Reference Methodology for an Agile Digital Business

Version Summer-2021

Current Author

- Asanka Abeysinghe | Chief Technology Evangelist- WSO2, Inc | @asankama

Original Authors

- Asanka Abeysinghe | Chief Technology Evangelist- WSO2, Inc | @asankama (Summer-2018 - Summer-2021)
- Paul Fremantle | former CTO and Co-Founder - WSO2, Inc | @pzfreo (Summer-2018 - Spring-2020)

> This document describes a reference methodology for modern agile digital enterprises. Our goal is to move organizations from the current maturity stage towards the integration agile stage. The outcome of moving higher in the maturity model is to increase the productivity of employees, save costs by reallocating resources from a "center of excellence" to self-organized teams, and provide a better customer experience by being digitally aligned. The methodology for moving through these maturity stages involves people, process and technology.

Introduction

As APIs, microservices, software as a service (SaaS), and serverless architectures evolve, integration is not fading away; instead almost every new application involves integration across an exploding set of endpoints. The microservices mantra of smart endpoints and dumb pipes fundamentally addresses a deeper problem. Integration must learn from code to become immutable, type safe, testable, continuously built and deployed so they are more robust, resilient and above all - agile.

A Methodology

A methodology formalizes an approach to achieve a particular goal or set of goals. Our end goal is to make an organization integration agile. The proposed methodology is an iterative process characterized by five stages. Additionally, the methodology defined in this document looks at the organizational improvements in three dimensions.

The methodology does not expect an organization to start from scratch, but rather to first conduct an assessment to understand the current state and define a path to move to the next desired level. This paper outlines a meta-process for organizations to refer and become integration agile.

Maturity Model

A maturity model defines how an organization can measure the current stage of integration agility and defines a direction for improvement in three dimensions: people, process, and technology. There are two foundational elements for enabling the continuous improvement of an organization and move to the right in the maturity model:

- For both people and process the foundation is the culture of an organization.
- For technology the foundation is the architecture designed and implemented by the organization.

Summary view

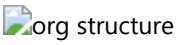
	Monolithic	**Fast Waterfall**	**API-Driven**	**Early Agility**	**Integration Agile**
People	Centralized	COE	API Teams	Decentralized	Self-Organised Teams
Process	Waterfall	Fast Waterfall	API Iterative	Semi-Continuous	Continuous
Technology	Silo	EAI/ESB	API-Driven	Early Agility	Continuous Agility

	Monolithic	Fast Waterfall	API-Driven	Early Agility	Integration Agile
Digital Alignment	Separate	Ad-hoc	Early Strategic	Digital-First	Adaptive

The maturity model defined here contains five stages, and each stage is defined in three dimensions. Let's look at each dimension in detail.

People

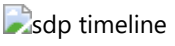
People and their ability to define solutions by understanding the current situation is the most important thing for an organization. Culture and the organizational structure control the way people operate, and it is the platform for enabling people's productivity. Culture plays an important role in adapting to change. Habits and best practices inherited from an organization's culture are more powerful than those merely enforced by policies.



The organizational and decision-making structure affects how innovative ideas from each part of the organization are implemented and delivered to end-users. Traditional divided pyramid, hub and spoke structures create a center-of-excellence (COE) mentality and can rapidly block the flow of innovation. As an alternative, we propose a podular organizational structure, which allows organizations to become adaptive, innovative and agile. People (organization & culture) must move from sequential waterfall and waterfall-agile approaches which include siloed "center of excellence" teams, to a fundamentally decentralized set of agile teams. In a podular structure, each of these teams then own the responsibility for building, running and managing their integration applications. One of the main characteristics of a podular organization structure is each pod act as an individual business unit [2] which allows to operate and take decisions independently as well as add the people with the required skills to the team.

Process

Process defines the steps an organization should take to achieve its goals. A process is connected with the people involved in its execution, as well as the technology being implemented to optimize their productivity. This document mainly focuses on defining a meta-process.



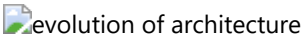
While many software development processes have been introduced, we can put them all into two buckets: waterfall and agile. To date, there have been many attempts to become truly agile, but it has not been an easy task due to several technical and cultural constraints. As a result, many enterprises have had to fall back to waterfall-agile or fast waterfall processes. However, now we are living in an era with of DevOps improvements and extremely flexible deployment infrastructures, which create continuous processes that allow teams to operate in a truly agile manner.

