The Agile Swab: Testing for Health and Maturity [1, 2]

Bill Sundermann

Abstract

Agile process maturity is a reflection of teams executing solid practices and maintaining good habits. Developing Agile methodology is an iterative process, but along the journey assessments are key to reaping the benefits. The "virus" of bad habits and shortcuts undermines the benefits and value stream.

Project Managers and Agile leaders should plan for, execute, and monitor health-check activities and build them into the project plan. Through a case study of Enterprise Architecture team leadership in new product development, we'll review challenges encountered in successfully implementing these reviews.

The result was a delivered cloud-based banking platform, providing clients with the agility and flexibility needed to meet rising expectations of bank customers. A set of flexible, scalable core banking product processors were built on a common platform with open, modular components using mature Agile methodology. The new platform and product generated higher sales to existing clients wanting to go digital and to net new clients attracted to the configurable, extensible, and scalable technology.

A "swab" to detect unhealthy practices is key to promote faster development cadence and heightened self-awareness for a pandemic-induced remote work force transforming the workflow and project management process.

Introduction

Until two years ago, the term "swab" was mostly confined to medical laboratories. The COVID-19 pandemic brought the word into everyday jargon as we were tested for a virus that changed

[1] *Editor's note: Second Editions are previously published papers that have continued relevance in today's project management world, or which were originally published in conference proceedings or in a language other than English. Original publication acknowledged; authors retain copyright. This paper was originally presented at the 14th UT Dallas PM Symposium in May 2022. It is republished here with the permission of the author and conference organizers.*

[2] How to cite this paper: Sundermann, B. (2022). The Agile Swab: Testing for Health and Maturity; presented at the 14th University of Texas at Dallas Project Management Symposium in Richardson, TX, USA in May 2022; republished in the *PM World Journal*, Vol. XI, Issue X, October.

the way we live, work, and socialize. Similarly, Agile "swabs" can be taken to provide test results for project health and maturity.

Unlike a one-time test to detect the presence of a virus, software testing occurs along the entire development and release process from unit test through final user acceptance. There are many places where an "illness" can be detected.

Agile processes have focused on the practice of shifting left to find and prevent defects early in the process. While a team can claim to "being agile" there are points where improvements may have stalled or where bad habits have led to a stagnant environment.

This paper reviews how, when "symptoms" of poor health are encountered, Agile teams can take a "swab" to isolate improvement opportunities and assess the overall level of adoption maturity.

Part I is a case study of the development and delivery of a modern banking platform and a CICD pipeline. This project is the foundation for detailing challenges faced, hurdles overcome, and specific areas where an Agile swab can test for issues and point to remedies. A focus on these improvements leads to positive outcomes that teams strive for and sustains the Agile transformation journey. Part II covers seven areas for Agile swabs.

Part I: A Digital Transformation Journey Case Study

FIS is a leader in banking and payments technology. Tracing its roots back to Systematics, the fifty year-old company has been at the forefront of financial services software through both internal development and strategic acquisitions. A primary line of business is core processing and the company offers a portfolio of solutions to large banks, community banks, and credit unions. During the last ten years, the company has faced increasing competition from fintech startups such as Chime, Varo, OnJuno, and many others offering either advanced mobile banking, related payment services, or both.

FIS responded with a large investment as commitment in digital transformation to meet these requirements from existing customers and prospects:

- Proven delivery excellence
- Single-source provider
- Commitment to current compliance
- Cloud deployment capabilities and flexibility
- Ease of integration with internal and 3rd party applications
- Speed to market
- Personalization of product offering
- Modernized interaction with bank customers

- Resiliency and Scalability
- Address data security complexities

From these requirements, FIS began development of a modernized banking platform featuring:

- Real-time customer experience
- Ability to personalize products
- Cloud advantages – lower cost, elastic scale, pay per use
- Flexibility to add new components, including 3rd party vendors
- Component-based architecture – self-contained, configurable for:
  - Customer
  - Product Management
  - Account Engine
  - Real Time Data Hub
  - Compliance
- API-first – core components exposed as APIs and available externally
- Cloud native – security, monitoring, resiliency, operation analytics, regulation
- Unified design principles that enable upgrades and modernization
- Open source to reduce cost
- Security at all levels in all forms

A primary objective was to move away from legacy monoliths that are perhaps the most common cause of technical debt in organizations. These traditional ways of architecting applications make it difficult for teams to deliver on the DevOps promise to accelerate value into the hands of customers. CI/CD can only be effective if it can build, test and deploy small components at speed.

