Applying Agile Principles to BPM

Mark von Rosing, Joshua von Scheel, Asif Qumer Gill

INTRODUCTION

The term "Agile" has attracted significant attention across industry and academia.[1] Agile is not new. The history of agile concepts can be traced back to 1930s. It has its foundation in iterative and incremental approaches. Many ways exist in which agile concepts can be applied across various disciplines and industry verticals, such as agile software development, agile project management, agile supply chain, agile manufacturing, agile service management, agile enterprise, and the list goes on. Similarly, agile concepts can also be applied to business process management (BPM) planning, analysis, architecture, design, implementation, operation, monitoring, and improvement. However, before jumping on the bandwagon of Agile BPM, it is important to understand what is meant by "Agile." What are the building blocks or principles underlying agile? What does it mean to use agile principles? What is the difference between agile and traditional non-agile ways of working? Why do we need to be agile? How is an Agile BPM capability established? The purpose of this chapter is to provide the precise and practical answers to these fundamental questions. This chapter is organized as follows. Firstly, it describes the agile thinking and its origin. Secondly, it describes the agile characteristics, values and principles. Thirdly, it describes the agile practices or ways of working. Fourthly, it describes the difference between the agile and traditional ways of working. Fifthly, it describes the application of agile ways of working to BPM and defines the Agile BPM. Sixthly, it discusses how to establish an Agile BPM capability by using the agility adoption and improvement model. Finally, it concludes the chapter with key take away points.

WHAT IS AGILE?

Although the basic "agile" term comes from the Latin word *agilis* and means to drive, do, and see. The basic meaning of agile is to move quickly, lightly, and easily. In the 1930s, the automobile industry introduced the first agile concepts through the introduction of optimization concepts and work splitting. Further, agile concepts have been applied within the lean manufacturing/lean consumption paradigms. With Agile's growing popularity, other industry segments started realizing that agile principles are not limited to any specific industry segment or functional group. Most relevant to this discussion, over the past decade the software industry has successfully adopted agile principles, and Agile has become a popular software project and product development methodology. Agile methods and practices can be traced back to the incremental software development methods as far back

The Complete Business Process Handbook. http://dx.doi.org/10.1016/B978-0-12-799959-3.00027-6

as 1957[2] before falling out of favor for the heavyweight waterfall method. More recently, the agile movement began to come back when, in 1974, a paper by E. A. Edmonds introduced an adaptive software development process.[3] Concurrently and independently, the same methods were developed and deployed by the New York Telephone Company's Systems Development Center under the direction of Dan Gielan. Also in the early 1970s, the concepts of Evolutionary Project Management (EPM), which has evolved into Competitive Engineering, got their start. These were followed with the so-called lightweight agile software development methods, which evolved in the mid-1990s as the carminative reaction against the waterfall-oriented methods, which were characterized by their critics as being heavily regulated, regimented, and micromanaged, and having overly incremental approaches to development. Proponents of these newer, lightweight agile methods contend that they are returning to development practices, which were present early in the history of software development.[4] Compared to traditional software engineering, agile development is mainly targeted at complex systems and projects with dynamic, "undeterministic", and nonlinear characteristics, in which accurate estimates, stable plans, and predictions are often hard to get in early stages, and big upfront designs and arrangements will probably cause a lot of waste, that is, not economically sound. These basic arguments and precious industry experiences learned from years of successes and failures have helped shape agile's flavor of adaptive, iterative, and evolutionary development.[5]

Early implementations of agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programing (1996), Adaptive Software Development, Feature Driven Development (1997), and Dynamic Systems Development Method (DSDM) (1995). After the Agile Manifesto[6] was published in 2001,[7] these have since been referred to collectively as "agile methodologies."

Although Agile is now being applied and discussed around software development, the core of Agile is also about the ability to structure organizations in such a way that they can embrace change and adapt quickly to service the customers in their ever-changing needs. However, taking a big-bang approach to Agile is not really a viable option for many organizations, as most successful adoptions of Agile are tailored to the strengths and limitations of the specific organization.

Like any other change, Agile adoption is not always welcomed right away and faces resistance. Organizations observe many types of frictions that reduce the momentum during Agile implementation. These frictions absorb energy because of the resistance at various levels. Friction is not a fundamental force but occurs because of the turbulence caused by the change. Three main types of frictions apply to the strategy linkage, organization, processes, and technical agility. In this way, Agile is referred to as a mindset, change, flexibility, nonfunctional requirement (link to strategy and goals), culture, and the ways of working, approach, or philosophy. This section discusses the basic definition of agility and introduces the agile features—the characteristics, values, principles, and practices of which Agile is composed.