Technology

Although the reference methodology is fundamentally about people and process, it has technology requirements that must be met to enable the right processes and approach. For example, decentralized continuous integration/continuous development (CI/CD) with canary deployment requires a cloud orchestration model, such as Kubernetes. Agile development of integration applications requires integration tools that support continuous build, test and deployment. This is pushing a shift away from configuration-over-code approaches such as an enterprise service bus (ESB) towards code-over-configuration approaches that bring type validation and integrated development environment (IDE) integration and work more effectively with version control and CI/CD tooling. Integration applications must work in cloud native environments, including containers and serverless. Integration applications must be observable using distributed tracing and monitoring tools.

As we described earlier, architecture is the foundation for technology. Architecture references the maturity of the overall enterprise and integration architecture of the organization as well as the technology adoption. In the WSO2 Reference Architecture (RM) for Agility paper [1] we discussed three emerging architectural patterns; layered, segmented and cell-based, which can represent the enterprise and integration architecture implemented or planned in any organization. The

desired architecture pattern has a direct impact on the technical usage and how an organization can enforce an agile approach (iterative architecture) in practice.

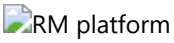


The architecture dimension represents the maturity of internal and external integration of the business, level of integration as well as how seamless.

Digital Alignment

Digital alignment is an outcome of the improvement of people, process and technology. Digital alignment mainly focuses on the maturity of digital transformation, both internally and externally, with employees and customers. Internally it affects how employees benefit from digital products and endpoints, such as APIs, events and streams, to increase their productivity and improve their decision-making process. Externally, it affects the experiences that customers and partners have with the digital products and endpoints focused on enabling an external transformation. In the current digital era, consumers are looking for a real-time, personalized, geo-sensitive and predictive experience from the products and the services they use.

Platform for Organizational Maturity



Together, culture and the architecture create a platform for enabling people, process, and technology to operate and improve. Culture plays the primary role in supporting people while architecture has the primary role in supporting technology. However, both of these foundations have overlapping and complementary roles in supporting all three pillars of digital alignment: people, process, and technology.

Maturity Model: Current outlook

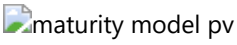
Agile Maturity Model: Matrix

	Monolithic	**Fast Waterfall**	**API-Driven**	**Early Agility**	**Integration Agile**
People	***Centralized***:Team is structured around a single project.	***COE***:Single or large development teams are controlled and governed by multiple centers of excellence (COE). Governance is complex.	***API Teams***:Teams are distributed; parallel projects are running, but dependencies and the execution model bring them back to waterfall.	***Decentralized***:Has distributed teams with centralized technology platforms and EA practices that create a COE.	***Self-Organised Teams***:Has decentralized teams and decision-making. Projects and systems connect by using the DevOps pipeline. Teams can rapidly and independently deliver projects.

	Monolithic	Fast Waterfall	API-Driven	Early Agility	Integration Agile
Process	***Waterfall***:ollows a waterfall process. Executes one project at a time.	***Fast Waterfall***:Follows a waterfall or spiral method. Has lengthy project release cycles. Is highly non-agile.	***API Iterative***:End-user applications and API development follow an agile process by dividing into small teams.	***Semi-Continuous***:Continuous processes are tied to the central platform. Has one build pipeline, which creates another COE	***Continuous***:Has a pipeline-first approach for every project and individual release pipelines for each team/project. Processes are fully automated.
Technology	***Silo***:Legacy and COTS* operate as silos. There is little or no EA practice. Data is aggregated manually.	***EAI/ESB***:Relies on legacy EAI and/or ESB technologies. Is highly non-agile	***API-Driven***:API management is used within organizations to streamline governance and provide decentralized discovery.	***Early Agility***:Uses lightweight ESBs with continuous integration and test, but still centralized, with manual discovery, governance, and other processes. Early adoption to Microservice architecture is occuring.	***Continuous Agility***:Has automated decentralized integration with APIs, streams and events published and discovered in federated registries, and CI/CD. Uses a cell-based architecture.
Digital Alignment	***Separate***:Business runs individually. Edge of the business is disconnected from management.	***Ad-hoc***:The business connects with partners and has initiated digital transformation. However, only internal consumers benefit.	***Early Strategic***:Provides a basic digital experience for the customer in a batch or near real-time manner.	***Digital-First***:Is connected with the ecosystem. Provides a multi-channel digital experience to customers in real time or near real time. Each consumer has a digital identity.	***Adaptive***:Offers a comprehensive multi-channel digital experience for consumers. Adapts based on consumer feedback and demand using analytics and AI.