A Next Gen Banking Development organization (also known as Enterprise Product Office (EPO)) was formed to build a new platform from the ground up. FIS eventually branded this offering as the Modern Banking Platform (MBP).

For the broader FIS development groups, the adoption and evangelizing of Agile principles was taken up by a dedicated team taking the moniker Agile League.

Within EPO, a team of enterprise architects was formed for the purpose of establishing a modern, common infrastructure for the MBP banking platform focused on the customer requirements and FIS solution sets listed previously.

Clients were demanding MBP solution sets to be readily compatible with public cloud providers such as Amazon Web Services (AWS) and Microsoft Azure. The Architecture team translated these requirements into a high-level product vision, or target state, that would provide the

scripted templates to procure the infrastructure and set up the target environments. Of highest priority was security, high availability, and disaster recovery processes needed by financial institutions.

Feature Set

With a long history in developing and implementing banking products and services, FIS laid out a capability grid as a basis for MBP design and as a blueprint for building the service-oriented architecture. The result was a Requirements Traceability Matrix that served to map the specifications to the delivered product and identified feature relationships among components. The ability to deploy standalone components to a cloud environment through a CI/CD pipeline was at the heart of the effort.

The high-level development strategy was to deliver standalone components for retail deposits and enterprise customer applications (both core and UI) such that the system could be deployed separately in the case where a client needed one or both components. These components would be self-contained and configurable. The product set would be flexible to include the ability to add new components, third-party applications, architected to expose as APIs.

An open-sourced architecture included JBoss application server and database certifications for both Oracle and Postgres. To build a deployment pipeline, each component would be in a Docker container, bundling all application dependencies and providing the ability to run the application in any environment. Container orchestration would be accomplished through Openshift, Red Hat's value-added Kubernetes. Helm charts were used to manage related sets of Kubernetes resources.

Project Team Organization

Organizationally, Development (DEV) scrum teams created for Retail Deposits and Customer would develop the services based on the product requirements outlined in the Requirements Traceabiliy Matrix.

Concurrently, the Enterprise Architecture (EA) team would define the patterns for building the container (the Docker image), deploying the container, and managing the container in the private cloud environment. These patterns were shared with DEV teams through training provided to create the Docker images for the components.

Other Company Stakeholders

As a large Fortune 500 company with 62,000 employees serving global clients using a vast number of internally developed products and acquired applications, FIS had an expanded internal network structure of technical support, governance, and administrative offices that constituted

primary or secondary stakeholders in the MBP product development. This section provides information about these stakeholders and the project complexity impact to the Enterprise Architecture (EA) team on planning, resource scheduling, initiating and controlling, and monitoring:

Leveraged Services. Initial development and deployment utilized the FIS Private Cloud managed by the Leveraged Services team. All requests for new instances of OpenShift were submitted through this team. As new users themselves, occasionally there were delays in setup and instances where the RedHat vendor would need to be contacted to answer questions.

Network Security. Any changes to firewalls or security protocols were requested through the internal FIS team.

Compliance. As a provider of software services to the financial services industry, the FIS Corporate Compliance team was frequently consulted with regard to new product development and adherence to all regulatory matters.

Product. As the MBP product development matured, the Product team responded to high-level questions regarding product features. Demos were requested which involved standing up new environments to show deployment flow and functional capabilities.

Sales/Marketing. The team that would be selling the product to new and existing customers required meetings to answer questions posed by future users of the systems. As a non-technical team, they also were dependent on EA to set up client demos, frequently on short term notice.

