Agile BA, CI/CD Agile Modeling Specification by Example | |
| | Program Based | Portfolio Based |
| | AgilePgM (MSP) Praxis Framework X-Team Program Recipes for Agile Governance Disciplined Agile (DA) MoP | SAFe AgilePfM SfPfM E-B PfM Bimodal PfM P3O |

Despite numerous methodologies providing team scaling solutions, only a select few offer formal collaboration opportunities. Notably, Scrum of Scrum, Nexus, Scrum@Scale, LeSS and Momentum incorporate formal events to facilitate collaboration. In the subsequent sections, we delineate the advantages these collaboration events offer to developers.

Scrum of Scrum

The Scrum of Scrum virtual team comprises members from different development teams, aiming to coordinate their efforts effectively. Teams adopting Scrum of Scrum not only synchronize their deliveries but also strive for a fully integrated product by the end of each sprint. This approach facilitates quicker resolution of conflicting development goals.

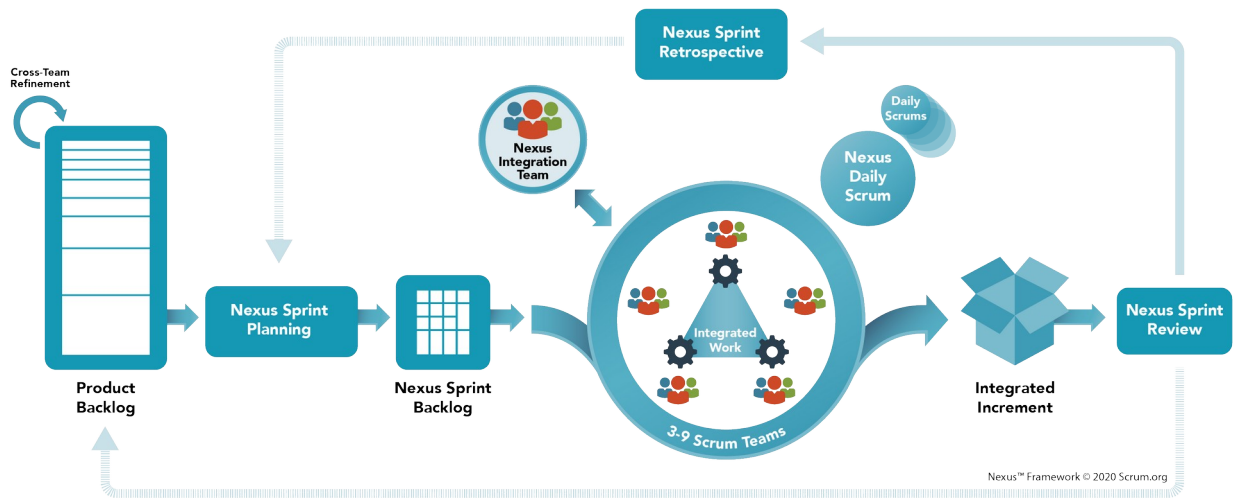
Team A, Team B, and Team C are represented in the following diagram along with their respective representatives, A, B, and C.



| Scrum of Scrum | Momentum |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| This scaling model is the simplest and provides the highest level of customization | A very complex scaling model that offers a great deal of versatility |
| Every day, representatives from each team meet for a Scrum of Scrum meeting. The frequency of meeting usually depends on the degree to which each development team relies on others | Ambassadors from each Momentum unit responsible for various forms of collaboration via chat, audio call, and video screen sharing |
| Representatives discuss their progress updates, coordinate their tasks during a sprint. It is usually an in-person group meeting | The purpose of the engagement is to assist with development. It is usually a one-to-one online meeting |
| This approach is mostly used by Scrum teams | Multiple methodologies are supported by Momentum |
| Prior to a Scrum of Scrum meetings, each team needs time to plan their upcoming sprint | The discovery team will identify the dependencies and type of collaboration needed before the planning meeting |
| The person who represents a team during the Scrum of Scrum can change over time. He can be replaced by another who is capable of representing the team and solve the issue at that point in time, including the Scrum Master | Anyone can be an ambassador because they are volunteers. It is essential that ambassadors have technical expertise in the product or a specific area of the product |

Nexus

A Nexus is a group of approximately three to nine Scrum Teams that work together to deliver a single product. Nexus help solve the dependency and collaboration challenges of cross-team work.

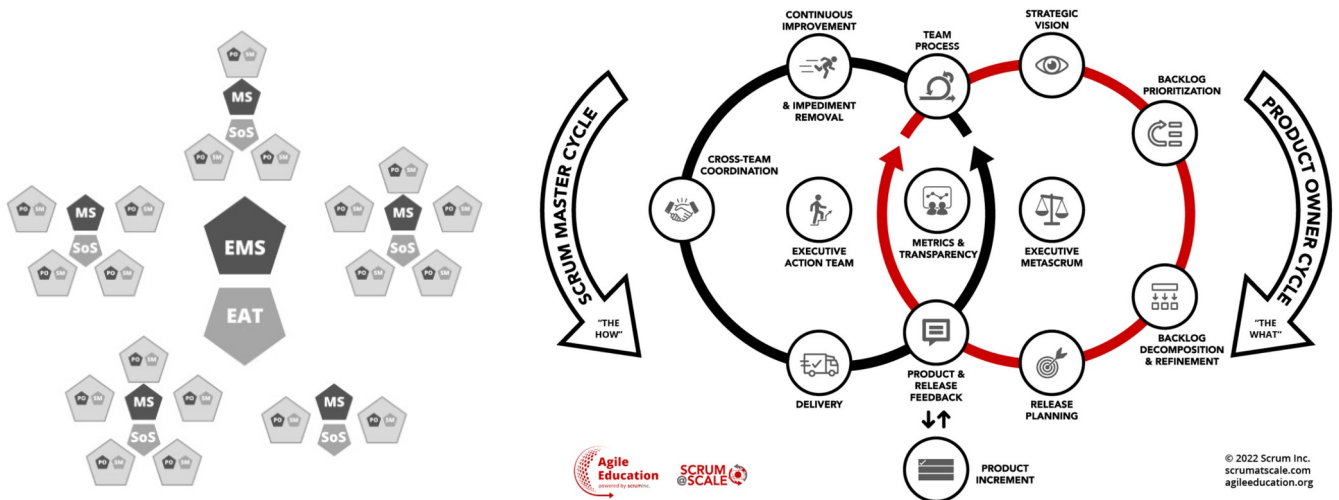


The Nexus Integration Team ensures that the Nexus delivers a Integrated Increment at least once every Sprint. The Nexus Integration Team consists of the Product Owner, a Scrum Master, and Nexus Integration Team Members. A Nexus Sprint Backlog exists to assist with transparency during the Sprint. The Integrated Increment represents the current sum of all integrated work completed by a Nexus.

| Nexus | Momentum |
|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Nexus integration team manages collaboration and dependency management | If an existing team doesn't meet the job requirements, a new team is formed or the existing team may be restructured |
| A complex task is divided into multiple components, with each team entrusted with the responsibility | Instead of fragmenting a complex task, a new team is established to collectively address and manage its intricacies |
| Regular Scrum meetings are supplemented by many meetings to achieve dependency management | The team consists of all the members required to complete the job, so there is no need for special meetings. |
| A Nexus sprint backlog is introduced for a better understanding of dependencies | Momentum strives to avoid dependencies |

Scrum@Scale

Scrum@Scale achieves linear scalability through a scale-free architecture. The Scrum@Scale framework has two cycles to assist teams in coordinating their work. Product Owner Cycle focuses on the “what” and Scrum Master Cycle focuses on the “how”. Scalability can be achieved by utilizing a Scrum of Scrum, a meeting attended by one person or representative from each team to communicate progress, impediments, and roadblocks.

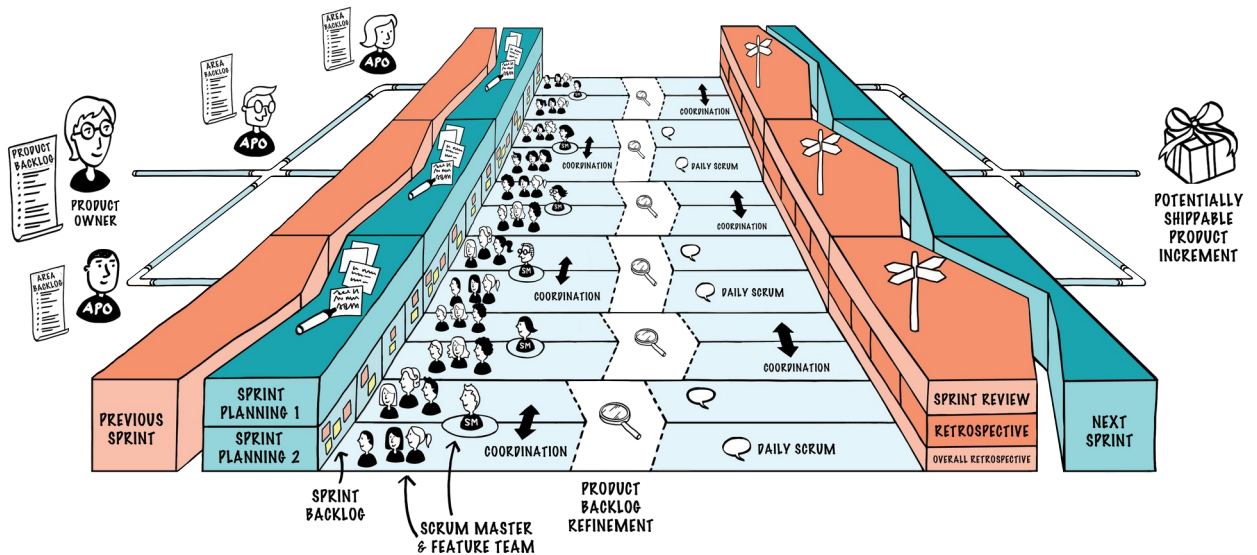


There are new roles introduced: Chief Product Officer – their part is to coordinate & collaborate with the different Product Owners to ensure a shared product vision and road map. Scrum of Scrum Master – their part is to ensure all Scrum Masters are aligned. Two new teams should be created and given the autonomy to succeed: Executive Action Team and Executive MetaScrum Team.

| Scrum@Scale | Momentum |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Chief Product Owner and Scrum of Scrum Master manage their respective cycles | Discovery team and development team manage their respective tracks |
| The Executive Action Team that accounts for an entire agile organization. Executive MetaScrum provides the forum for leadership & stakeholders to express their preferences | Discovery team review meetings provide opportunities for leadership and other stakeholders to express their preferences |
| Executive Action Backlog is introduced and additional events like Scaled Daily Scrum and Scaled Retrospective are also introduced | The retrospective hub manages process related issues. Various forms of formal collaboration engagement opportunities are provided by Momentum |

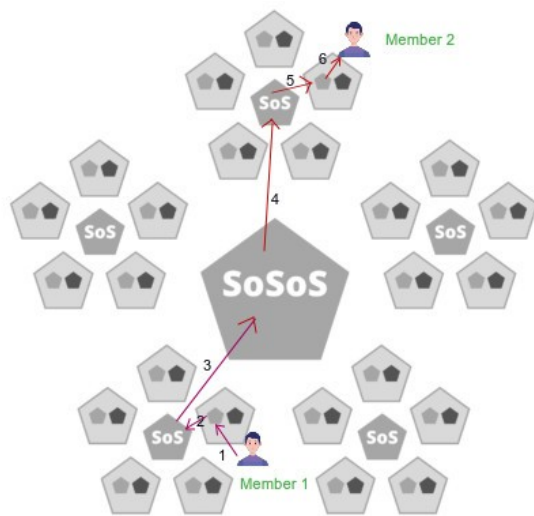
LeSS

LeSS builds upon Scrum to support its use in a larger context and how to scale it across larger organizations and beyond the one team. LeSS comes in two configurations: Basic LeSS for two to eight teams (10-50 people) and LeSS Huge for more than eight teams (50-6000+). A new role - the Area Product Owner (APO) - and additional artifacts and meeting changes are included in LeSS Huge.



| LeSS | Momentum |
|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| There should be one Product Owner for the entire LeSS group and he is accountable for the product success | Everyone is responsible for the success of the product, and there is no product owner role |
| Every product area has its own Area Product Owner | Each discovery track should have at least two members of the discovery team |
| Sprint planning has two parts | Cycle planning has three parts |
| The group works from a shared single backlog for the entire product | There are multiple product backlogs |
| All teams work in a synchronous sprint, focusing on the entire product | Team may be involved in overlapping cycles with other team |
| There is a single definition of done for all teams in the group | There are should-haves and nice-haves for every job documents |

Momentum

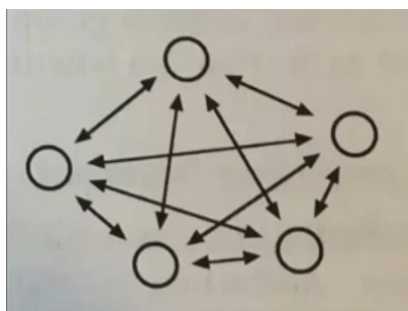
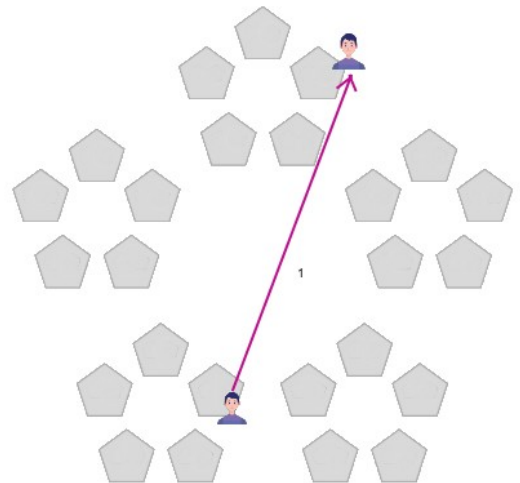


Scrum of Scrum, Nexus, Scrum@Scale, and LeSS achieve scaling by extending specific aspects of the Scrum framework. Collaboration between developers typically occurs through a broker model, often facilitated by the Scrum Master, Product Owner, or another developer.