(*Commercial off-the-shelf software. **Center of Excellence.)

Maturity Model : Pragmatic view



The maturity model has a definite vertical separation of each stage, but most organizations might find it hard to fit into one vertical pillar. As a result, an organization may align with three different horizontal stages as depicted in the diagram.

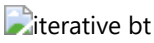
This approach helps an organization in planning the transformation and where to put more effort and focus.

For example, an organization might move to a complete cloud-native infrastructure with a mature CI/CD pipeline, but the team structure might not be podular enough to have a genuinely agile enterprise.

How to Move to the RIght in the Maturity Model: Methodology

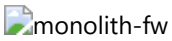
> ***" Transformation is a journey without a destination"*** - Marilyn Ferguson

Approach: Iterative business transformation



The overall approach of moving from one stage to another is iterative. Plan, implement, review, improve, and go back to plan. Also, start small by beginning with a small group, a single project, and one line of business instead of going for a company-wide approach. Skipping stages is dangerous because any change takes time for people to adopt and become productive. Therefore, minimizing change at each stage is an important factor in being able to continue business as usual while the transition is happening in parallel.

Monolithic to Fast-Waterfall



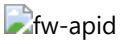
People: Organize development teams by introducing the basic software development methodologies described above. Incorporate governance by introducing a source control system and test frameworks. Code quality and design reviews can be manual.

Process: Introduce a primary software development process, such as the waterfall or spiral instead of using an ad-hoc development process.

Technology: Define an enterprise architecture using a layered architecture pattern. Categorize each layer based on the functionality, most organizations follow a "system of system" (SoS) [4] view when defining each architecture layer. Start connecting internal and external systems by using integration and messaging middleware.

Digital Alignment: Build internal applications and dashboards by utilizing the consolidated data gathered by integrating internal systems and partners.

Fast-Waterfall to API-Driven



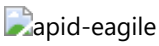
People: Build multiple project plans and groups by planning the execution of projects in parallel.

Process: Introduce agility by converting API and end-user application development teams into small teams. Use APIs as the connectors for these small teams.

Technology: Start an API program, and expose the core business capabilities created by integrating internal and external systems as APIs. Follow API design guidelines [5], and standardize the APIs in the industry. Encourage internal and external app developers to use the APIs when developing applications to consume business functions and data. Use federation, and move towards a segmented architecture from the layard architecture.

Digital Alignment: Encourage internal and external app developers to deliver creative and competitive ideas as a digital experience for consumers by utilizing the APIs. Allow partners to seamlessly connect using the APIs.

API-Driven to Early Agility



People: Distribute the decision-making and management to each project team. Provide technical capabilities as a shared service through a platform. Enforce governance and policies through the platform.

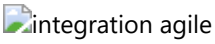
Process: Enhance the agile process introduced in the previous stage by bringing in continuous integration and continuous delivery (CI/CD) by associating with the DevOps team that manages the centralized infrastructure and technology platforms.

Technology: Move to a segmented architecture based on the scope of the services. Allow decentralized deployment whenever possible. Encourage the use of automated DevOps practices, and initiate continuous integration and delivery. Start using lightweight deployment environments, such as hypervisor-based virtual machines and containers.

Digital Alignment: Provide a unique digital identity for each internal and external consumer. Build a real-time and near real-time information exchange with consumers through digital apps. Extend the consumer's digital reach to multiple channels in addition to web and mobile by extending Internet of Things (IoT) capabilities.

Early Agility to Integration Agile

The primary goal of this paper is to help make your organization more integration agile. Hence we are emphasizing this section compared to the rest of the transition steps already described. Most organizations have an active effort to move to the right in the maturity model and fulfill consumer demand and stay on top of the competition. Best practices and the guidelines provided in this section helps the organizations to move right in the maturity model regardless of the current maturity level.