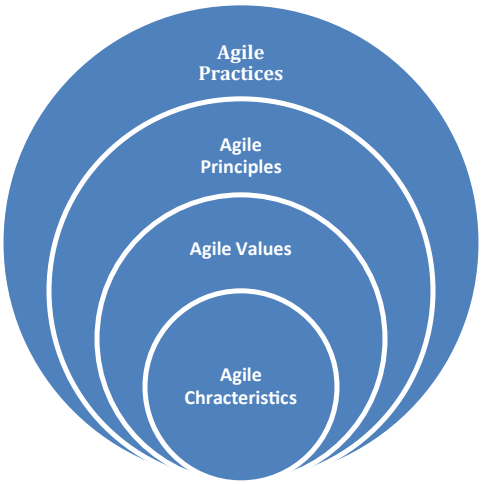

FIGURE 1

What is agile?

Figure 1 shows the conceptual relationship between agile features. At the core, and by far the most critical to the nature of Agile, are its characteristics; slightly less important are the values that are employed when Agile is practiced. This is followed by the agile principles that guide how Agile is applied, and then finally are the agile practices that form the basis for work within an agile setting.

Qumer and Henderson-Sellers (2008) provide the following precise definitions of agility and agile methods.

"Agility is a persistent behavior or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment, and applies updated prior knowledge and experience to learn from the internal and external environment."[8]

"A software development method is said to be an agile software development method when a method is people focused, communications oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening time frame and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)."[9]

AGILE CHARACTERISTICS

The agility definition highlighted the five fundamental agile characteristics: responsiveness, flexibility, speed, leanness, and learning. These five characteristics can be used to describe and measure the agility of an object or entity.

- *Responsiveness*: is the ability of an object or entity to scan and sense the external and internal opportunities; and form an appropriate response according to the situation at hand.
- *Flexibility*: is the ability of an object or entity to accommodate expected or unexpected changes.
- *Speed*: is the ability of an object or entity to provide a speedy or quick response to expected or unexpected changes.
- *Leanness*: is the ability of an object or entity to provide a speedy and flexible response with optimal or minimal resources without compromising the quality.
- *Learning*: is the ability of an object or entity to learn through continuously managing and applying up-to-date knowledge and experience.[10]

AGILE VALUES

Similarly, the six agile values provide fundamental statements that describe agile preferences:

1. Individual and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan
5. Keeping the process agile
6. Keeping the process cost-effective

The agile values one to four were provided by the Agile Manifesto (2001). The fifth agile value "keeping the process agile" was provided by Koch in 2005.[11] The sixth value of "keeping the process cost-effective" was provided by Qumer and Henderson-Sellers.[12]

AGILE PRINCIPLES

Agile Software development is based on 12 guiding principles, which are set out in the Agile Manifesto[13]:

1. Our highest priority is to satisfy the customer through early and continuous deliver of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As the 12 guiding principles make clear, they are software centric, to apply in the BMP context. We will show later how they can be tailored to apply in a different setting with great effect.

AGILE PRACTICES

A number of agile methods exist (e.g., XP, Scrum, and Lean). These methods provide concrete agile practices that adhere to the agile characteristics, values, and principles. The scope of each of the methods is slightly different from the others. For instance, XP focuses on employing technical software development practices such as Refactoring, "Pair Programming", Automated Testing, Continuous Integration and so on. Scrum focuses on project management practices and the use of "Sprints" to deliver functionality. Generally, agile development is supported by a bundle of concrete practices covering areas that may include the full range of product development from requirements, design, modeling, coding, testing, project management, process, quality, and so on. The result is that we learn two things: first, the differences indicate that no standard single agile method is available, which may be applied or adopted off-the-shelf; and second, the best practices from different agile methods can conceivably be combined to create a situation-specific agile method. What is important to note here is that the key to being agile is to focus on harnessing agile characteristics, values, and principles underlying the specific agile practices.