For instance, in a Scrum@Scale setup involving 25 teams, if two developers need to clarify technical details, it would necessitate six interactions.

In Momentum, scaling is facilitated through the introduction of a novel role called ambassador, eliminating the need for intermediaries in collaboration. This streamlined approach allows for instant and direct communication among team members. Additionally, the model can be effortlessly extended to accommodate unlimited teams, spanning across various functions such as leadership, sales, and customer support teams.

For instance, in a 25-team Momentum setup, if two developers need to clarify technical details, it only requires one direct interaction, significantly reducing communication overhead.



The diagram illustrates how each microservice calls upon others, highlighting the intricate inter-dependencies inherent in microservices architecture. With Momentum, teams can navigate these complexities with agility and responsiveness, ensuring efficient development and seamless integration of microservices.

For a microservices project requiring continuous and immediate collaboration due to interdependent microservices, a software development methodology that supports such interaction patterns is crucial. The Momentum methodology, with its emphasis on instant and direct communication facilitated by ambassadors, is well-suited for this scenario.

1.2. Culture

Momentum holds the belief that developers can surpass their scheduled objectives and deliver enhanced value when operating within a positive organizational culture. The success of initiatives under the Momentum methodology hinges upon the presence of a collaborative culture and a flat organizational structure.

- The first-order effects of performance are people, communication, and talent
- There is no management role within the organization like Scrum Master or Manager
- There is no status quo role, such as that of a Product Owner
- Three levels of hierarchy are recommended: the CEO, leaders, and team members
- By eliminating endless meetings, the team saves time and energy
- A reward-free and punishment-free approach
- A culture of end-to-end automation and JIT working documentation
- Instead of counting velocity, the team focuses on product value
- Manage collaboration engagement rather than the story point
- It is not allowed to run a meeting for more than 60 minutes
- All meetings can be attended online by participants

To use Momentum effectively, people need to become aware of the following values:

- Humans are intrinsically drawn to learning and exploration
- Embracing empathy and being intelligent are two sides of the same coin
- Environment shapes our behaviour
- Work shrinks or expands based on developer interest
- Motivation comes from within when we understand why we are doing what we are doing

2. Characteristics

Understanding the following factors is crucial for comprehending, implementing, and adapting Momentum's practices in managing the software development process.

- Complexity
- Job Shaping
- Project Life Cycle
- Uninterrupted Time
- Theory O Governance
- Individual Personality
- Individual Psychology
- Motivation
- Principles
- Law of Mobility

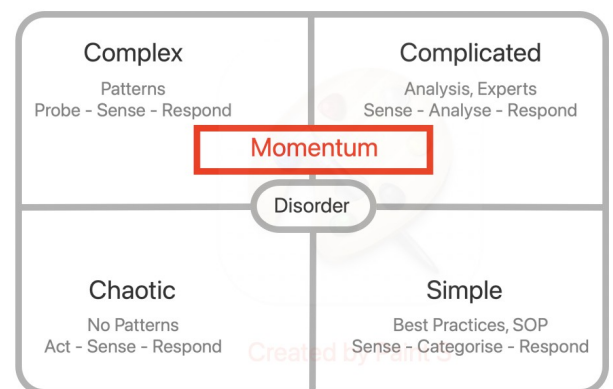
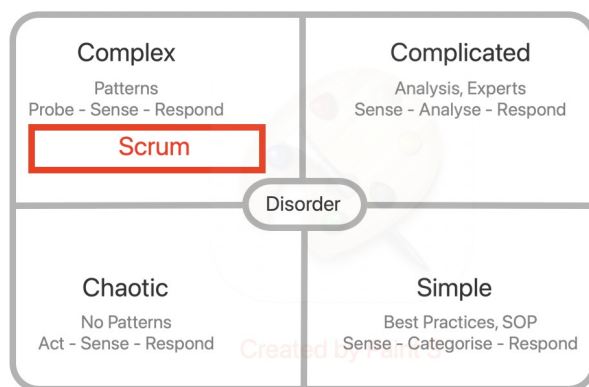
2.1. Complexity

Cynefin theory delineates five sense-making domains: Simple, Complicated, Complex, Chaotic, and Disorder. When engaging with organizations, we focus on understanding the Complicated and Complex domains.

In the Complex domain, the outcome of actions is uncertain, making it challenging to predict which actions will yield desired results. To navigate this terrain effectively, it's essential to probe the environment, observe the effects of actions, and then select appropriate responses. Conversely, certain aspects of organizational contexts may be relatively Complicated and amenable to analysis.

Scrum methodology is particularly adept at managing complex adaptive problems. In software development, some components fall within the Complex domain, while others fall within the Complicated domain.

The Momentum system acknowledges and responds to both the complex and complicated aspects of a task, adapting its approach accordingly.

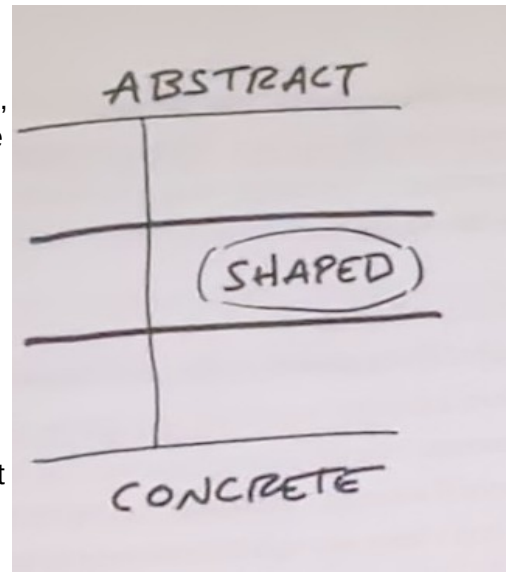


| Scrum | Momentum |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| Provide adaptive solutions to complex problems in order to generate value | Assesses complex and complicated problems and treats them accordingly |
| Suitable for loosely coupled complex problems | It is most suitable fit both for loosely coupled complex problems and tightly coupled complicated problems |
| Perform poorly if the complexity is outside the team's control | Ability to perform at a team, product, and organization level |

2.2. Job Shaping

In Momentum, every idea or requirement undergoes a shaping process before being passed to the development team. This process entails upfront efforts to set boundaries, address unresolved questions, minimize risks, and mitigate uncertainties. The output of the shaping process is a job document.

The requirement should be shaped at the right level of abstraction: not too vague nor too concrete. It is common for product managers to err on one of these two extremes. In wire-frames, concepts are too concrete, leaving little room for creativity on the part of developers or designers. Words are too abstract and under-specified work grows out of control without boundaries.



Shaping has 5 main steps that we cover here.

- **Set Boundaries**

We figure out how much time the raw idea is worth. This gives us the basic boundaries to shape into.

- **Rough out the elements**

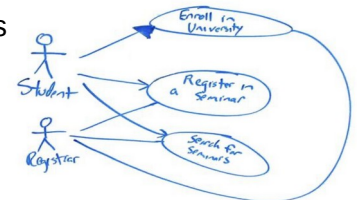
Then comes the creative work of sketching a solution. We do this at a higher level of abstraction than wire frames in order to move fast and explore wide enough range of possibilities. The output of this step is an idea that solves the problem within the time but without all the fine details worked out. This way the developers retain the creative process. Among other approaches, the following practices can help.

- **Mental Model Practice**

Create a mental model of a proposed solution to the problem. Mentally execute the model to see if it solves the problem, and provide samples for the model to produce correct results. When sufficient sample inputs have passed the test, assume the model to be a suitable design model and begin representation of the design

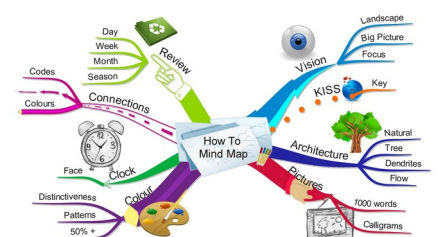
- **Conversation Model Practice**

Develop a simple conversation design model. It facilitates a comprehensive understanding of how different solution components interact. As a result, it brings flexibility by being open to change and improvement. Furthermore, it facilitates collaboration among stakeholders and establishes a feedback loop.

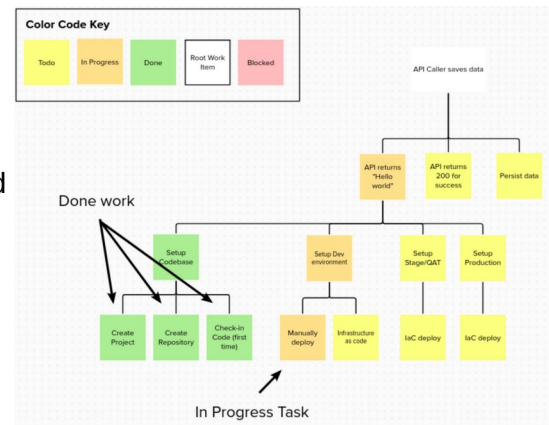


- **Mind Map Practice**

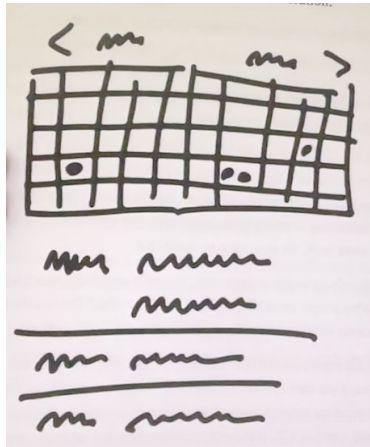
A mind map is a diagram that outlines the ideas in a visual format. The process facilitates the generation of new ideas, simplifies complex ideas, and improves the collaboration process.



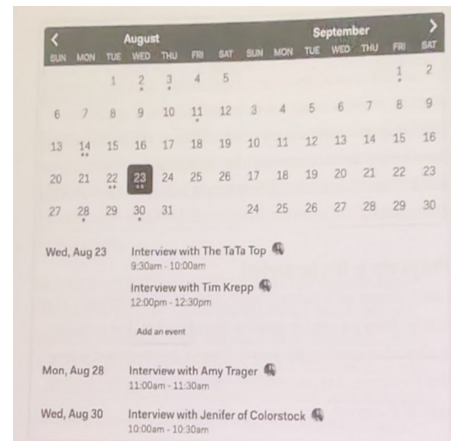
- Discovery Tree practice
Feature mapping is a tree of tasks required to complete the feature. Discovery Trees helped visualize work, track it, and create a shared context and understanding. The discovery trees really helped with Fluid Teaming as an incoming team member could very quickly get a picture of the work and the context it sat in. A simple and natural way of story creation, project tracking and shared ownership



The level of fidelity we used to define the solution is shown below. This is a rough sketch of the Dot Grid concept



The finished product that was created by the designers looked like this at the end of the cycle. Here is a screenshot of the Dot Grid when it was first launched








- Address risks and rabbit holes**

Once we think we have a solution, we take a hard look at it to find holes or unanswered questions that could trip up the team.

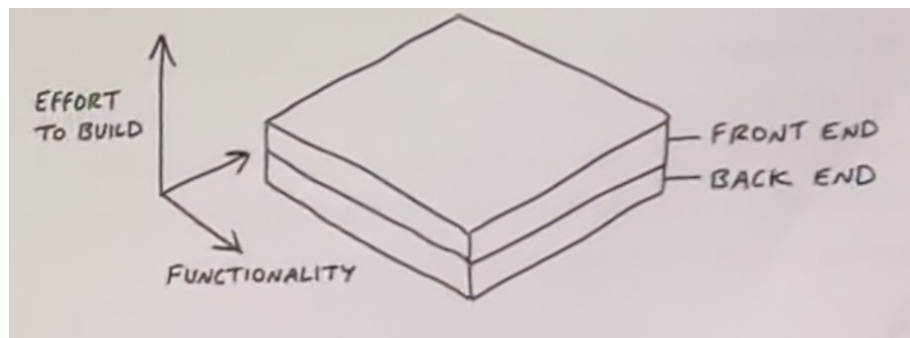
We amend the solution, cut thinks out of it, or specify details at certain tricky spots to prevent the team from getting stuck or wasting time. We also include “must have” and “nice to have” attributes. We can also address dependencies, never tried out solutions, spike opportunities.

- Recognizing the job skeleton**

The job scope and nature is recognized in this step. This job skeleton decides how the developer is going collaborate and work together to approach the problem. There are 5 types of job skeletons.