People

Agile-core

Because being iterative is fundamental to an integration-agile organization, culture transformation must also be handled iteratively. Forming a small group which we call the Agile-core is the first step in this process. The agile-core team is a critical success factor in the entire transformation journey of an organization. Therefore be mindful when picking people for this team. There are three rules to follow. First, pick the handful of people who accept change and adapt quickly. Secondly, pick the people who have the required skill set or can easily be trained. The third is to pick a diverse set of people who represent different levels and roles. The idea here to follow the train the trainer and let the agile-core to train others and transform the entire organization into an agile, digital workforce. Many approaches can taken in this transformation activity, such as structured training, boot camps, hackathons, lunch-and-learn sessions, and innovation labs.

Self-Organized teams

We noticed distributed teams with parallel running projects in the early maturity stages. However, distributed teams depend on various Centers of Excellence (CoE) when it comes to end-to-end delivery of a project. The motivation behind self-organized teams is to have a proper decentralized structure. A podular organizational structure provides the platform for creating self-organized teams and operating successfully. Self-organized teams plan, build, run and manage their applications independently. At the same time, the business functionality owned is exposed by using endpoints which expose data as APIs, events, and streams.

Everyone is an Agile Master

Most agile methodologies encourage having an agile master for each agile team. However, self-organized teams operate differently. Team members are engaged, empowered and entrusted to implement innovative ideas and deliver as digital products to end-users. Therefore every team member acts as an agile master in different stages and areas of the project, regardless of any hierarchical levels assigned by the organization.

Process

Iterative

An integration agile organization uses an iterative execution model in business and technical processes. It is easy to define iterative processes but hard to execute without a conventional culture and architecture. The sub-topics we discuss under Organization & Culture are vital to creating a supportive culture. Fundamentals of the iterative approach are about starting small while having a bigger vision and continuously iterating towards the target. Planning iterations that range between one month to three months is effective.

Continuous (Agile + DevOps)

The continuous process we describe in this document is a combination of agility and enhanced DevOps, which is part of development and utilizes cloud-native infrastructure. The combination of DevOps and development enables an end-to-end automated continuous process. This approach streamlines the release process and increases development teams' productivity.

Programmability is a critical factor for true agility. This approach goes beyond infrastructure automation, which is a common practice by enabling programmatic access to automate actions, procedures, processes, and runtimes used in an application's entire lifecycle. With programmability and end-to-end automation, teams can increase productivity and flexibility and quickly decommission repeatable tasks; they can also spend more time on implementing innovative ideas.

Code over Configuration

A configuration-based approach, coupled with the utilization of heavy middleware layers, is the primary blocker that prevents development teams from operating in a real agile mode. It is difficult to link configuration-driven development models with a continuous release pipeline and fully automated application lifecycle with multiple lifecycle stages (environments). Additionally, most of the middleware runtimes are not microservices and cloud-native friendly. As a solution, a coding-based approach that uses a programing language optimized for integration will allow teams to meet the expected agility and compliance with a continuous pipeline.

Integration First

The programming model has changed during the last decade to an integration-first approach. This is mainly due to the programmable and reusable endpoints that enterprises have created using various distributed architecture patterns, such as service-oriented architecture (SOA), event-driven architecture (EDA), and microservices architecture (MSA). As a result, every programmer is an integration engineer today. Moreover, the modern enterprise architecture is integration driven, with apps and services that each have tens to hundreds of composable endpoints.

Technology

Pipeline tuned

Developers spend much time building their sandbox environments and linking them to the source control system and build pipeline. Sometimes they do not follow standards due to manual configuration and linking. The concept of a pipeline ready is a solution for that which provides the sandbox environment and links to the development tools in a fraction of time using an automated process. Pipeline readiness increases the productivity of developers and enforces governance and development standards. Additionally, it makes the developers own most of the DevOps responsibilities from day-1 and maintain project specific CI/CD pipelines.

Multi environment based application lifecycle

The traditional development and deployment lifecycle of an application build is based on a minimum of three environments: development, testing and production. Some organizations have added a staging environment as a safe site to debug production issues. However, the iterative and rapid release nature of the self-organized team requires the addition of more environments to be productive, as well as have an infrastructure, CI/CD pipeline, and DevOps practice support. We are looking at adding blue-green, canary, and automated A/B testing environments to the application lifecycle.