AGILE VERSUS TRADITIONAL WAYS OF WORKING

Agile and traditional waterfall methods are two distinct ways of developing software. The Waterfall model can essentially be described as a linear model of product delivery. Like its name suggests, waterfall employs a sequential set of processes as subsequently indicated in Figure 2. Development flows sequentially from a start point to the conclusion, the delivery of a working product, with several different stages along the way, typically: requirements, high-level design, detailed implementation,

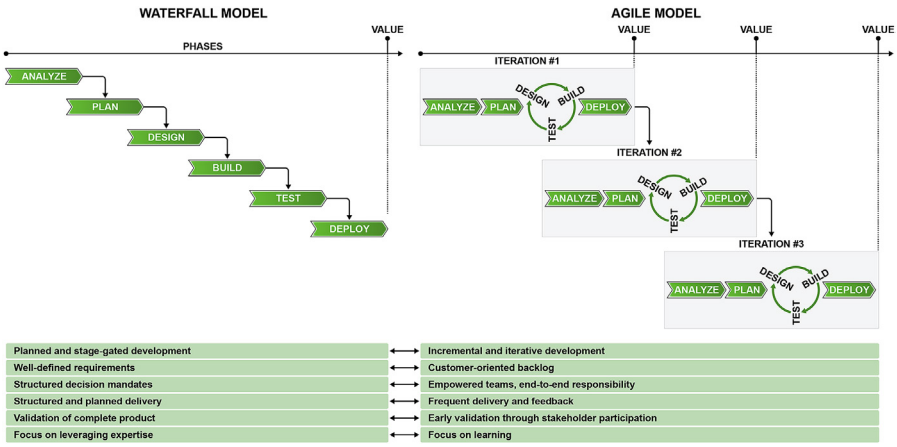

FIGURE 2

Agile versus traditional waterfall.

verification, deployment, and customer validation, often followed with stages to cover the running/maintenance of the product, and to address the need for continuous improvement.

The emphasis of Waterfall is on the project plan and managing all work against the plan. For this reason, a clear plan and a clear vision should exist before beginning any kind of development. Because the Waterfall method requires upfront, extensive planning, it permits the launch of a known feature set, for an understood cost and timeline, which tends to please clients.

Furthermore, Waterfall development processes tend to be more secure because they are so plan oriented. For example, if a designer drops out of the project, it isn't a huge problem, as the Waterfall method requires extensive planning and documentation. A new designer can easily take the old designer's place, seamlessly following the development plan. As described above, Agile offers an incredibly flexible design model, promoting adaptive planning and evolutionary development. Agile might be described as freeform software design. Workers only work on small packages or modules at a time. Customer feedback occurs simultaneously with development, as does software testing and deployment. This has a number of advantages, especially in project environments in which development needs to be able to respond to changes in requirements rapidly and effectively.

By way of comparison, instead of a big-bang waterfall product delivery, Agile focuses on delivering early value or product features in small increments, which is referred to as a minimum viable product or as having minimum marketable features. An agile project is organized into small releases, in which each release has multiple iterations. Within each iteration just enough work is pulled off the stack, planned, analyzed, designed, developed, tested, integrated, and then deployed in the production or a production-like staging environment. During and following

the iteration the product is demonstrated to concerned stakeholders for feedback and commitments. Each iteration also involves retrospective activity, which is aimed at identifying and addressing the issues of the agile practices. In each iteration, different developers may work on different modules or requirements (also known as user stories) throughout the development process and then work to integrate all of these modules together into a cohesive piece of working-software release. In summary, this can be seen as a process, which consists of analysis and planning stages, followed by a rapid design, build, and test cycle, all of which then ends with deployment.

Experience with the agile approach has shown that it can be especially beneficial in situations in which it is not possible to define and detail the project requirements, plan, and design upfront. Agile is also an excellent option for experimental circumstances. For example, if you are working with a client whose needs and goals are a bit hazy, it is probably worthwhile to employ the agile method. The client's requirements will likely gradually clarify as the project progresses, and development can easily be adapted to meet these new, evolving requirements. Agile also facilitates interaction and communication—collaboration is more important here than doing design in isolation. Because interaction among different designers and stakeholders is key, it is especially conducive to teamwork-oriented environments.

Figure 2 compares and contrasts key elements of Agile and Waterfall Development. In this figure, we see graphically the life cycle of each development model. Below each type of life cycle are listed the key properties of each method and how they relate to the equivalent properties of the alternative method.

AGILE BPM

Although Agile is not a silver bullet that can be applied to all problems, however, it does provide ways of working that could be suitable to the circumstances in which frequently changing business and customer requirements or other conditions of uncertainty force the organization to pursue quick wins for developing capabilities, services, or systems. As Agile is about making complex things simple or simpler, this section of the chapter will highlight how the agile concepts can be applied to enable BPM in all the various areas and disciplines as defined in by Qumer and Henderson-Sellers.[8] We must, however, keep in mind that agility of process is not in and of itself Agile BPM and that to incorporate agility into BPM actually requires a fundamental shift in the strategy, operations, and tactics of the way BPM works and how modeling is carried out in an organization. This section tackles this up-to-date subject in the context of:

1. The benefits and limitations of Agile and how to apply it to BPM
2. An Agile BPM method
3. A firmly defined terminology
4. A concept to develop agile capabilities in the BPM Center of Excellence (CoE).