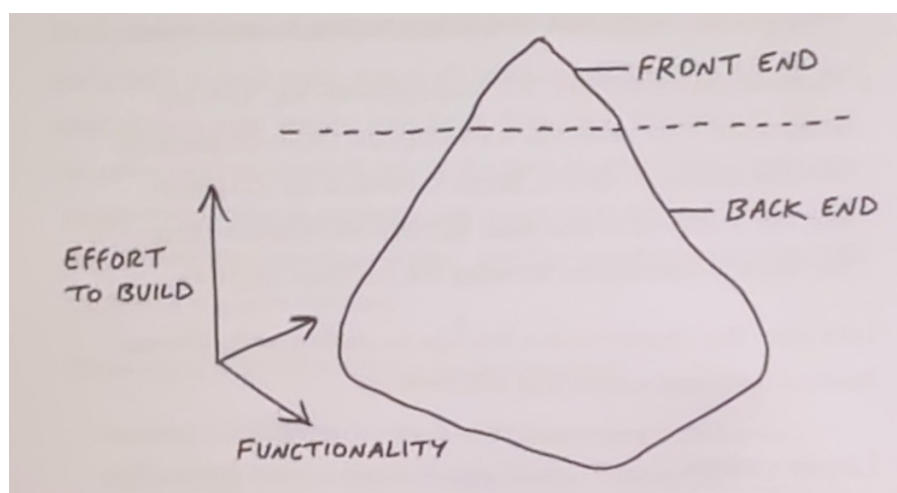
| | | |
|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Small  | Medium depth  | Large depth  |
| Medium breadth  | Large breadth  | |

Most software project require some UI design and a thin layer of code below. Work like this looks like a layer cake: you can judge the work by UI surface area because the back-end work is think and evenly distributed. This may be fall under medium breadth or large breadth depending on the scope.



Sometimes there is significantly more back-end work than UI work or vice versa or more complex work required between front-end and back-end. This kind of work is like an iceberg. This may be fall under medium depth or large depth depending on the scope.

These skeletons define the job scope and nature.



- **Write the Job**

Once we've shaped it enough, we package it with formal write-up called a job document. The job summarizes the problem, constraints, solutions, dependencies, rabbit holes, should-have features, nice-to-have features, skeleton and no-gos.

| Scrum | Momentum |
|-------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The Product Owner creates the user story | The discovery team members create the job |
| The Product Owner may simply forward the customer's request | A requirement is properly shaped into a job by members of the discovery team |
| It is possible that the Product Owner lacks technical skills | Technical skills must be possessed by at least one member of the discovery team |
| The team is a first-class citizen | The job is a first-class citizen |
| Scrum never split the team | Momentum never split the job |
| The product backlog item goes to the team | Developers who are interested in the job pick it |
| A product backlog item is usually a user story finalized during sprint planning | Feature jobs typically undergo scope hammering throughout the development cycle |
| To achieve progress, Scrum believes that dividing features into user stories and assigning them to teams is the best approach | Momentum believes it is important for the whole feature to be shaped properly before passing it on to developers, so the developers have autonomy and motivation |

2.3. Project Life Cycle

Momentum understands that every project is different and evolves over time, so the methodology must also be tuned and modified. There are several phases in a project, and each phase calls for a different methodology.

A table illustrating various phases and the methodologies that are most appropriate for them can be found below.

| Project Phase | Envision | Risk Mitigation | MVP | Production | Maintenance |
|---------------|-----------------|-----------------|----------|------------|-------------|
| Methodology | Design Thinking | Scrum | Shape Up | Momentum | Kanban |

- Design thinking provides a structured process that helps innovators break free of counterproductive tendencies that thwart innovation. It is social technology that blends practical tools with insights into human nature. It is a good suited for customer discovery, idea generation, and testing experiences.
- Scrum generate values through adaptive solutions for complex problems. It is iterative and incremental. As a result of its nature, it can be used for concurrent development.
- Shape Up delivers value through the shaping process, scope hammering through the cycle and providing uninterrupted time for developers. Due to its nature, it can be used for plans that do not change frequently.
- The Kanban method is iterative, but not necessarily incremental. As a result of its nature, it can be used for frequent plan changes. Kanban has planning meetings only on demand and no special meetings about process. Kanban visualizes project workflow, spotlights bottlenecks the day they occur, and forces team members to immediately resolve the issue or swarm to fix it.
- Momentum brings the benefits of Scrum, Kanban, and Shape Up.

| Scrum | Momentum |
|--------------------------------------------------|--------------------------------------------------------------------|
| Suitable for risk mitigation | The best fit for the production phase |
| Assume one model is applicable to all situations | Utilizes best practices across all methodologies |
| Changes to roles or meetings are not allowed | Modifications to roles, tracks, teams, and meetings are encouraged |

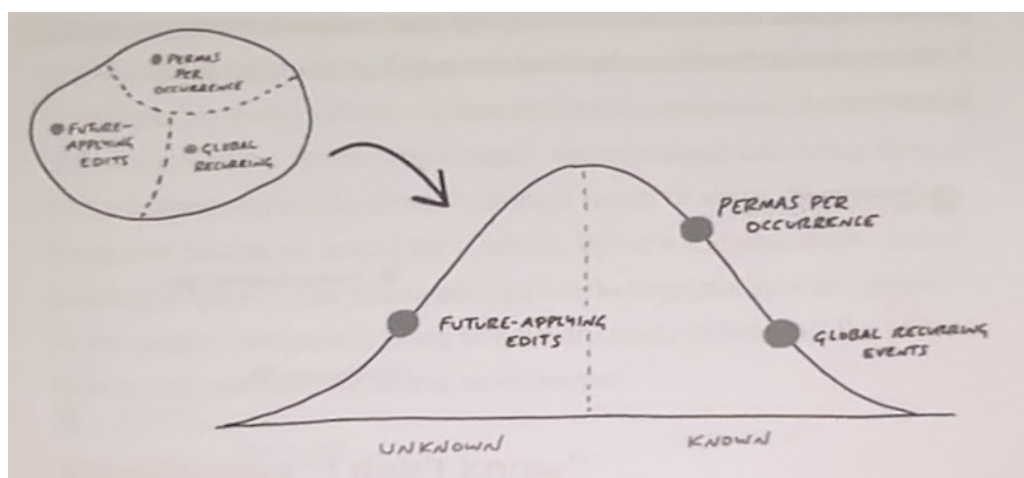
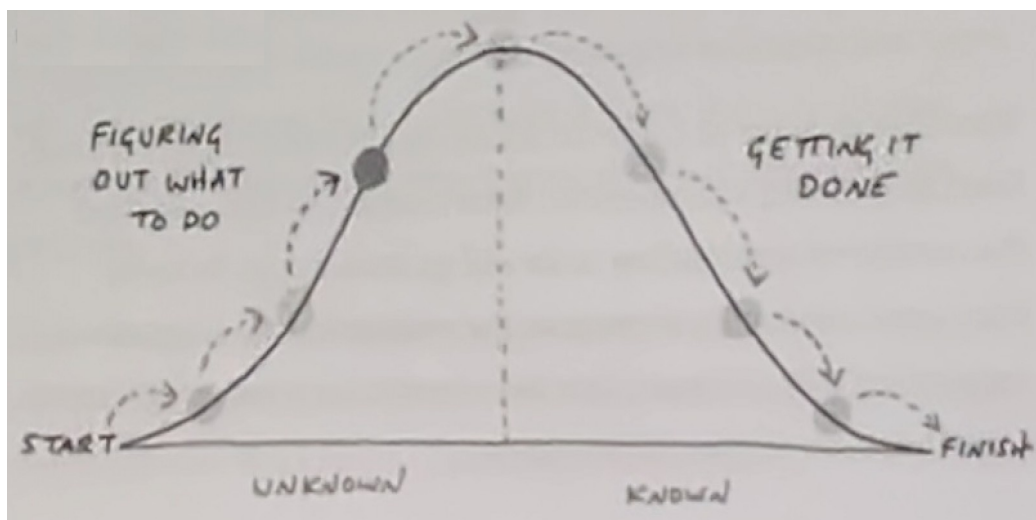
2.4. Uninterrupted Time

Team members are not allowed to be interrupted or distracted. Interrupting the team with requests breaks the commitment. Progress and momentum are second-order things, like growth or acceleration. They can't be described by a single point. You need an uninterrupted curve of points.

Taking someone away for one day to fix a bug or help another team isn't just a waste of time. As a result, you lose the Momentum they built up and it will take time for them to regain it. When you lose the wrong hour, it can ruin your entire day. It is possible to ruin a week by losing a day.

Work is like a hill

Every piece of work has two phases. There is the uphill phase of figuring out what to do and the downhill phase of doing it. The following hill chart shows the uphill and downhill work.



Status without asking

Having the full context of where the work stands, team members intuitively drag scopes into position and save new updates to the feature. These task updates are provided once every few hours by developers.



| Scrum | Momentum |
|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| The purpose of the daily Scrum is to check the status | The Daily Scrum is used for technical strategy, tactical planning, and problem solving |
| All team members must participate in the daily Scrum & this interrupts work flow | Only developers facing complex work should participate in the daily scrum |
| Product backlog refinement event interrupts the flow | There is no product backlog refinement meeting |
| Daily Scrum events are used by developers to share updates with the team | Hill charts are used to share the updates with the team |

2.5. Theory O Governance

The brutal fact is that about 70% of all initiatives fail. To effect successful change, first grasp the 2 basic theories of change: Theory E and Theory O. Theory O principles guide Momentum practices.

- **Theory E**

Theory E change strategies usually involve heavy use of economic incentives, drastic layoffs, downsizing, and restructuring. Shareholder value is the only legitimate measure of corporate success.

- **Theory O**

Theory O change strategies are geared toward building up the corporate culture: employee behaviors, attitudes, capabilities, and commitment. The organization's ability to learn from its experiences is a legitimate yardstick of corporate success.

| Dimensions of Change | Theory E | Theory O |
|----------------------|--------------------------------------------------|---------------------------------------------------------------|
| Goals | maximize shareholder value | develop organizational capabilities |
| Leadership | manage change from the top down | encourage participation from the bottom up |
| Focus | emphasize structure and systems | build up corporate culture: employees' behavior and attitudes |
| Process | plan and establish programs | experiment and evolve |
| Reward System | motivate through financial incentives | motivate through commitment—use pay as fair exchange |
| Use of Consultants | consultants analyze problems and shape solutions | consultants support management in shaping their own solutions |

2.6. Individual Personality

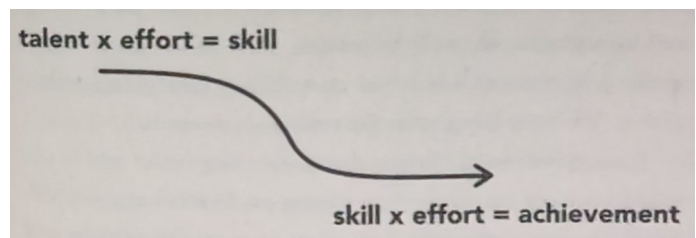
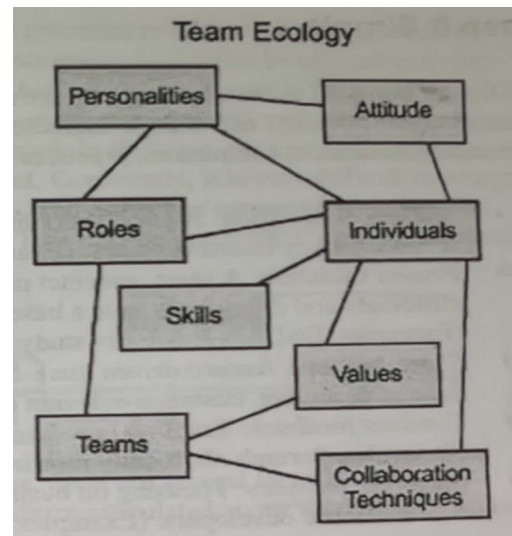
The leader must recognize the unique personality, strength, talent, skill, grit, grown mindset, potential capabilities, learning style, working style, roles, preferences, values, projects & workflow, inclusion & fairness, dedication to fulfilling commitments and collaborative techniques of team members.

People are stuffed full of personality. It varies by the time of day, by the age, by the culture, by the temperature and by who else is in the room.

A person's style and chemistry are important factors in their relationship. Myers & Briggs, DISC Mental Models test gives personality assessment.

People perform better when their job assignments are aligned with the strengths of their personalities, not their weaknesses.