Test driven development

Traditionally testing of an application is a secondary action executed by a separate group by creating another silo or a CoE. In an integration agile environment, the project team itself is responsible for delivering a high-quality application. (Remember, project teams plan, build, run and manage the application.) Therefore testing is part of the project team, which is executed by the developers. Automated tests (unit, performance, and integration) with test data and test systems (e.g., mock services) have to be created before the development and automated tests run against the code from day-one. Developers are required to enrich the test framework by introducing many test scenarios based on various business and technical use-cases (functional and non-functional). Some organizations call it the test grid, which strengthens the quality of the application.

Cloud-native

The heavily decentralized nature of a self-organized team and distributed DevOps responsibilities require a proper infrastructure to support integration-agile organizations. A cloud-native architecture based on containers and container orchestration systems helps development teams to utilize a company-wide standardized infrastructure to deploy and run applications and any dependencies. Using containers to autoscale, package applications, tune the pipeline, and support CI/CD by rapid environment provisioning are a few basic cloud-native capabilities that can be utilized by self-organized teams. Also, advanced features like event-driven and function-based architectures can utilize a cloud-native infrastructure to enhance the overall integration agility.

Cell-based architecture

Emerging architectural patterns, such as layered and segmented architectures, are centralized or depend on centralized CoEs. On the other hand, microservices defined in a microservice architecture (MSA) are too granular to treat as an architecture unit. Hence, self-controlled teams require a reference architecture to define the logical and physical architecture boundaries in the enterprise as a whole.

We introduced the cell-based architecture[1] as a solution for this problem to have a microservice and cloud-native architecture leading enterprise architecture to utilize in an integration-agile environment. The cell-based architecture enables the conversion of legacy systems, services, and data into cells, and it reuses the functionality in combination with the cloud-native cells. In a nutshell, the cell-based architecture makes it possible to build on brownfield development.

Open source

In an integration-agile environment, you expect to do many innovations in labs (note that research and development is part of the self-organized teams), as well as rapid application development by following iterative execution model. In such an environment, you cannot afford to wait for lengthy (and slow) procurement processes to adopt the technologies that will allow you to deliver on your innovative ideas. Open source plays a significant role here by giving you access to robust and stable technologies that are used and contributed by a broader community. You can pick technologies delivered under business-friendly open source licenses, such as Apache 2.0, to avoid any corporate red lines in the future.

The above paragraph explained only one side of the effect of open source for an organization, which is the usage of open source and increase productivity. Contributing to open source and creating an open culture with open source practices are two other aspects.

Contributing back to the open source project mandate by some of the open source licenses but more than that contribution is a usage ethic of open source. Organizations can create ecosystem teams and contribute to open source projects based on relevance. There are many advantages to this approach. Learning from a wider audience outside the organization and increasing the usage of the code the ecosystem teams write are few advantages that can highlight.

Isolated teamwork within a business or functional unit can convert into collaborative development work using open source practices. Most of the organizations start with innersourcing (a word coined by Tim O'Reilly in 2000) - collaborative development across the teams, but within the organization is the main characteristic of innersourcing. Furthermore, the

code is not available in a public repository for outsiders to access or contribute. The second stage is open source that the development and access to the code are available for outsiders but most contribution coming from the internal community. The third stage is an open community that the project donated to an independent foundation such as Apache Foundation (ASF) or Cloud Native Computer Foundation (CNCF). Governance of the project handled by the foundation and contribution can come from anyone using or willing to commit to the project.

Digital Alignment

Consumer-driven requirements

One of the critical characteristics of a podular architecture is bringing self-organized development teams to the edge of the business and providing access to consumers. Minimizing the gap between the consumer and the producer helps to implement solutions that are actually required by the consumers without working on hypothetical requirements that trickle down through many CoEs. The development teams can have a direct dialog with the consumers using various channels, analyze consumer behavior, and review the market outlook when identifying and prioritizing business requirements. Small iterations and rapid application development help self-organized teams to deliver a feature quickly and test run it with the actual consumers (including the rollback of a feature). Technical teams tend to complicate the requirements by looking at it from a technical point of view. In an integration-agile environment the requirements teams look at are from the consumer (or business) point of view and simply based on the consumer experience and competitive advantage.