The Benefits and Limitations of Agile and How to Apply It to BPM

We have seen that Agile offers several benefits (e.g., value to customer, organization, staff, and community) over traditional ways of working. We have seen, for example, that Agile focuses on developing a minimally marketable or viable product or service features, which will provide value to customers and community. In contrast to the traditional waterfall approach, it focuses on delivering value early to customers and community in short increments, which range in duration from anywhere between a few weeks to months. This seems helpful for the organizations and staff seeking to improve time to market and quality while reducing the cost of production and failure. Clearly then, agile ways of working not only help delivering value early, but they also seem appropriate in recognizing the risks and failure early to mitigate their impact.

A part of exploring the potential around Agile BPM also includes understanding the traditional problems and challenges when adapting a new concept. As with so many things, resistance or friction impede adaptation of a new way of thinking and working with agile concepts. For an organization adopting Agile BPM concepts, numerous challenges are possible. However, the most common challenges we have encountered are as follows:

- *Static Friction:* The force that must be overcome before agile concepts can be implemented in a nonagile organization, for example, friction observed before piloting first Agile BPM project.
- *Dynamic Friction:* The force that must be overcome to maintain uniform agile motion and the friction encountered when people don't see immediate results after a new Agile BPM project. It is important for the Agile BPM leader to constantly communicate value of "inspect and adapt." Once the BPM CoE and the organization learn to manage incremental value driven by agile process, dynamic friction starts diminishing by itself.
- *Political Friction:* The force resisting agile progress because of politics that can come from the BPM CoE or the organization itself. A good Agile BPM leader can influence negative politics by persuasive communication in Agile's favor.
- *Knowledge Friction:* The force that must be overcome due to the BPM CoE and the organizational lack of competencies and resources who understand Agile and its precepts, workings, and value. Most organizations use external consultants or hire an Agile BPM specialist to train, coach, and mentor employees so that they gain an agile knowledge base.

Once a solid agile knowledge base is in place in the BPM CoE, this friction will generally start to diminish.

These frictions limit the ability of an organization to maximize the use of Agile BPM in an optimized Way of Working, Modeling, and Governing. Table 1: Indicates the typical friction factors across organizational areas.

Friction is not the only challenge organizations will face when moving from traditional BPM to an Agile BPM way of thinking. Other influencing factors cause Agile BPM to fail or to deliver significantly lower value than expected:

- *Innovation is only done from the process perspective*: As it is in the IT world, the current view of Agile is very much defined by the Software/Application

Table 1 ***Example of Friction Factors Across Organizational Areas***

	Static	Political	Dynamic	Knowledge
Organizational	We are unique	Agile BPM versus non agile BPM	Not yet getting value for agile	Waterfall versus agile
Process	Why change?	Change control vs embrace change	Agile process is too fluid	Fear of the unknown, only know BPM waterfall methods
Technical	Where to start?	BPM CoE and process architecture committee vs community of practice	Not enough resources for process and software automation projects	Resources lack agile competencies.

and underlying technology perspective; this gives rise to a degree of vagueness of requirements, especially within the business layer. We also see this in the process community, in which they limit business innovation and transformation to what they can see the process can do rather than working in its context. Often this is based on a traditional BPM focus around optimizing the existing processes. However, this view limits Agile BPM concepts from enabling true business agility. The reason is that the current IT Agile methods only have feedback loops between the Plan and Deploy phases for a BPM project, placing an emphasis on the feedback from what is possible in the process and creating a disjointedness loop back to the business. Resulting in Agile BPM teams not having gone through the multiple agile business iterations capturing the value and performance aspects relevant in the Analyze and Plan phase (the Business Layer Context). Therefore, Agile BPM needs a better business requirement loop, which is elaborated on later in this chapter.