The methodologies name the roles that must be present on a project, but do not specify the personality characteristics each role requires.



| Working style | Coaching approach |
|--------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Analyzer - He requires extensive information before taking on a task, and he hates making mistakes | <ul style="list-style-type: none"> • Providing him with ample classroom time • Engaging him in role-playing • Providing him with time to prepare for challenges |
| Doer - He uses trial and error to enhance his skills while grappling with tasks | <ul style="list-style-type: none"> • Giving him a simple task, explaining the desired outcome, and getting out of his way • Gradually enhancing a task's complexity until he masters it |
| Watcher - By watching others, he develops his skills | <ul style="list-style-type: none"> • It would be beneficial for him to shadow top performers |

| | |
|------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Collaborator - He enjoys working with others | <ul style="list-style-type: none"> Recommend concurrent and pair programming to him |
| Lone Wolf - He prefers working alone | <ul style="list-style-type: none"> A good fit for complex, deep-focus programming work Develop a plan that rarely changes and place him on the development track |
| Explorer - He prefers to solve different problems every day and is curious about discovering new things | <ul style="list-style-type: none"> Providing him with research and development assignments Place him on the discovery track or the development track where the plan is likely to change frequently |
| Teacher - He prefers to share his knowledge with others | <ul style="list-style-type: none"> Assign him to the role of ambassador |
| Learning Mode - Recognize how one learns – as a reader, a listener, a writer, or a speaker | <ul style="list-style-type: none"> Using feedback analysis can help you find out |

Skill Acquisition Level

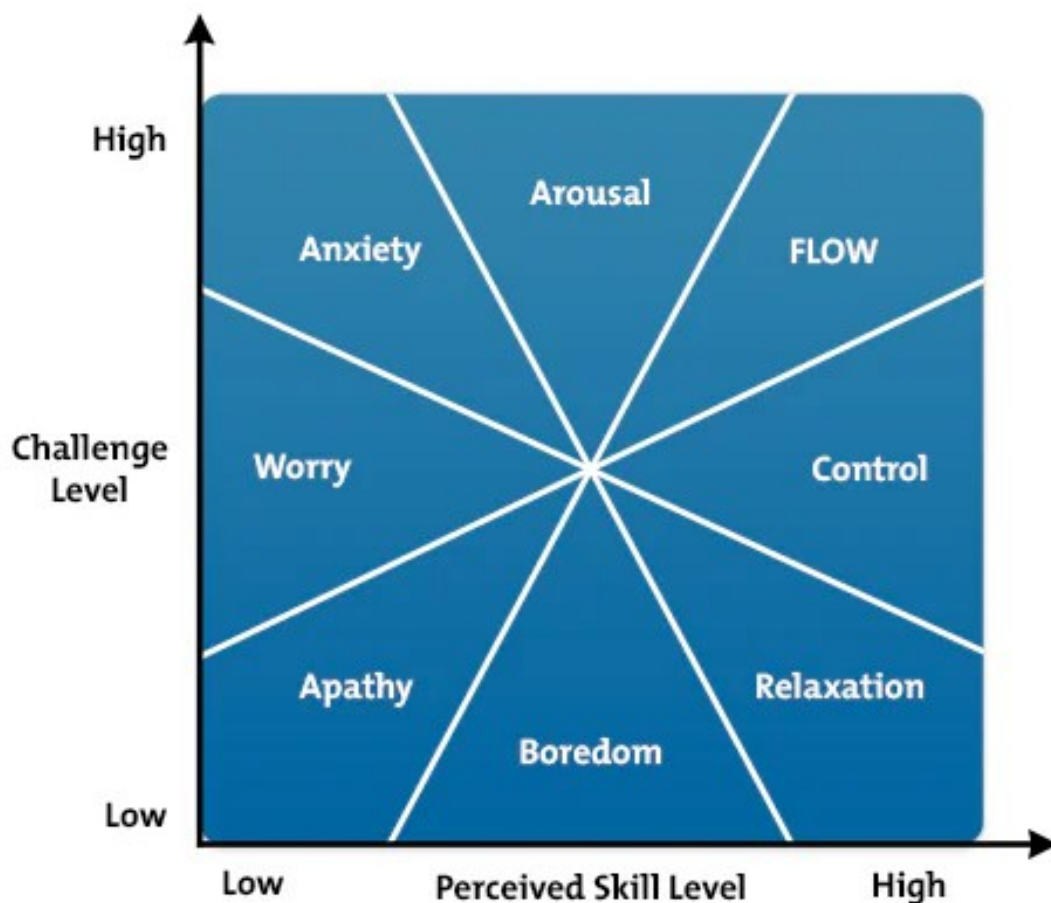
The Dreyfus model of skill acquisition is a model of how learners acquire skills through formal instruction and practicing, used in the fields of education and operations research. The team member involved in discovery track should possess a proficient level of skill.

| Mental Function | Novice | Adv. Beginner | Competent | Proficient | Expert |
|-----------------|-----------------|---------------|-----------|------------|----------|
| Recollection | Non-Situational | Situational | | | |
| Recognition | Decomposed | | Holistic | | |
| Decision | Analytical | | | Intuitive | |
| Awareness | Monitoring | | | | Absorbed |

The Flow Model

The Flow Model shows the emotional states that we're likely to experience when trying to complete a task, depending on the perceived difficulty of the challenge, and our perceptions of our skill levels.

For example, if the task isn't challenging and doesn't require a lot of skill, we're likely to feel apathy towards it. But facing a challenging task without the required skills could easily result in worry and anxiety. To find a balance, and to perform at our best, we need a challenge that is significant and interesting, and we need well-developed skills, so that we're confident that we can meet the challenge. This moves us to a position where we can experience "flow" (being totally involved and engaged in the activity).

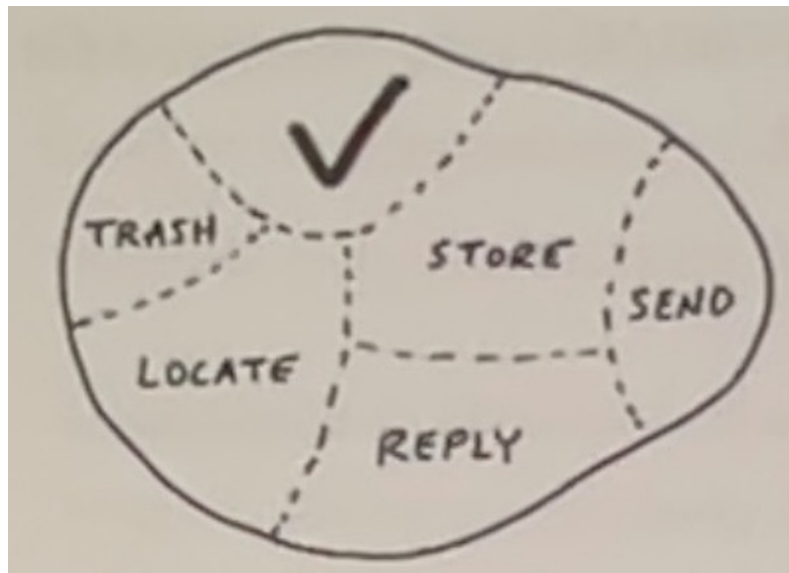


10 Components of Flow : Having a clear understanding of what you want to achieve, Being able to concentrate for a sustained period of time, Losing the feeling of consciousness of one's self, Finding that time passes quickly, Getting direct and immediate feedback, Experiencing a balance between your ability levels, and the challenge, Having a sense of personal control over the situation, Feeling that the activity is intrinsically rewarding, Lacking awareness of bodily needs, Being completely absorbed in the activity itself.

Working Mode

Developers can work in different modes in Momentum.

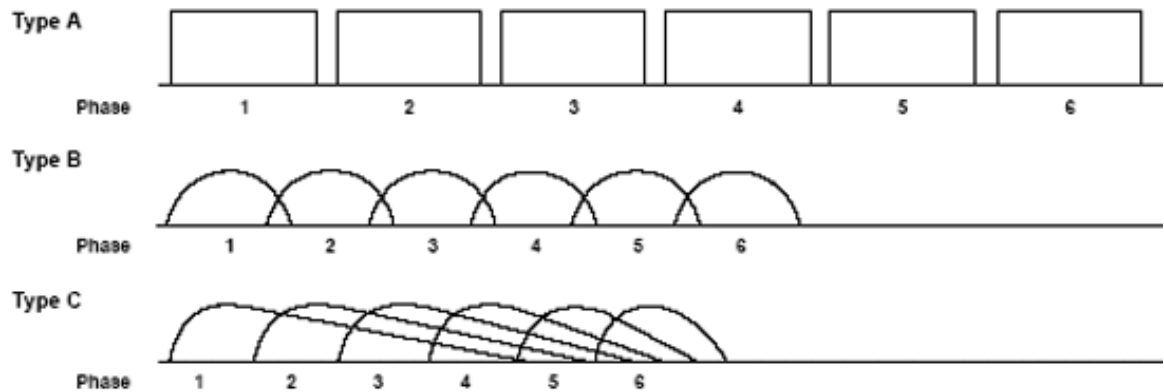
- Solo programming: It is preferable for individuals to work alone when they volunteer. A small, repetitive job is the best candidate for this type of programming at the beginning. A large depth complex work is also a good choice once he has expertise in the area
- Pair programming: It is a dialogue between two people trying to simultaneously program and understand how to program better. As a result, coding can be done with twice the brainpower, and one person in each pair is able to think about long-term issues. It is best suited for medium-sized, research-focused projects.
- Concurrent programming: Groups of developers can plan and develop simultaneously with concurrent programming. In this type of programming, the job can be divided into individual tasks that do not overlap with each other.



- Deep focus programming: Developers can concentrate on complex tasks with deep focus programming. This type of programming is best suited to jobs that have overlapping tasks.
- Overlap programming: The following image illustrates the difference between the traditional, linear approach to product development and the rugby or overlap approach. The sequential approach, labeled type A. The overlap approach is represented by type B, where the overlapping occurs only at the border of adjacent phases, and type C, where the overlap extends across several phases. This approach is essential for companies seeking to develop new products quickly and flexibly. The shift from a linear to an integrated approach encourages trial and error and challenges the status quo. It stimulates new kinds of learning and thinking within the organization at different levels and functions. Just as important, this strategy for product development can act as an agent of change for the larger organization. The energy and motivation the effort

produces can spread throughout the big company and begin to break down some of the rigidities that have set in over time.

Sequential (A) vs. overlapping (B and C) phases of development



Manage Team Member's Weakness

If the team member has weakness apply these strategies

- Find the team member a partner with complementary talents
- Reconfigure work to neutralize weakness
- Activate team member strength

Release not transformation

Management theory define the behaviors they expect from people and tell them to work on behaviors and don't come naturally. They praise those who can overcome their natural style to confirm to preset ideas. In short, they believe the manager's job is to mold, or transform, each employee into the perfect version of the role.

Great managers don't try to change person's style. They never try to push a knight to move in the same way as bishop. They know that their employee will differ in how they think, how they build relationships, how altruistic they are, how patient they can be, how much of an expert they need to be, how prepared they need to feel, what drives them, what challenges them, and what their goals are.

These differences of trait and talent are like blood types: they cut across the superficial variations of race, sex, and age and capture the essential uniqueness of each individual.

Like blood types, the majority of these differences are enduring and resistant to change. A manager's most precious resource is time, and great managers know that the most effective way to invest their time is to identify exactly how each employee is different and then figure out how best to incorporate those enduring idiosyncrasies into the overall plan.

To excel at managing others, you must bring that insight to your actions and interactions. Always remember that great managing is about release, not transformation.

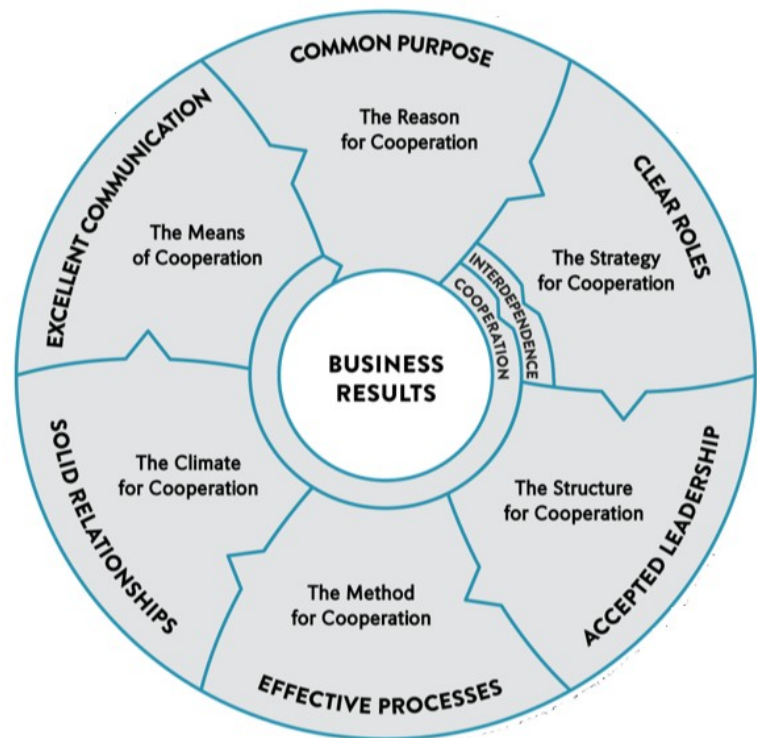
It's about constantly tweaking your environment so that the unique contribution, the unique needs, and the unique style of each employee can be given free rein.

High performing team model

The Performance Factor, a book by Pat MacMillan, gives the best graphical representation of a high-performing team model.

The model is represented as a circle because each characteristic is equally important for the team to achieve a desired and measurable business result.