Client Services. This group had responsibility for supporting existing clients operationally if they were running separately in FIS data centers or in an outsourcing service provider relationship. As development progressed, they required responses to pre-production checklists, reliability data, security validation, and scalability resources.

Security. Identity and Access management for MBP was handled through a home-grown IdentityProvider (IdP). This team supported the set-up user id's, passwords, and product level authorizations for all test environments. This team also conducted penetration tests requiring responses and remediation.

Corporate Systems. The NextGen Banking group was dependent on Corporate Systems for creating Jira projects, Confluence wikis, making modifications to workflows, adding issue types, and general support of Atlassian products and tools.

Finance. Funding for IDSW (Internally Developed Software) was reviewed and approved by Finance. At the beginning of each project, Finance required documentation submitted for project

funding. This included a Project Definition Document (PDD) and a financial analysis of costs and benefits.

Enterprise Project Management Office (EPMO). Project governance required the preparation of specific artifact documents and a project audit was conducted to verify all information was present in a Sharepoint site created for each project.

Corporate Time Entry. Related to Finance, all FIS employees were required to enter hours worked on their IDSW projects. Individual Project Managers establish time entry "buckets" within the Planview tool and allocated the team members to these categories. Actual hours were tracked against the budget and a forecast of remaining hours was provided to Finance.

Vendors. FIS systems utilize applications from outside vendors to manage infrastructure and provide specialized services such as code scanning and security scanning. For example, MBP adopted OpenShift to handle container management. Support was needed from the RedHat vendor technicians to assist with issues encountered with configuration and performance in the FIS private cloud.

> All of these stakeholders were integral to the development of the Modern Banking Platform, but also contributed to the complexity of the transformational project. As with all Agile software technology development projects, added stories and tasks tied to multiple stakeholders can impact release timelines.

Scrum and Scaled Agile Framework (SAFe)

The DEV project teams adopted a scrum framework based on the need to deliver increments of working software and build the platform iteratively. The Deposits and Customer teams each contained multiple Scrum teams that were organized according to functional areas – UI, transactions, database, etc.

The EA Scrum team worked on platform infrastructure in support of the DEV products. The team established 4-week Sprints that aligned with the DEV teams. A daily scrum meeting was held and other Scrum ceremonies were conducted, including an end-of-sprint Demo where the EA architects would showcase working code to the DEV teams and other stakeholders.

Initially, the EA team focus was on 1) developing the patterns to containerize and deploy the components in CICD pipeline 2) integrating a Data Protection Framework (DPF), and 3) integrating an Identity Access Management (IAM) that was FIS proprietary (IdP).

After completion of the architecture to deploy the MBP to a cloud environment in a safe and secure framework, the EA team compiled an Architecture Roadmap to define the path for future development and improvements. The Roadmap was derived from architecture enhancements

deemed essential by the strategy leadership as well as modernization captured from meetings with existing and potential clients.

Part of the challenge for the EA team was integrating the patterns developed from new technology into the sprint plans for the component teams. At the beginning of each iteration, the Deposits and Customer teams would already be fully allocated with feature development and bug fixes. The solution was the development of a Reference App and staged adoption discussed below.

Development and Architecture Standards for Code Management

To achieve the objective of frequent delivery at a sustainable pace, the DEV organization needed a code management tool designed for simultaneous feature development with the ability to commit, merge, and release code across multiple applications in a unified manner. Git was adopted as the code version control system, replacing Subversion. Along with Bitbucket as a code repository management solution, Git provided a method to create feature branches from the primary (master) branch.

Two main workflows, Master and Develop, provided the best structure for creating feature branches off of Develop and the added step of a Pull Request to do code review and quality scans. After the pull request was approved, the commit is to the same branch and task was marked complete.

Work was split up into feature branches. This provided a strict quality gate between code developed and code getting back to the main line. The pull request ensured a secure gate and controlled when changes were migrated. Agile practices facilitated controlling releases in a sprint timeline. For example, if 30 feature branches were created representing stories and only 20 are ready at the end of the sprint, then pull requests are created only for those 20.