- *Multiple changes at once:* Changing both value and performance expectations as well as changing business requires the organization to introduce far too many changes during the iteration's Design, Build, and Test phases. This makes it very difficult for the Agile BPM teams to complete the process analysis, process alignment, process changes, process design, process automation, and so on, in the required 2-4 week sprint cycle. Agile BPM, therefor,e needs to build a better requirement and execution approach into the overall approach.
- *Users are not sure of what they can get versus what they want*: Business and process users are not always sure of what they can get from BPM initiatives. As Steve Jobs said, "It's really hard to design products by focus groups. A lot of times people don't know what they want until you show it to them."[14]
- "Having a developer who has a deep understanding of the business is often better than an inexperienced business person with no understanding of how

technology can enable work about "requirements." This also touches upon the challenge we previously discussed in which many process experts limit the business innovation and transformation aspects to what they can see that the process can do. The result is that value derived from the process as well as the execution of the innovation cycle, therefore, for the most part, comes too short and does not deliver the desired result or value.

- *Limited executive sponsorship equals limited agility*: Agility requires sponsorship at the highest level. It requires dedicating top performers and empowering them to challenge the status quo. That is, not just automating the existing siloed approach; executives need to resolve innovation and transformation blockers rapidly and with a focus on the final goal.
- *Agile Focus is operational and not strategic*: Agile teams tend to focus on operational accomplishments and report it to tactical level, however, at the expense of strategic business objectives. In BPM, this is tragic, for missing the big picture can lead to long-term failure at the expense of apparent success in the short term.
- *Agile has too little governance*: Agile teams often lack sufficient checks and balances; if this occurs, they can cause lots of damage in a very short time. Agile BPM must interlink with BPM Governance (see chapter BPM Governance).
- *Giving up on quality*: Because of the high demands or urgency that is placed on process deployment, an Agile BPM team often falls back on the crutch of checking only for process pain points/process defects instead of maintaining a high level of quality of overall BPM changes. (see the chapters on BPM Change Management and BPM Governance)

Some of these failure points are indicated in the following Agile figure, which for many organizations represents the current agile method with its failures and problems.

In Figure 3 we see the Agile development method laid out to show the Analyze and Plan stages, followed by the design, build, and test cycle, all of which then ends with the deployment stage with the key criticisms, or weaknesses, mapped to the applicable points in the method.

An Agile BPM Method

To overcome the challenges of the agile method and enable an organization to adapt Agile, BPM must enable strategic alignment and provide the necessary link to performance and value expectations, requirement management, coordination with business impact and changes, better quality, and thereby value creation and realization. For this, we need to augment the traditional agile approach to incorporate a stronger requirement management and an agile feedback loop in the analysis phase. This loop should consider all layers of the enterprise, that is, business, application, and technology, thus allowing the use of these

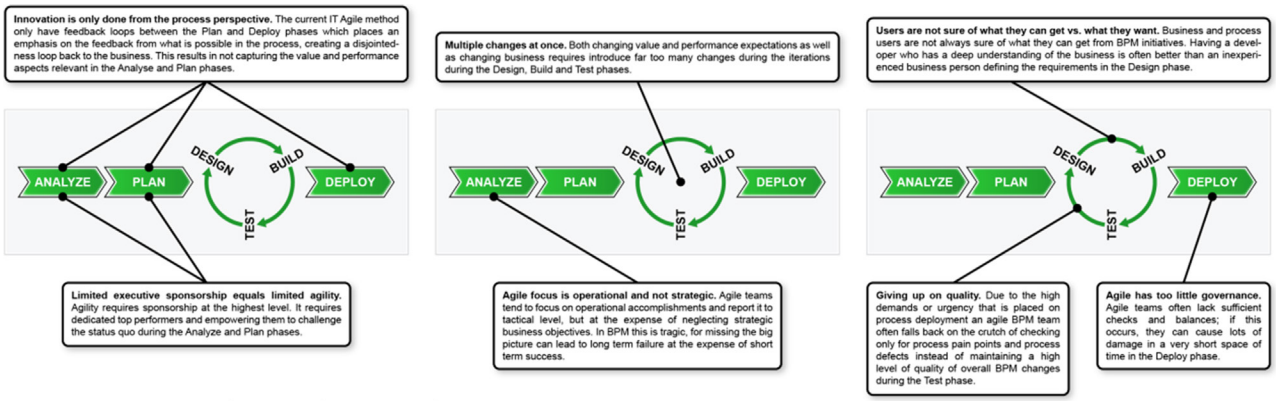

FIGURE 3

Agile weakness point indicators.[15]