The center of the circle and the ultimate goal. The point of teaming up is accomplishing results that you, as an individual, can't achieve alone. However, those results need to be measurable.



| Scrum | Momentum |
|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Suitable for concurrent programming | Provide a variety of programming options |
| Working towards the sprint goal | The team works on the feature |
| The Scrum methodology is based on lean thinking and empiricism | The Momentum is influenced by modern theories of personality, talent, collaboration and Alfred Adler's psychology, Agile values and principles |
| The "sit together" principle of Extreme Programming (XP) is implemented through the Commons and Cave model. | While Momentum endorses work-from-home approaches, it also advocates for the implementation of the Commons, Cave, and Deep Cave models. |

2.7. Individual Psychology

Adlerian theory is a holistic approach to psychology that emphasizes the importance of overcoming feelings of inferiority and gaining a sense of belonging in order to achieve success and happiness. This theory also focuses on the importance of social interactions and community involvement in order to promote individual growth.

This approach to therapy is based on the theories of Alfred Adler. His approach emphasized each individual's need for connection, belonging, and striving to overcome feelings of inferiority.

Momentum uses Adlerian psychology to resolve team conflicts and trade-off issues.

Problem behavior

Adlerian psychology, a distinctive branch of psychological thought, looks into the reasons behind kids' problem behavior. It breaks it down into five stages; this helps us see what's really going on beneath the surface and why kids do what they do.

The 5 stages of children's problem behavior are:

- Demand for admiration
- Attention drawing
- Power struggles
- Revenge
- Proof of incompetence

Reward and punishment are outdated

A child soon considers a reward his right and demands a reward for everything. He considers that punishment gives him the right to punish in turn, and the retaliation of children is usually more effective than the punishment inflicted by the parents. Children often retaliate by not eating, fighting, neglecting schoolwork, or otherwise misbehaving in ways that are the most disturbing to parents. Natural and logical consequences are techniques which allow the child to experience the actual result of his own behavior.

- Natural consequences are the direct result of the child's behavior. Natural consequences are usually effective. For example, if a child runs too fast, the natural consequence is they'll fall.
- Logical consequences are established by the parents, and are a direct and logical – not arbitrarily imposed – consequence of the transgression. The goal of logical consequences is to encourage positive behavior and not to punish. To effectively administer this approach, teachers must consider these 3 R's: Related, Reasonable, Respectful

Related – The consequences must be directly related to the behavior. For example, if a child draws on the wall, asking them to sit in the “naughty corner” is not a related consequence. Instead, the teacher must instruct them to wipe off the drawing. Related consequences make behavioral expectations clear to the child.

Reasonable – The consequence must also be age-appropriate and fair. For example, if Hailey grabbed a doll from Diane, subjecting Hailey to the “naughty corner” wouldn’t be reasonable. Instead, the teacher must encourage Hailey to give the doll back to Diane and apologize to her. Harsh consequences will cause resentment in the child.

Respectful – The consequence must not involve shaming or blaming the child. The teacher must communicate and enforce the consequence with a respectful tone while showing empathy. This will help the child understand that it’s not a punishment but an encouragement to behave appropriately. For example, if Liam speaks rudely when trying to get the teacher’s attention, the teacher can say calmly, “Liam, I would like to help you, but only if you speak politely,” and then return to whatever they were doing until Liam speaks politely.

Key theoretical principles

The following points summarize some of the key theoretical principles of what became Adlerian therapy

- Holism – Humans must be treated as a single unit rather than divided into separate parts.
- Superiority striving – Despite obstacles, we naturally strive (actively and creatively) toward excellence and task completion.
- Purpose – Humans work toward specific goals in life, driven by future hopes rather than past experiences.
- An idiographic approach – While generalizations can be helpful, every human must be considered unique.
- Soft determinism – Biology and environment influence, but do not determine, behavior.
- Freedom to choose – We are responsible for choosing our behavior from a limited number of options. Yet, we often make poor choices because of a lack of knowledge or education.

2.8. Motivation

Herzberg's Two-Factor Theory proposes that two sets of factors influence job satisfaction: hygiene factors and motivators. Herzberg investigated fourteen factors relating to job satisfaction in their original study, classifying them as either hygienic or motivational factors. Motivation factors increase job satisfaction, while hygiene factors prevent job dissatisfaction.

- **Hygiene Factors**

Hygiene factors, like salary and working conditions, don't motivate but can cause dissatisfaction if inadequate.

- **Motivators**

Motivators are factors that lead to job satisfaction and motivate employees to perform better. These include meaningful work, recognition, responsibility, opportunities for growth, achievement, and advancement. These factors are intrinsic to the work and are related to an individual's need for personal growth and self-fulfillment.

According to Herzberg's Two-Factor Theory, the motivators are the most potent in driving job satisfaction and motivation. Herzberg suggests these factors promote higher performance as they fulfill individuals' deep-seated needs for personal growth and self-fulfillment. However, the exact factor motivating most would vary based on the individual's values and personal needs.



2.9. Principles

Bureaucracies are authoritarian power structures. They enforce rules, often dull and suffocating, and pave the way for politicking. They discourage initiative and entrepreneurship. Bureaucratic organizations are inertial, risk-averse, and not open to innovation. In such organizations, there is no room for personal growth. Employees are limited by a list of responsibilities dictated by their roles, and don't really have voice. All power is concentrated in the hands of managers, especially those at the very top.

Treated as resources, employees feel highly disconnected from what they do at work, and don't contribute as much as they can. Unfortunately, the vast majority of modern organizations still exist in the old traditional paradigm that curbs the freedom of employees and impedes progress:

Humanocracy address a simple but deep question: Why do our organizations so often disappoint us and what can we do about it? Part of the answer involves rebuilding our organizations atop a set of post-bureaucratic principles. Principles of Humanocracy are embraced by Momentum.



2.10. Law of Mobility

Open Space Technology

The law of mobility comes from Open Space Technology, allowing members to switch teams at any point if it makes sense.

"Resilient learning team members form a collaborating, self-organizing team.

They have the autonomy to determine how they approach the work.

The essential motivators - purpose, co-intelligence, and autonomy

- lay the foundation for enabling high performance.

Purpose and co-intelligence without autonomy equals dependencies and bottlenecks"

- Diana Larsen and Tricia Broderick

Momentum supports both fluid teaming and dynamic reteaming. In Momentum teams are formed around product backlog item, delivery schedule like Kanban (daily-wise plan change), Scrum (weekly plan change) or Shape Up (monthly plan change) and individual personality.

- **Fluid Teaming**

Fluid teams are a concept where team composition is dynamic and can change as needed to meet the evolving requirements of a project. Unlike traditional fixed teams, where members remain constant throughout a project, fluid teams adapt by reallocating resources and personnel based on the current phase of work, skills required, and challenges faced.

The purpose of fluid Scrum Teams is to organize around the work at hand. The starting point is a larger pool of people, forming a team. Every Sprint, the people from the team organize into smaller teams to tackle specific complex issues. Every Sprint the pool of people will reorganize differently because of the different nature of the issues at hand.

- **Dynamic Retesting**

Dynamic Retesting is about accepting the fact that the team changes over time. People come, people leave. Teams grow and may need to split. Teams may dissolve. The purpose of Dynamic Retesting is to cope with changes in the team composition. Helfand identified several patterns of dynamic retesting:

- One by one - Adding or removing one person at a time.
- Grow & Split - Growth of the team and splitting it into multiple teams.
- Merging - Combining two or more teams into one.
- Switching - Switching people from different teams.
- Isolation - Extracting a small team from a larger team.

Big Teams

Big teams can be valuable too. Small teams aren't the answer to everything. It is the starting point of Fluid Scrum Teams. I see how product environments may require teams larger than typical Scrum Teams with a wider variety of skills.

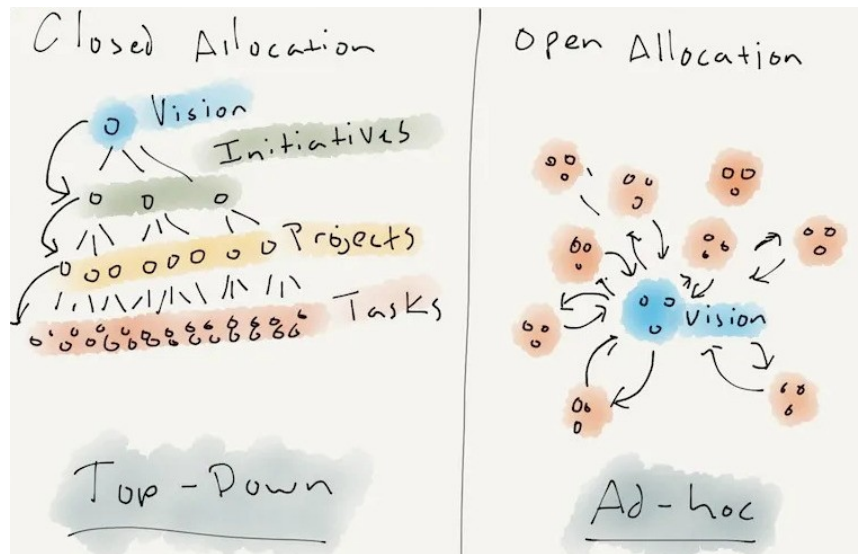
At its core, a pool of people to form Fluid Scrum Teams is larger than a typical Scrum Team. Every Sprint, they reorganize themselves to have focused smaller teams. But this happens from the premise of a larger team.

Open Allocation

Open allocation refers to a style of management in which employees are given a high degree of freedom in choosing what projects to work on, and how to allocate their time. They do not necessarily answer to a single manager, but to the company and their peers.

They can transfer between projects regardless of headcount allowances, performance reviews, or tenure at the company, as long as they are providing value to projects that are useful to the business goals of the company. Open allocation has been described as a process of self-organization.

Open allocation is not a free-for-all. In fact, it increases individual accountability because no one has the excuse of being put on a bad project or landing under a shitty boss. If you fail to make an impact in an open-allocation shop, it's on you. You had the same opportunities to succeed as everyone else.



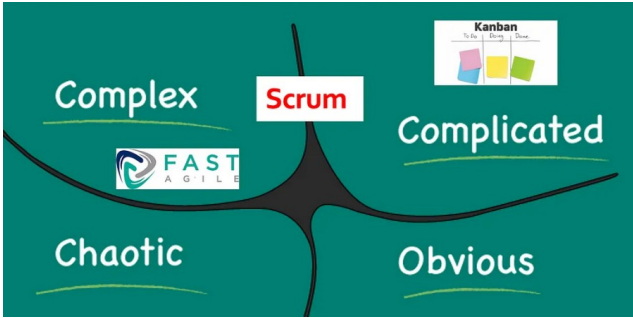
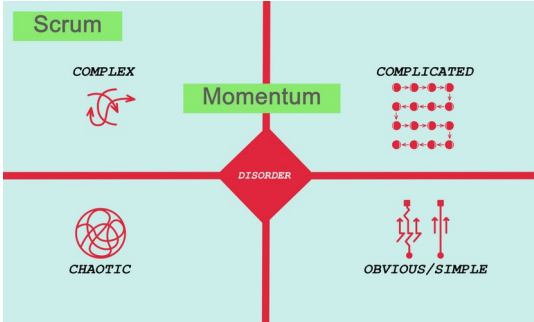
No one can force you to do something, unless it's an existential issue for the organization. On the flip side, you can't force other people to do things. If you have an idea that can add value to the firm and people will follow you, you can do it. You don't have to be a "manager" to start an initiative. If you're not ready to lead, you follow. There's no stigma associated with following (which most people will), and you're expected to follow if you're not ready to lead.

Open allocation does not mean "everyone gets to do what they want". A better way to represent it is: "Lead, follow, or get out of the way" (where "get out of the way" means "leave the company").

4 Typologies of People

The human aspect is important in Open Allocation and to build a great team you have to take into consideration the 4 typologies of people: Subtractors — everybody starts here — the input they require is greater than their output. Based on how fast and well they learn to function within the team, usually 6 months, they shift to Adders. In an Open Allocation setup, most go beyond that and can become Multipliers.

Dividers are the people who make whole teams less productive. Unethical people are dividers, but so are people whose work is of so low quality that messes are created for others, and people whose outsized egos produce conflicts. Long-term (18+ months) subtractors become "passive" dividers because of their morale effects. Dividers smash morale, and they're severe culture threats, capable to take down entire companies.

| Fluent Scaling Technology | Momentum |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>FaST combines Open Space with Open Allocation.</p> | <p>Momentum is based on multiple methodologies including Open Space and Open Allocation</p> |
| <p>FaST teams are formed around work. It has 5 steps as follows</p> <ul style="list-style-type: none"> • Merge teams into a collective • Throw work on the wall • Let teams emerge and self-organize around work • In two days, we will meet back and share progress • Repeat | <p>Teams voluntarily organize themselves around following factors</p> <ul style="list-style-type: none"> • Product backlog items utilizing job documents • Delivery schedule, which may follow methodologies such as Kanban (with daily plan adjustments), Scrum (with weekly plan updates), or Shape Up (with monthly plan revisions) • Individual personalities |
| <p>FaST says</p> <ul style="list-style-type: none"> • The Scrum framework works well if the problem is part complex and part complicated • The FaST methodology works well with complex problems  | <p>Momentum says</p> <ul style="list-style-type: none"> • The Scrum framework works well if the problem is fully complex • Momentum methodology works well if the problem is part complex and part complicated  |
| <p>FaST does not offer a formal collaboration mechanism</p> | <p>Momentum offers a formal collaboration mechanism</p> |

3. Scaling Setup

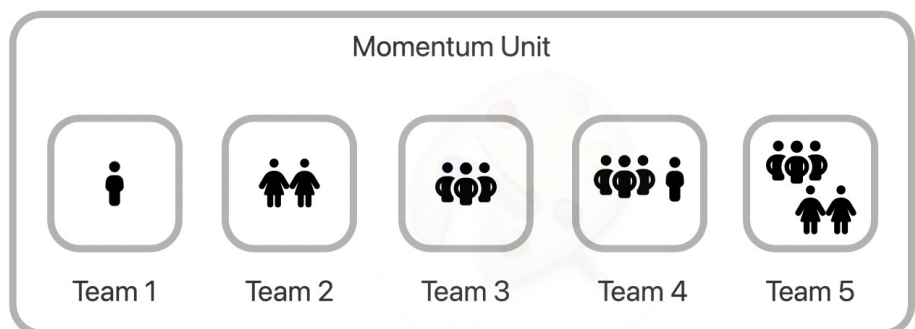
3.1. Team

Momentum's basic unit of development is the Team.