Technology Injection Lifecycle Utilizing a Reference App

As the MPB release cycle matured, there were ongoing technology updates included in the Architecture Runway for each release. These stories related to updates to Oracle, adoption of the Spring Framework, adoption of Flyway database migration tool, to name just a few.

The EA team constructed a standalone Reference App to develop and test the patterns that the core component teams would adopt. The Reference App was intended as a mirror image of each component. In practice the effort to maintain the Reference App to be exactly like the component apps was time and cost prohibitive (many developers from the components would need to be involved to keep all code current). To introduce advanced technology to the components, a 3-stage process was created using an Alpha, Beta, and Gamma release process:

Technology Refinement – this was the client view of the process to establish technology patterns for adoption and included Pre-Alpha – technology analysis and planning.

- Alpha – Defined the completion of each iteration of this technology introduction phase.
- Beta – Defined the completion of each adoption of the Alpha by the architects assigned by the component team with the intent to provide on-going engagement, education, and adoption.
- Gamma - As Development teams finalized Beta testing, new technology was integrated into the advanced level testing needed for Integration, Regression, and Performance Testing. Additional testing included compatibility with external applications.

FIS grew a scalable Agile development team for delivering its Modern Banking Platform through a CI/CD pipeline. Upon adopting Agile, companies implement practices that adhere to the principles for continued improvement. There is no final goal or end state, but rather the commitment to a sustainable pace and the practice of reflect and adjust to become more effective in the future.

Part II: Agile Swabs: Ensuring Healthy Agile Practices

If asked about the status of their Agile journey, teams may report that the culture has changed, the Agile principles are being followed, and that improvement is evident. Still, there are areas where commitments may have weakened over time, shortcuts have been taken, and habits have formed that can lead to limiting the potential of the team and introduce infective agents that disrupt the healthy Agile process. Just as a test for the presence of a virus involves taking a swab and testing for the presence of a problem, an Agile Swab tests specific areas that can detect the presence of "viruses" causing the symptom, impacting the health of the work, and affecting the maturity of the process.

Following are seven Agile Swabs that can be taken in areas of the project lifecycle where symptoms are presented indicating an unhealthy or immature process that can be improved.

1. Definition of Ready

Symptom: Stories committed to in the Sprint are not completed, work is done on features outside the scope of the sprint, and additional downstream work is needed during testing.

Agile Swab: Review the team agreement on the Definition of Ready. Perhaps the description is complete, but there is not sufficient information on how the feature will be tested or demoed. The description needs to include how the feature can be tested, including an acceptance test. A description of how to demo the story ensures that all flows have been considered. Estimations are necessary to expose any risk the story might not be completed during the sprint.
Additionally, all dependencies related to another team providing part of the work, both subtasks and delivery dates, must be included. Requiring external dependencies be resolved before a story is included in the iteration should be included in the Definition of Ready. However, strict rules that prevent overlapping development across teams could impede progress.

2. Definition of Done

Symptom: Defects are identified in higher level testing that should have been caught in unit testing. Stories are not providing the guidance for pre-implementation activities: discussion, estimation, design.

Rework is required after a feature has been accepted as "done" and there is misunderstanding and conflict between the development team and the customer or product owner related to what will be delivered.

Agile Swab: Completion Criteria should be defined at the organization level, not just by individual teams. This avoids misunderstanding and conflict regarding the definitions of "Code Complete", "Unit Tested" "Peer Reviewed", and "QA Complete". Any needed documentation tasks are determined by the Scrum Team at the beginning of the Sprint. With the guidance of the Scrum Master, the Development Team determines when the Completion Criteria have been met. At that point, the story is considered complete.