Start with a MVP

The consumer-driven nature and business first culture result in rapid changes to the requirements and the user experience. Consumers are eager to consume new digital experiences and make themselves productive during day-to-day activities. Once they don't get the experience they are looking for, they switch the service providers. Also, there is a competitive advantage in introducing a new idea to the market before competitors provide the same. Planning, implementing and launching a minimal viable product (MVP) to the market is the best way to stick to the iterative execution model, rather than having long product delivery cycles that follow waterfall or fast-waterfall methodologies. Self-organized teams get the foundation and platform by staying at the edge, close to consumers to understand them correctly and define the MVP.

Deliver digital-native applications

As we explained earlier in this paper, consumers in the modern digital era come with certain predefined expectations. To reemphasize the current digital expectations, these include real-time, personalized, geo-sensitive and predictable experiences from digital applications. Disruptive applications released from the self-organized teams have to be digital-native to win the market and provide long-term services for the consumers.

Enforce feedback loops

A useful iterative execution model requires feedback coming from usage and runtime behavior. A cell-based architecture enforces the cell gateways to route communications and provide enough hooks to capture the data required to identify feedback about usage and the behavior of runtimes. Constructive feedback has to be considered when planning the next iteration in order to improve the user experience and runtime behavior.

Transformative purpose

Defining the transformation strategy based on a purpose helps to run a successful transformation program as well as provide results align with the business. Different frameworks can use to achieve this, most popular frameworks are OKR (Objectives and Key Results) and MTP (Massively Transformative Purpose). Defining the purpose and measurable results is an exercise the digital strategist or the steering committee required to conduct before executing the activities of the transformation strategy. Having a thorough analysis of the internal priorities, customer demand, competition, and market outlook help to define the transformative purpose. The maturity model can use to start the process by conducting an assessment as well as a guide during the transformation journey.

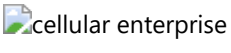
Conclusions

Digital transformation is the critical success factor of the modern enterprises, and industry leaders have identified digital transformation as the fourth industrial revolution. Therefore, organizations have to plan and execute the transformation without losing their existing people, systems, data and customers. However, integration has become an essential factor in application development with the explosion of consumable endpoints we created in last two decades by building highly distributed systems. Organizations are trying to follow agile execution models for their integration but fall back to the traditional waterfall or spiral models due to non-alignment of the organizational structure, culture, architecture, technology usage, and digital strategy of their integration teams. This paper outlines a maturity model for organizations to identify where they stand with regard to their integration agility, and to help plan a transformation path to become truly integration agile. The content gathered and offered is based on our experience working with many enterprises on their integration and digital transformation initiatives.

Highlights of the next version

Cellular enterprise

Using biology as a metaphor to define modern organization structure build with a podular architecture consist of many self-organized teams. Cellular enterprise defines units associated with an application from the human who designs and builds to a composite that deploys at the end. Refer the Forbes article for more details.



Cell Type	Description	Unit	Mapping
C-Cell	Composite Cell	Composite	Decentralized deployment
D-Cell	Development Cell	Development sandbox	Autonomous development
A-Cell	Architecture Cell	Architecture block	Cell-based Architecture
B-Cell	Business Cell	Business function	Cellular business model
H-Cell	Human Cell	Self-orgenized teams	Podular organization (team of teams)

References

[1] WSO2 Reference Architecture for Agility paper: https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

[2] Dave Gray - The Connected Company - http://www.xplaner.com/connectedco/

[3] Iterative, Segmented Architecture - https://www.slideshare.net/asankama/iterative-architecture-your-path-to-ontime-delivery

[4] System of Systems (SoS): https://en.wikipedia.org/wiki/System_of_systems

[5] API Design Guidelines: https://wso2.com/whitepapers/wso2-rest-apis-design-guidelines/

[6] WSO2 Agile Maturity Assessement: https://wso2.com/agile-maturity-assessment

[7] WSO2 Impulse (Strategic Consulting): https://wso2.com/strategic-consulting/

[8] The Celleular Enterprise (Forbes article): https://www.forbes.com/sites/forbestechcouncil/2020/06/29/the-cellular-enterprise/#786824316832