- Teams are fluid, dynamic, short-term, and developers volunteer for them
- A team can consist of one or more members. Even a single individual can constitute a team. The teams are cross-functional and feature-focused. Teams can be co-located or distributed
- A team can have any number of members. It is as beneficial to have one member of a team as it is to have ten members.
- The ideal size of the group should be less than six, after which subgroups will be formed. By creating conscious subgroups according to the type of work, more benefits can be gained
- There are a variety of methodologies that each team can use, including Scrum, Kanban, or Shape Up, or no methodology at all. Teams can choose the length of their cycle

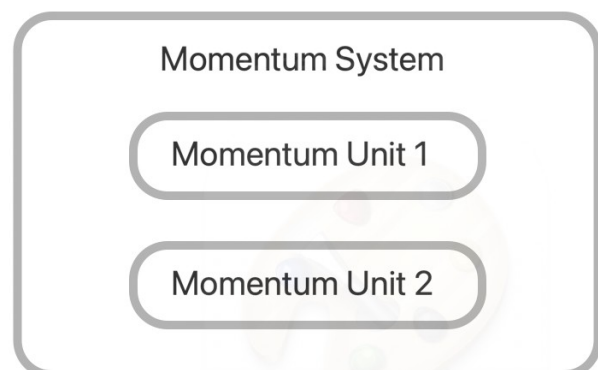
Momentum Unit

A Momentum Unit is a collection of teams working on a product or area of the product. In order to better meet customer needs, product areas can be created based on where requirements come from. Every Momentum Unit can have an unlimited number of teams.



Momentum System

Collection of Momentum Units are called Momentum System. Momentum System consists of stable, long running and a product goal oriented team members.



| Scrum | Momentum |
|---------------------------------------------------|---------------------------------------------------------------------------------------|
| Static and long-running teams | Fluid, dynamic and short-running teams |
| Focus on team level | Focus on product level |
| Concentrate on solving complex problems | Put an emphasis on individual strength and the skeleton of the job |
| Does not allow developers to move to another team | Whenever possible, developers should volunteer to be part of any team that needs them |

3.2. Tracks

The nature of unshaped work makes it risky, complex, and difficult to schedule. We therefore have two separate tracks: one for shaping (discovery track) and one for building (development track). The Momentum Unit can be divided into two tracks. There may be one or more teams in each track.

1. Discovery Track
2. Development Track



3.3. Roles

There are four roles in Momentum System.

1. Discovery Role
2. Developer Role
3. Ambassador Role
4. Support Role



The following table shows how the different capabilities fits into the following 4 roles.

| Discovery Role | Developer Role |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Techno-functional consultant Domain specialist, Architect Product strategist, Business analyst Product manager, UX designer Sales manager, Marketing manager | Software developer, DevOps specialist Manual tester, Automation engineer Performance automation engineer Software security engineer UI designer, SRE engineer |
| Ambassador Role | Support Role |
| Expert software developer Technical coach, Collaborator | Agile coach, Product leader, CxO Professional coach, Facilitator |

Discovery Role

Techno-functional experts, product manager, business analyst, architects and UX designers are part of the discovery team. This track would benefit from Scrum as a methodology.

A team member may participate in a design thinking and user research workshop to elicit requirements. Members of a team can create and share JIT working documents, which are also known as job documents in Momentum.

Responsibilities

- Understand the business motivation for building the product
- Understand the users that will use the product and the problem solves for them
- Learning the market and customers' needs
- Collaborate with the others to ensure a shared product vision and road map
- Managing and prioritizing the product backlog
- Iteratively build and test prototypes with the users
- Shaping and creating the job document from the user requirements
- Choose the right team for the job, or work with the developers to form a new one
- Communicating the job to the developers. Verify the work completed by developers
- Participate in acceptance testing to validate product
- Responsible for showcasing the completed development work and Job document
- Responsible for collaboration with both discovery & development track members

Developer Role

Team members who are part of developer role are software developers, testers, automation experts, Devops, UI designers. Scrum, Kanban, or Shape up are the suitable methodology for this track.

Responsibilities

- Volunteer for the job, selecting the development methodology
- Finding development partners for pair, concurrent and deep focus programming
- Responsible for collaboration within the team and outside team
- Build, test, automate, deploy, release within the cycle

Ambassador Role

An ambassador is an expert software developer on the team. Each development track can have one member in this role. Anyone can participate in this role since it is a volunteer role. Within the development track, this role is rotated.

Responsibilities

- Primary responsibility for fostering collaboration within and outside the team
- Providing product coaching and mentoring to team members
- Assisting the discovery team with technical details
- Assist in forming new dynamic teams by collaborating with discovery team members
- Inviting new members to the team
- He does not participate in the development process and is not a team representative

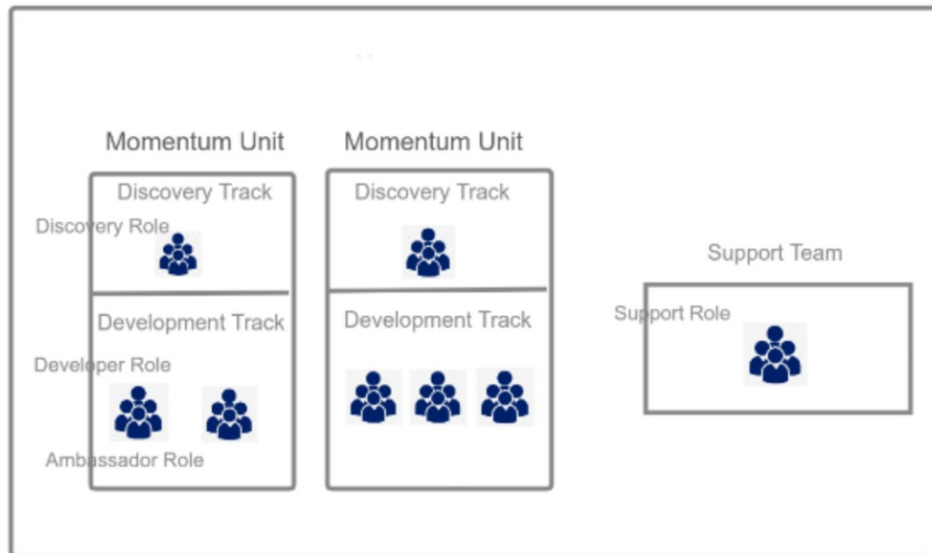
Support Role

Support roles include product leaders, agile coaches, professional coaches, facilitators, on-site customers and other leadership roles. The support role does not belong to the Momentum Unit, but to the Momentum System.

Responsibilities

- Identifying team members' talents, skill sets, and personalities
- Responsible for the team member's success
- Responsible for managing process improvement
- Responsible for facilitation, conflict management, providing feedback, and coaching team members

Momentum System

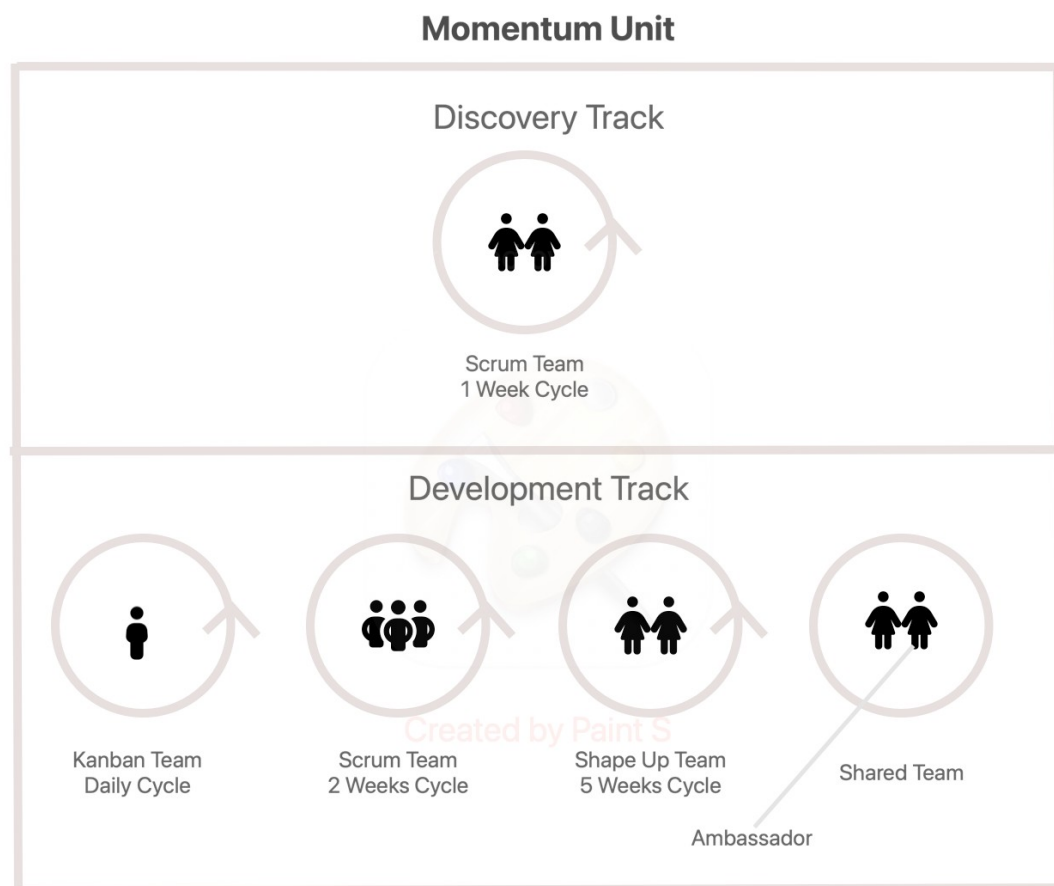


| Scrum | Momentum |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The Product Owner is responsible for creating user stories. There is only one Product Owner, not a committee. Product owner is accountable for the product's success | Members of the discovery role are responsible for creating job documents. Discovery roles can be filled by one or more team members. It recommend at least to have 2 members. A product's success depends on everyone |
| Developers and daily scrum events are given more attention. | Putting more emphasis on discovery team members shaping the process and discovery track sprint reviews. |
| A Product Owner may have an MBA, marketing or sales background | Members of the discovery team must have a techno-functional background |
| The sprint review discusses completed development work and upcoming sprint goals | At the cycle review meeting, completed development work and jobs are presented |
| Stockholders cannot improve the user story until it is completed | Members of the support role have the opportunity to improve requirements before they are developed |
| Within the cycle, release to the customer is optional | Within the cycle, a release to the customer is required |

3.4. Configuration

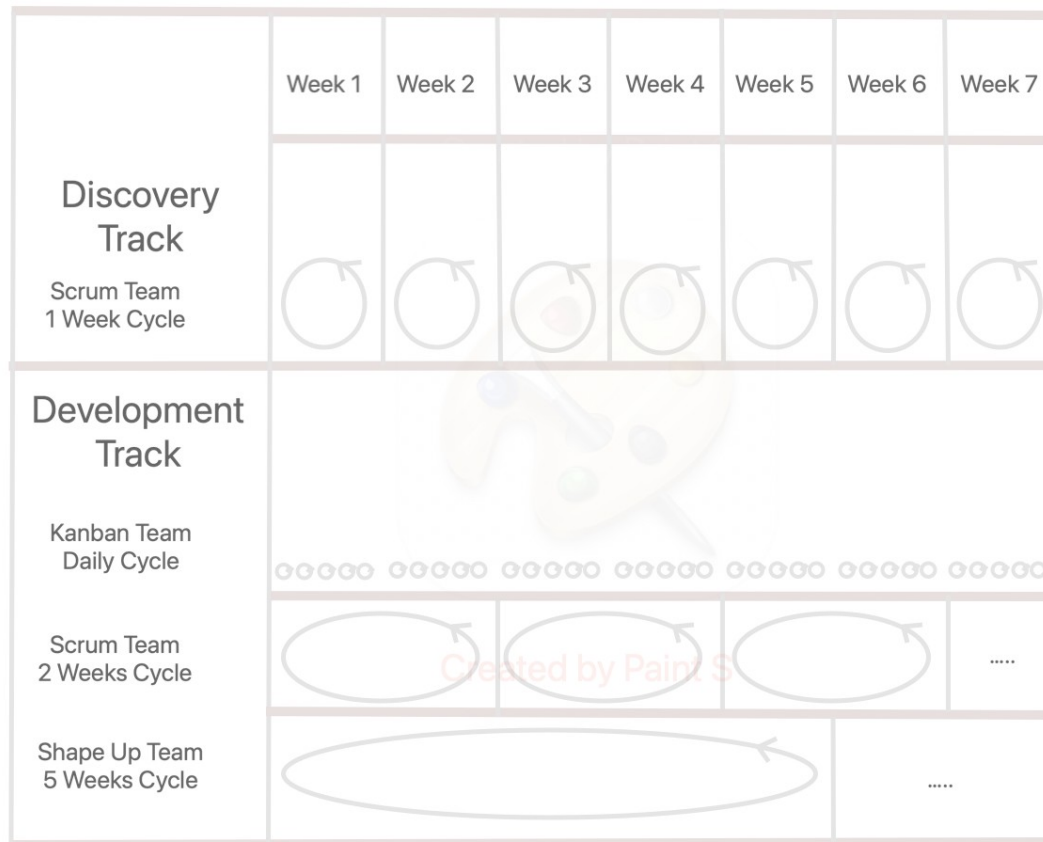
The following settings are available for every Momentum unit

- There are two tracks available: discovery and development
- There can be more than one team for each track
- As there are no limits to the number of teams, each can have one or more members
- Ideally, teams should consist of five people, but it's up to participants to decide how many members they should have
- The number of shared members is not included in the team count
- Scrum is used in the discovery track, while Kanban, Scrum, Shape Up, or any other methodology may be used in the development track
- Momentum recommends that each team have overlapping cycles. In this way, the discovery team manages their cycle planning and review meetings
- Shared teams do not have their own cycle and are actively involved with development teams. A shared team member can attend all development track cycle meetings



The following picture illustrates the cycle length for each team.