The Acceptance Criteria are expectations for a specific Product Backlog Item as defined by the Product Owner and should be established prior to the beginning of a Sprint. Assistance from the Development Team and ScrumMaster may be needed. When the Development Team meets these Acceptance Criteria, the Product Backlog Item is ready for a Product Owner review in the form of a Demo. This approach allows enough flexibility and is adaptable around the definitions

of "Code Complete", but clearly delineates the roles and responsibilities associated with delivering and finalizing work on features.

3. Working Agreements

Symptom: Interactions between team members is limited and there are disagreements regarding responsibilities and assignments. Interactions with other teams is disjointed and multiple different messages are communicated on status, impediments, and required inputs. The team is consistently unable to deliver on their commitments, impediments are not surfacing, and there is lack of shared responsibility.

Agile Swab: Working agreements capture the shared understanding of a team in terms of how the individuals and the team as a whole want to work with each other and with their stakeholders.

It is a Scrum Master's responsibility to ensure a team has co-created working agreements. It is also the Scrum Master's responsibility that a team revisits and improves their working agreements over time.

Working agreements are an essential ingredient in any team's journey towards more and more self-organization – in short, they answer the Who? Why? How? and What? questions
In psychologist Bruce Tuckman's model of team development, the Forming stage is where goals, timelines, rules and regulations, and roles are discussed. Is it important that the Working Agreement be developed here to facilitate how team members will communicate internally, how collaboration tools will be used along with the expected level of detail, and how the team will interact with external teams.

Many Agile teams develop a Working Agreement, then find it necessary to refine it when situations highlight an incomplete description of roles. This includes who will respond to outside requests for information, steps for acceptance of work into the team, team meetings and format, and escalation procedures.

In their book *Agile Retrospectives, Making Good Teams Great*, authors Esther Derby and Diana Larsen provide a helpful team group activity for establishing a set of behaviors that will be supportive of team productivity. In the retrospective, the agreements proposed and prioritized should be new behaviors or those that are not yet part of the team dynamic (Derby/Larsen p.47) This exercise should answer the question "What do we need to start doing?".

4.Metrics

Symptom: Metrics for team performance tend to modify team behavior to optimize that metric. Examples are teams informally handling defects rather than logging into the tracking system or cutting back on productivity to more fully scrutinize every piece of code.

Agile Swab: A review of reporting needs will define the metrics to capture. If the metric does not provide information that leads to actionable steps, its collection should be discontinued. Any effort expended to capture unnecessary data is wasteful. Don't just measure a process or point in time because you can, measure to learn, identify actionable recommendations, and fix areas needing improvement. Review what is measured and the level of effort. The real purpose of measuring is to reduce uncertainty (Cohn, p 429). Measurements don't need to be exact to provide value. If the effort to take the measurement exceeds the value of the information it provides, different approaches should be adopted.

For example, the standard burn-down chart provides a view of remaining effort, but it can potentially obscure how well the team is focusing on incremental completion of business value. A useful metric is the adoption of the Agile-V Balanced Scorecard Metrics that Guy Beaver described in an article for AgileConnection (Beaver). With its emphasis on business value (as opposed to deliverable status), this scorecard offers a powerful approach to providing daily readout to management:

> *The burn-down combined with the burn-up provides a simple, powerful and accurate "Driver and Outcome" scorecard. By focusing on both, the team is provided an Agile Scorecard that shows both capacity utilization and team focus. If time passes with no stories completed, the team should ensure that it is utilizing the daily standup meetings to synchronize and focus on as few stories as possible, so that business client priorities can be worked to completion in the correct order. The ideal burn-down and burn-up charts, paired side-by-side resemble a "V" hence the name "Agile-V Scorecard"*

A Balanced Scorecard is also useful to provide upper management with a perspective across the organization in the areas of financial, customer, business processes, and learning and growth with a goal to measure desired outcomes and predict drivers of those outcomes. For a properly implemented agile team, this line-of-site measurement happens naturally and is controlled daily (Cohn, p 439).