Momentum Unit Cycle



| Scrum | Momentum |
|----------------------------------------------------------|----------------------------------------------------------------------|
| The team must have a minimum of 3 members | Teams with one member are as beneficial as those with five members |
| Product backlog items are measured by story points | Do not measure the size of the job, but rather identify its skeleton |
| Team velocity should be measured | Measure collaboration engagement rather than team velocity |
| The aim is to present the completed work to stockholders | Aim to release the new software feature to customers |

3.5. Backlogs

There are three levels of backlog in every Momentum System

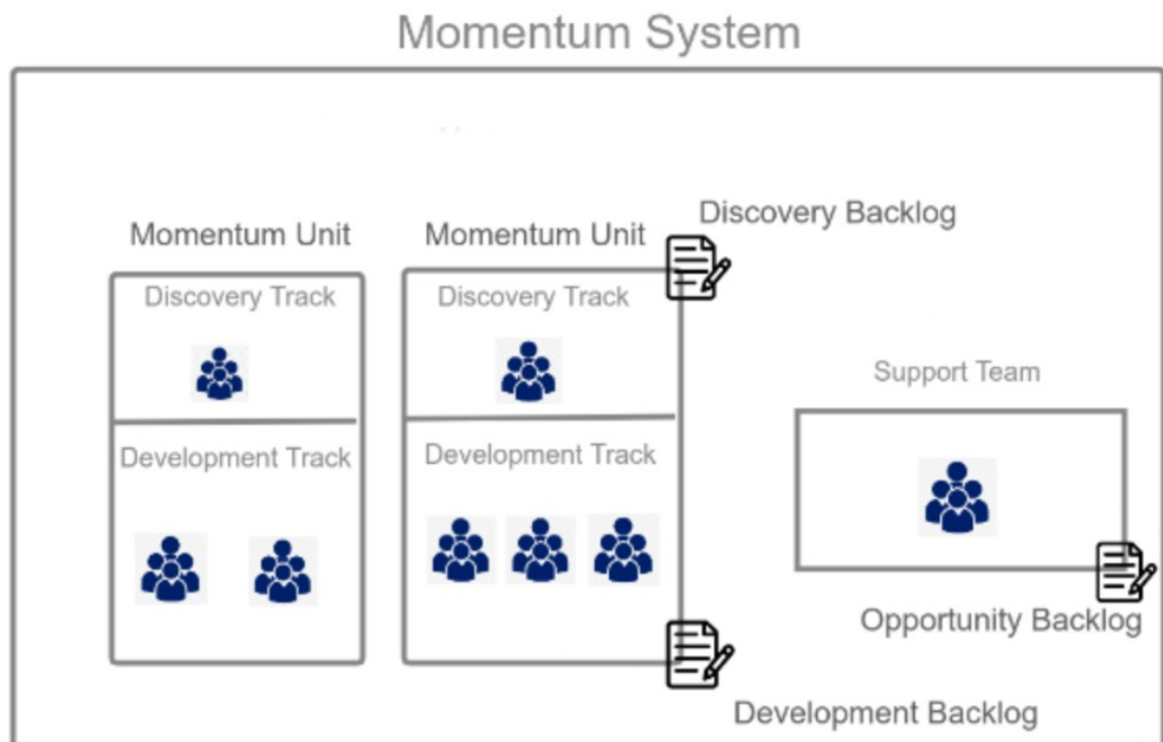
- Opportunity backlog
- Discovery backlog
- Development backlog

Momentum is made up of multiple Momentum units. Momentum's opportunity backlog contains ideas, issues, strategic initiatives, and opportunities. A team member who is part of the support role is responsible for managing the opportunity backlog. As soon as opportunity backlog items have been categorized and prioritized, they are moved to the respective Momentum unit's discovery backlog.

In the discovery backlog, you will find ideas, requirements, spikes, bugs, and everything else related to a product or a specific product area. A shaping and reviewing process is applied to every item in the discovery backlog.

Discovery team members move it to the development backlog if it gets prioritized.

Each development team picks a job from the development backlog when the cycle starts.



3.6. Meetings

Despite altering many of its practices, Momentum preserves the core values of the adopted methodologies. Momentum System allows team members to attend any meetings. Below is a list of meetings.

Cycle Planning

A 60-minute planning meeting is organized by discovery team members. A member of the discovery team and all members of the respective development team and shared team members attend the planning meeting. The planning meeting is divided into three parts. The first and second parts are discussed in 60 minutes each.

Part 1: A member of the discovery team communicates the job in the development backlog in both written and verbal forms

Part 2: The developers and testers clarify their doubts and suggest scope hammering.

Part 3: The testing process is divided into several phases. These statuses can be shared as early-draft-1, early-draft-2, and final-1 in a hill chart. Developers commit to the testing team later in the day when they can expect their early drafts. Part 3 is not a meeting, but rather an opportunity for testers to schedule their time for future tests.

Cycle Daily Meeting

The daily meeting is organized by the development team and it must be less than 60 minutes. Meetings can be used for brainstorming with team members or planning concurrent programming. It is an optional event.

Cycle Review

A review meeting is organized by the discovery team and lasts up to 60 minutes. Attendees include members of the support team. There are three parts to the review meeting. It is an optional event for development track members.

Part 1: The team presents the completed development work from the development backlog

Part 2: The team presents the completed job document in the discovery backlog

Part 3: The team presents their plan for the upcoming cycle in the opportunity backlog

Cycle Retrospective

A retrospective meeting is only held by members of discovery and development teams of Scrum teams. Maximum time is 60 minutes for this meeting.

The list of meetings and details of participation are provided here.

| | Cycle Planning | Cycle Daily Meeting | Cycle Review | Cycle Retrospective |
|-------------------------------|----------------------------------------|----------------------------------------|----------------------------------|----------------------------------------|
| Discovery Track | | | | |
| Scrum Team 1 Week Cycle | ✓ | ✓ | ✓ | ✓ |
| | Discovery & support role members | Discovery role members | Discovery & Support role members | Discovery & Support role members |
| Development Track | | | | |
| Scrum Team 2 Week Cycle | ✓ | ✓ | | ✓ |
| | Development role & shared team members | Development role & shared team members | | Development role & shared team members |
| Kanban Team Daily Cycle | ✓ | | | |
| | Development role & shared team member | | | |
| Shape Up Team 5 Week Cycle | ✓ | | | |
| | Development role & shared team member | | | |

- **Requirement Filters**

In Momentum, every requirements pass through 5 levels of filters before completion.

- A first step is at the discovery process, the discovery team prepares a job document.
- A second step is to present the job document at the sprint review and have members of the support role review it. Members of the support role can provide feedback before it is developed.
- In the third step, the developers work on it after cycle planning and deployment for testing.
- A fourth step involves the discovery team members verifying BDD for the features.
- The fifth step is to demonstrate the feature to support role members during the sprint review

| Scrum | Momentum |
|--------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A Kanban board or Scrum board is used to share development statuses | The hill chart is used to share the status of development. |
| Scrum boards or Kanban boards are used by testers to pick their testing items | It is not possible to share the status of phased testing on a Scrum board or Kanban board. Hill charts can be used to share multiple phased testing updates |
| The review meeting must be attended by developers | Participation in the review meeting is optional for developers |
| The backlog also includes process improvement and learning objectives | The backlog consists only of job documents. Items for process improvement are directly added to the retrospective hub |
| We are committed to developing the software in a short amount of time and verifying its functionality with our customers | We place a great deal of importance on verifying the requirements and shaping them |

3.7. Collaboration

At a higher level, the research reveals several characteristics that define successful teams:

- Everyone on the team talks and listens equally, keeping contributions short and sweet
- Members face one another, and their conversations and gestures are energetic
- Members connect directly with one another – not just with the team leadership
- Members have back-channel or side-channel conversations within the team
- Members periodically break, explore outside the team, and bring information back

The data also establish another surprising fact: Individual reasoning and talent contribute far less to team success than one might expect.

There are 2 ways collaboration can be achieved: in an informal setting and in a formal setting.

Collaboration engagement in informal setting

If team members know each other or are physically present at the same location, the informal approach is effective. It is possible for anyone to request information or a meeting, but there is no guarantee the requester will receive the information on time. This is where formal engagement comes in handy. It is impossible to improve this informal engagement because it cannot be measured.



Collaboration engagement in a formal setting

Momentum limits the formal cycle meetings as much as possible but it provides a formal framework for collaboration. This formal collaboration engagement is enabled by the ambassador role. When team members are distributed, working remotely, or sitting some other room, this approach is effective. When a member of the team seeks assistance from outside his team or outside his Momentum Unit, it is most beneficial to do so.

Ambassador role members can be requested for information or assistance in development activities by anyone in the Momentum system. This is a one-on-one conversation. Requests can be raised at three levels. In order to raise a request at Level 2 or Level 3, a JIT working document about the problem must be prepared.

Level 1 – Live chat support. Responses must be provided within 10 minutes

Level 2 – Support for audio and video calls. Responses must be provided within 20 minutes.

Level 3 – Support development activities by screen sharing with audio or video. Responses must be provided within 30 minutes

Level 4 – A member of a team may request another member of the team to participate in an informal meeting or in a formal meeting for a variety of reasons. Momentum calls this an ad-hoc meeting. Two hours before the meeting, a request must be made and a JIT working document about the problem must be provided. The meeting is a group one.

There are 9 types of ad-hoc meetings: Decision-making meetings, Planning meetings, Brainstorming sessions, Bonding meetings, Review meetings, Update meetings, Production issues meetings, Customer new request meetings, and Customer support meetings.

It is possible for participant to rate each other after the engagement, allowing this to be measured and improved.

3.8. Communities

A community is a group of volunteers with a common interest or topic who strive to deepen their knowledge and take action through discussions and interaction with peers. It is completely voluntary and informal to participate in communities.

A community is not a team and does not implement items. There are communities for functional practices (e.g. design and architecture) as well as for other interests, such as infrastructure tooling, communication, product management, etc.

Community members are spread throughout the organization. It is important for communities to be dynamic. A community can be started by anyone, but if there is no passion in it or it is working dysfunctional, then it will die.

Communities' aims and authority

- **Learning**
Communities promote knowledge sharing, learning, education, coaching, identifying work, and improving skills. Clean-code communities and test communities are examples of communities of practice
- **Cross-team agreements**
There are product-level or enterprise-level cross-team concerns to address. UI standards, architectural guidelines, and test automation practices are examples

Both aims are often fulfilled by many communities. While communities cannot make decisions for teams, they can produce something that the teams can adopt.

Community tips

- The community coordinator should have a passion for the concern and desire to cultivate a caring community
- Make an active effort to recruit participation from all teams

3.9. Team Formation

In Momentum methodology, A team is fluid, dynamic, short-term, where developers volunteer to participate. Teams are formed based on the job's urgency, priority, skeleton, dependencies, work mode, individual personality, and collaboration style. As an example of how you can understand this concept, visualize every job as a new project. Once the discovery team recognizes the difficult task ahead, the following steps will assist in the formation of a new dynamic team.