Michael Cohn advocates for the four perspectives suggested by Liz Barnett for Forrester Research (Cohn, p.439):

- Operational excellence – how well is the team adding to productivity under time and cost constraints.
- User Orientation – how well is the team delivering features and technology desired by customers
- Business Value – how is the team helping with cost savings, additional sales, and technical capabilities.
- Future Orientation – how well is the team building skills and capabilities to inject new technology.

In short, metrics should be used to focus effort and aid understanding, not place blame or enforce compliance (Cohn, p.444)

5.Product Owner Commitment

Symptom: Product Owner and Sales Teams overcommit to customers in terms of capabilities of the system, feature set of the product, and delivery timeline. This results in forced changes to the Product Roadmap, disruption to the Program Increment (PI), and work overloads.

Agile Swab: At high levels in the organization, the principles of Agile and Scrum need to be communicated. Teams that are client-facing need to understand and commit to the practice of adding new requirements to a Product Backlog to be reviewed, prioritized, and slotted for development work. The Product Owner needs to commit to Development Teams to be accessible. It is the role of the Product Owner to come to sprint planning meetings to identify the conditions of satisfaction, or acceptance criteria, for each high priority user story in the product backlog (Cohn, p318). Explaining stories from a business perspective helps the team to understand the value and to build test cases that align to the customer requirements. Working Agreements should address the interaction between the team and outside product support teams to ensure that added requests are handled properly and do not disrupt the sprint flow.

6.Knowledge Waste

Symptom: The same lessons learned keep appearing on sprint reviews or retrospectives. There are no documented solutions, resolution ownership is not assigned, and valuable information is not recorded and made available. New team members struggle to get up to speed and experts are repeating themselves when training developers. The amount of time expended on tasks peripheral to committed Stories is excessive causing slippage in completion and late projects. Bug fixes are found to be rooted in a failure to add automated tests that would have detected the issue earlier.

Agile Swab: With dispersed team members, remote work normalization, and increasingly complex system solutions, the need to capture knowledge is critical and needs to be eliminated. Knowledge waste refers to either lost opportunities to learn or learning less than what could have been gained from a situation (Cohn, p213).

Knowledge waste can be divided into 3 categories (Allen Ward, 2007):

- Scatter – When anything breaks the flow of work. Look at any new rules introduced or standardized reports required that could be covered by clarifying communication requirements on impactful changes. Responsibility for results, not process adherence should be the goal.

- Handoff – Separation of knowledge, responsibility, action, and feedback. Embrace whole-team responsibility. By eliminating hand-offs, problems created by waiting and by the need to transfer knowledge from one person to another are eliminated.

- Wishful Thinking – making decisions without adequate information to support those decisions. Discarded knowledge is the failure by teams to capture acquired knowledge in useful formats.

Addressing knowledge waste is a direct investment in added productivity and employee satisfaction.

7.Retrospectives

Symptom: The Sprint retrospection after each sprint produces little feedback from the team and improvements to be implemented are not agreed upon, documented, communicated, or tracked. The format and objectives are poorly communicated leaving participants with no incentive to fully participate.

Agile Swab: The Retrospective is a key ceremony in a successful Agile transformation. This is the opportunity team members have to reflect on significant events that have occurred since the previous meeting, to make decisions aiming at remediation or improvement, and inspect/adapt teamwork methods. Scheduling a Retrospective is easy. The skill is in the preparation needed to solicit engaged participation and valuable feedback. To restore the retrospection to healthy and productive interaction, the leader should review the process and commit the effort to making the meeting a success.

In their book *Agile Retrospectives: Making Good Teams Great*, Derby and Larsen provide an example of the meeting structure for a software development team:

1. Set the stage – introduction, invitation for everyone to speak, outline the approach, and review working agreements.
2. Gather data – share hard data, review metrics, and post timeline.
3. Generate insights – ask Why? What? and How? (more powerful than Who? When? and Where?)
4. Decide what to do – list and plan potential experiments and improvements; sign up and commit to tasks.
5. Close the retrospective – decide on steps for retaining learning; express appreciation.