- Understanding what makes the job require a new team
- Understanding why the existing teams may not be a good fit for the new job
- Identify the type of talent, skill set, and personality profile needed for the new job
- Look for members on the communities page and contact ambassadors for more information & let everyone know about the new job opportunity through the jobs hub
- Analyze the profiles of volunteers
- Discuss the methodology, cycle length, collaboration style. A team could agree on such things as: Working synchronously or asynchronously? Core hours? Solo work or ensemble Collocated or distributed? If collocated, then office, someone's home, or a coffee shop?
- Create a dynamic team
- Once the job is completed, the members join their previous units or find another task

| Scrum | Momentum |
|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Provides support to one team | Provide support to multiple teams |
| There can only be 3-9 members in a single team | There is no limit to number of team members but the recommended size is 5 members, and all shares the same goal. If it is more than 5, all members must work for the same feature |
| The team consists of permanent members. Multiple priority issues are handled by one team | The team consists of fluid team members. It is the responsibility of each team to take care of its own priority issues. |
| A sprint goal is being worked on by the team | A feature is being worked on by a team |
| Sprint plans do not change | Cycle plans do not change |

3.10. Scenarios

This is an example of a product managed by 50 developers, organized into 5 Momentum Units, with each unit comprising ten developers. We'll analyze how Momentum handles different hypothetical scenarios.

| Track | Scenario | Momentum Solution |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Development Track | There is a need for lighter assistance from another team | By using the 4 levels of formal collaboration engagement, developers can request assistance from respective unit's ambassador |
| Development Track | There is a need for stronger assistance from another team within the unit or another unit | Volunteers from another unit temporarily join the team for a specific task, then return to their units afterward |
| Development Track | There are multiple Momentum units involved in developing a new feature | Create a dynamic team whose members are drawn from all 5 teams rather than splitting the feature into 5 sub items |
| Development Track | Developers want frequent information from customer support or implementation team | Add an ambassador role to the respective departments. This allows the developer to contact the other departments |
| Discovery Track | There is a member of the discovery team who needs lighter assistance from another unit | Using the 4 levels of formal collaboration engagement, an ambassador from another unit can help a discovery team member |
| Discovery Track | The discovery team needs stronger assistance from another unit | Members of another discovery team temporarily join the team for a specific task, then return to their units once the task has been completed |
| Operations Track | Scaling the track – SRE's operational team performs complex cloud operations tasks that require separate cycles | A new track that requires an SRE operational team should be added to the unit |
| Development Track & Discovery Track | Scaling the Momentum unit – There is a need for both discovery and development team members when developing a new feature | Make a temporary Momentum unit and create volunteer teams from both the discovery and development tracks. Once the volunteer teams have completed their work, they return to their original units |

3.11. Implementation Guidelines

The following guidelines help to implement Momentum methodology.

- Do not start Momentum with its full capabilities. Start small and go slowly
- Start using just 2 members. One for discovery and another for development
- First focus on the discovery track and then focus on the development track
- Depending on the scenarios you encounter on the project tackle it with Momentum recommendation
- Identifying the team member personalities and interest are key to unlock their potentials
- Always set the feature expectation properly by using job documents
- Very careful with fluid teaming and dynamic reteaming. Without setting the proper organization culture, this put the project in risk
- Adopt peer-based hiring approach & lean budgeting
- Make sure every practices align with agile values and principles
- The team can be made up of one member or a group of many members, but everyone should work towards a single goal. If team splits into multiple sub groups and focus on different goal is a sure way to failure.
- One ambassador role may be required for every ten developers, as a rough estimate
- Identify where the stress is created in the system, is it in the discovery track, development track or delay in a lack of specific skill-set like UI or architecture. In order to fine-tune the momentum, the most important step is to recognize the stress. It may also be possible to help the team member learn the skill or to add more members to the discovery team or development track or to give them more autonomy and freedom
- As team members move between different goals, teams, and cycle times, the product must always expect 100% end-to-end automation. Quality assurance of the product is maintained through the implementation of unit tests, integration tests, UI automation tests, and performance tests. A culture of test automation is crucial to the momentum approach's success
- George Box's well-known quote asserts that "All models are wrong, but some are useful," emphasizing the importance of adaptability in problem-solving. Similarly, Momentum alone isn't a solution. The key to success lies in consistently refining and enhancing the model that best fits the project type and phase.

4. Performance Management

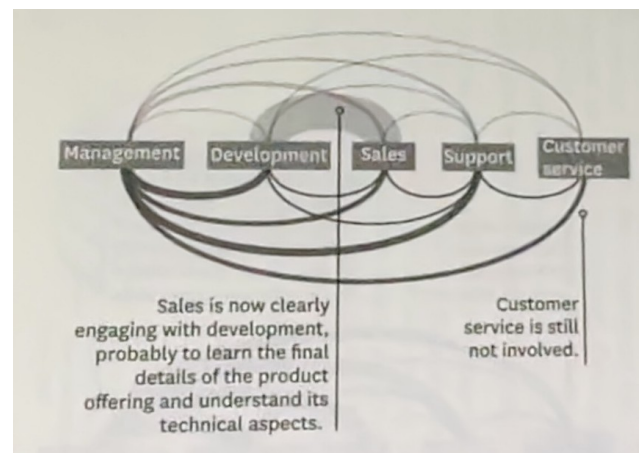
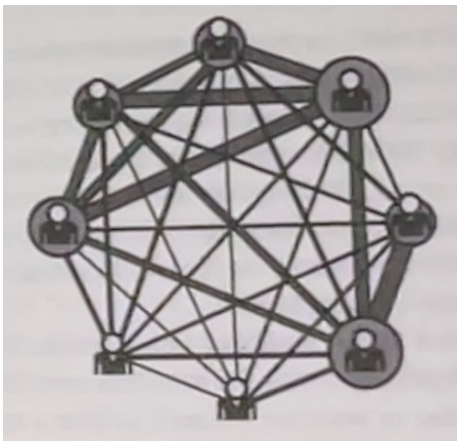
Instead of monitoring developers' work hours, user story size, or team velocity, the momentum methodology concentrates on creating an environment that supports developers in delivering outcomes. The hubs serve as dashboards that enhance team transparency and collaboration

4.1. Collaboration Hub

The collaboration hub allows team members to request collaborative conversations, rate their effectiveness, and generate reports.

- Evaluate the effectiveness of ambassadors' collaboration engagement
- Provide the various levels of formal engagement of the team member

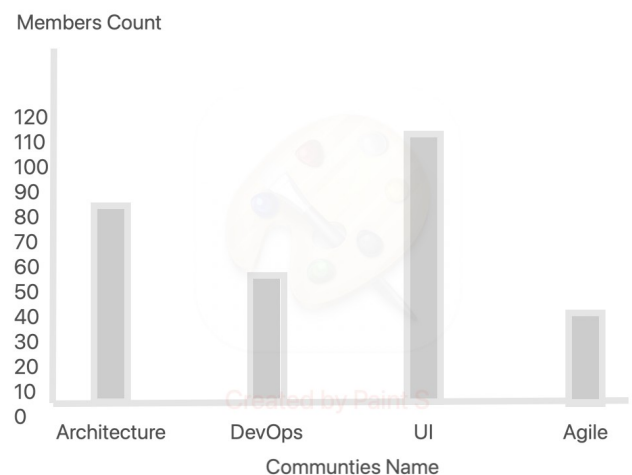
Two team members (bottom and lower left) are not engaged, while another (top right) engages effectively.



4.2. Community Hub

The community hub provides information about all the special interest groups and options for joining or creating a group.

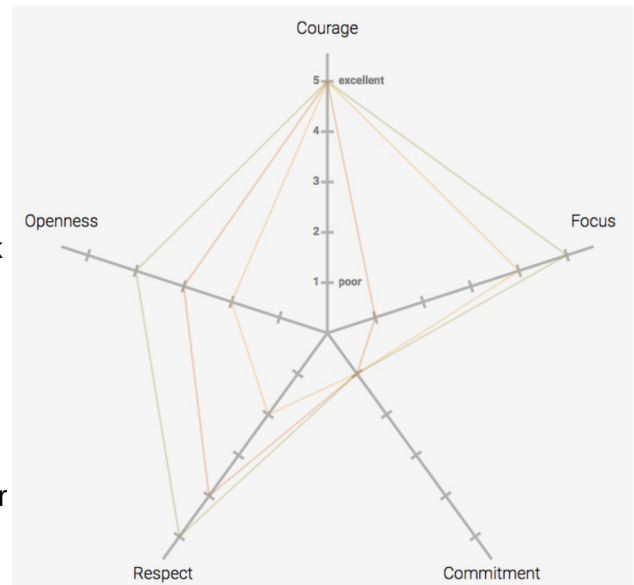
- It lists active communities
- A list of team members and their special interests communities
- Produce a range of reports about the activities and outputs of the communities



4.3. Retrospective Hub

All the findings and improvements from each team's retrospective are listed together at a retrospective hub. Organizing regular retrospectives helps people identify and resolve issues, handle obstacles, and improve practices. Based on these findings, we have divided them into five categories: productivity, capability, quality, capacity, empowerment, and enjoyment.

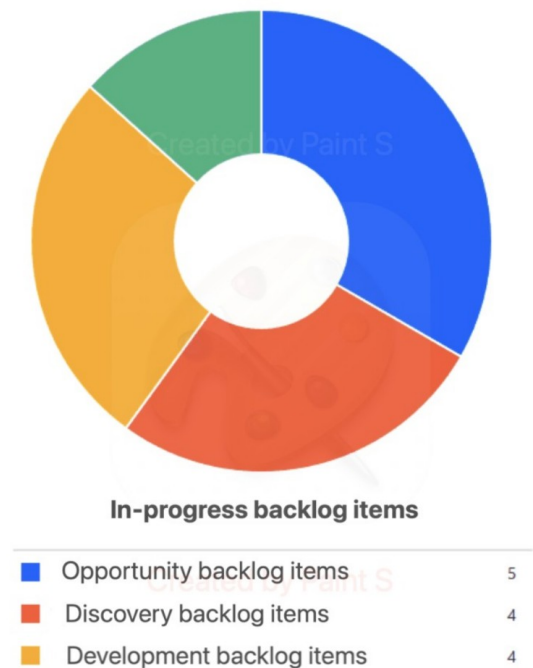
- The Scrum team holds a retrospective meeting and adds their findings.
- Whenever there are team or systemic issues, Kanban and Shape-up teams can add them
- The hub accepts anonymous feedback about the past cycle or ongoing problems
- Additionally, it allows users to rate the cycle from one to five
- There are many templates available for retrospectives on a hub as well
- It is the support team's constant duty to monitor these hubs and provide solutions or feedback as needed



4.4. Jobs Hub

All backlogs are listed in the Jobs hub: opportunity backlog, discovery backlog, and development backlog.

- There is an option to explore present and past backlog items with different statuses and filter them based on different criteria
- Backlog items selected for past releases
- Analyze the backlog items' cycle time and lead time
- Allow team members to express interest in working on a specific backlog item and allow anyone to invite others to work on it



4.5. Product Value Hub

This hub helps measure, manage, and increase the value of the product. The goal is to improve outcomes, reduce risks, and optimize investments. The following measurements are listed in the product value hub.

| Value | Measuring |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Customer usage index | Measurement of customer usage by feature |
| Customer satisfaction | Some form of sentiment analysis to gauge customer engagement and happiness with the product |
| Employee satisfaction | Some form of sentiment analysis to gauge employee engagement, energy, and enthusiasm |
| Product cost ratio | Total expenses and cost for the product being measured, including operational costs compared to revenue |
| Market share | The relative percentage of the market not controlled by the product; the potential market share that the product might achieve if it better met the customer needs |
| Production incident count | The number of times in a given period that the development team was interrupted to fix a problem in an installed product. The number and frequency of production incident can help indicate the stability of the product |
| Time to remove impediment | The average amount of time from when an impediment is raised until when it is resolved. |

5. References

- 22 Most Effective Adlerian Therapy Techniques and Worksheets – Jeremy Sutton
- Accelerate – Nicole Forsgren, Jez Humble and Gene Kim
- Agile Software Development: The Cooperative Game – Alistair Cockburn
- Agile Software Development Ecosystems – Jim Highsmith
- Agile Project Management: Creating Innovative Products – Jim Highsmith
- An Agile Adoption and Transformation Survival Guide – Michael Sahota
- Cracking the Code of Change – Nitin Nohria and Michael Beer
- Grit – Angela Duckworth
- Head First Agile – Andrew Stellman and Jennifer Greene
- Helping People Change - Richard Boyatzis, Melvin Smith, Ellen Van Oosten
- Herzberg's Two-Factor Theory Of Motivation-Hygiene – Charlotte Nickerson
- High Performing Teams: What Are They and How Do I Build One - Branislav Moga
- How Google Works – Eric Schmidt and Jonathan Rosenberg
- Humanocracy – Hamel and Zanini
- The Dynamic Reteaming Ecocycle - Heidi Helfand
- Large-Scale Scrum – Craig Larman and Bas Vodde
- Leading Teams – Mary Shapiro
- Managing Oneself – Peter F. Drucker
- Organise your venture like the best tech firms - Open Allocation - Tudor-Nicolae Birlea
- Out of The Crisis – W. Edwards Deming
- Shape Up – Ryan Singer
- Switch: How to Change Things When Change is Hard – Chip and Dan Heath
- Team Habits – Charlie Gilkey
- The Art of Doing Twice the Work in Half the Time – Jeff Sutherland
- The Courage To Be Disliked – Ichiro Kishimi and Fumitake Koga
- The Evidence-Based Management Guide – Scrum.org
- The Kanban Pocket Guide – Daniel Vacanti
- The Flow Model By the Mind Tools Content Team
- The New New Product Development Game - Hirotaka Takeuchi and Ikujiro Nonaka
- The Strange Life of Nikola Tesla – John R.H. Penner
- The New Science of Building Great Teams – Alex Sandy Pentland
- What Great Managers Do – Marcs Buckingham
- X-Teams – Deborah Ancona and Henrik Bresman