The five steps in this structure helps the team understand different points of view, follow a pattern of thinking, provides a comprehensive view, and allows for open discussion. Most importantly, the Retrospective should be structured to fit the distinct team dynamic.

Agile Swab and Agile Assessment

Companies adopt Agile principles to improve. Improve quality, improve speed, improve team building – all for the objective of improving value measured by customer satisfaction.

Investing time and people to the effort naturally spawns the questions that need response – How are we doing? Is our Agile adoption working? Are we doing better and by how much? Where should we be focusing? What is our direction going forward?

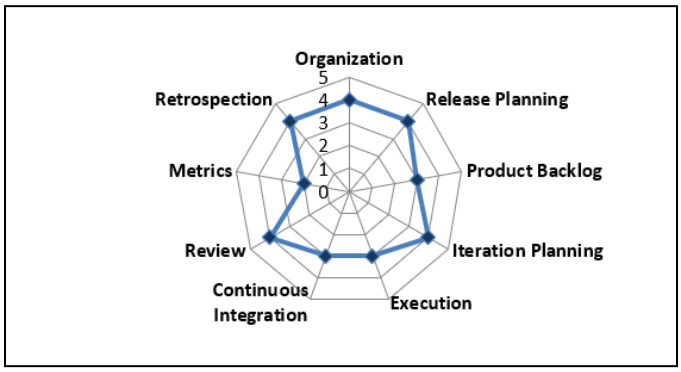
If the Agile culture has been in place for a sustained length of time, but results are below expectation, the Agile Swabs mentioned above can be used to take samplings from the process, review and examine, and initiate improvements to further advance health and maturity.

There are a variety of Agile assessment tools available. An Agile assessment doesn't need to perfectly gather data and measure performance. As Cohn realized after working with his clients, when asked "How are we doing?" what they really wanted to know is "How are we doing compared to our competition?". Being more agile than the competition is indicative of the organization's ability to deliver better products more quickly and cheaply (Cohn, p.434)

At its best, the team should develop a self-coaching tool to assess Agile adoption maturity. Agile fits best into companies that want to affect change, and the tool should provide data and feedback on the areas targeted for change.

Many organizations have adopted a simplified assessment tool that provides a web-like diagram that shows a maturity ranking based on Yes or No answers to questions tied to iterative practices such as:

- Organization
- Release Planning
- Product Backlog
- Iteration Planning
- Execution
- Continuous Integration
- Review
- Metrics
- Retrospection



While helpful in gathering a snapshot view, the format of simple Yes/No answers and the undocumented responses lends itself to a quick management update and not a definitive Agile health indicator. The owner of the assessment should sample some of the responses and collect corroborating evidence of reported practices.

An Agile assessment should be customized to incorporate the activities of the team and should vary over time. Experience shows that having multiple sets of questions, each focusing on a different aspect of the development process provides the most useful results (Cohn, p.434).

Summary

One of the principles of the Agile Manifesto states: *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*. Using the Agile swabs to detect unhealthy practices, teams can take corrective action that ensures the development process can respond to change and continues to mature.

Reference List

Cohn, Mike. 2010. *Succeeding with Agile*. Addison Wesley.

Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin, 63*(6), 384–399. https://doi.org/10.1037/h0022100

Beaver, Guy (March 8, 2007). The Agile-V Balanced Scorecard Metrics. *The Agile Connection* https://www.agileconnection.com/article/agile-v-balanced-scorecard-metrics

Ward, Allen C. 2007. *Lean product and process development*. Lean Enterprise Institute.

Derby, Esther, and Diana Larsen. 2006. *Agile Retrospectives: Making Good Teams Great.* Pragmatic Bookshelf.

About the Author

Bill Sundermann

Texas, USA

Bill Sundermann, PMP is a team builder and Agile coach with 27 years of experience in the financial services technology industry. He holds an MBA from SMU and is a certified PMP and SAFe Release Train Engineer focusing on helping teams in their Agile journey. His self-care routine includes 5Ks, bike rallies, pickleball, and domestic botanical upkeep.