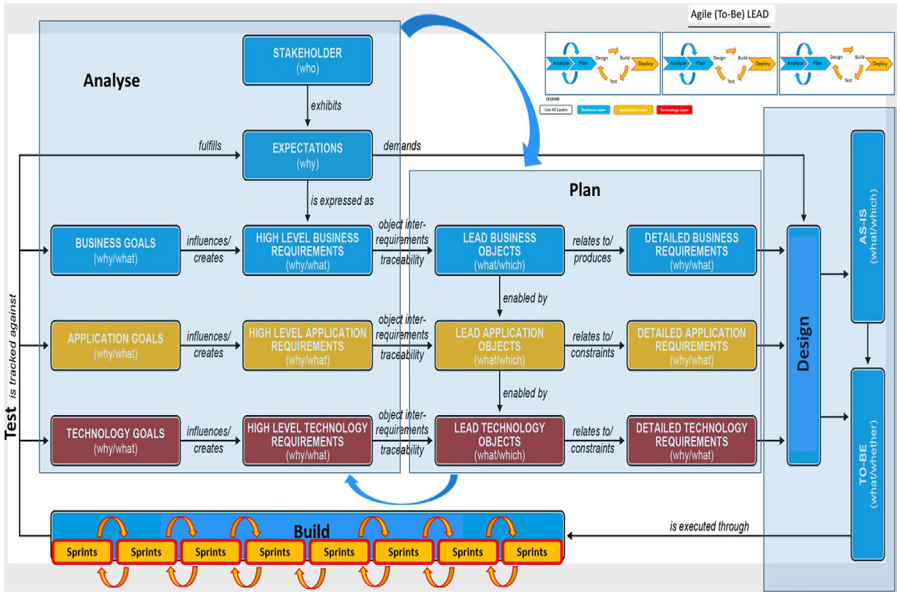

FIGURE 4

Details of agile BPM way of working.[16]

requirements in an Agile Way through the design Build and Test phase and to assess testing against the requirements prior to deployment. This is shown in Figure 4.

Agile Analysis

Agile analysis, in the context of Agile BPM, suggests active collaboration with the stakeholders to identify the requirements with necessary details at the release and iteration levels, instead of trying to get the complete detailed requirements up-front. The key difference (compared to traditional process analysis) is that the Agile BPM focuses on the relevant value and performance drivers and analyses in which and how they can be executed. A process in scope can be identified, modeled, analyzed, and decomposed into subprocesses for the Agile BPM project. Within each subprocess, a set of requirements is documented at high level in terms of process user stories in collaboration with the stakeholder at the beginning of the project. The identified user stories or requirements within each subprocess are estimated and prioritized. The prioritized process requirements are organized into short iterations and releases. A set of high-level process requirements or user stories for a given iteration can be further clarified, detailed, and confirmed (signed off) just before the start of the iteration (Zero Iteration). For instance, user stories planned for any given iteration can be detailed and signed-off beforehand. The detailed signed-off user stories can be made ready (just-in-time documentation) one iteration in advance before the start

Table 2 *Example of an Agile BPM Template Developed in the Analysis and Planning Phase*

Requirement#	Who/Whom Specification For Example, Stakeholder/ Owner	Where Specification For Example, Layer, Objects, Area (Process, Service, Data, Infrastructure, etc.)	What Specification: High-Level Requirements	What Specification: Detailed Requirements
#				
#				
#				
#				
#				
#				

of the process development of those user stories in the next iteration. It is important to note here that high-level user stories should only provide enough details that are necessary for estimation and prioritization, and should not lock in unnecessary low-level details, which may hinder the adaptability of the Agile BPM project.

Agile Planning

Traditional ways of BPM planning focus on the detailed up-front planning. Agile BPM ways of working require planning at project, release, iteration, and day level. Agile BPM focuses on initial high-level project plan that outlines number of project releases, resources, risks, and cost and benefits estimates. Out of the high-level and detailed requirements a project plan is developed to outline when and which requirements can be meet throughout as the project progresses in small releases. Table 2 illustrates an example of such a template/artifact, used to relate the captured components relevant in the plan phase aspects that is, Stakeholder/ Process Owner, relevant objects and the high-level (nonfunctional) requirements and detailed (functional) requirements. Such a template typically is in the form of a map, which can start as a simple row, and when information is added produces a catalog of rows. Because a release plan only focuses on the release in hand and the first two or three iterations for that release, such a template has the purpose of building an inventory or index list of the relevant stakeholders, objects, and requirements from the different relevant architectural layers that from the analysis phase can be used and tracked against in the planning phase.

Table 2 is the template of a map that captures requirements in a high-level and detailed form and indicates both who has an interest in the requirement and where within the layers and business objects each requirement resides. This enables the agile practitioner to use the release plan to track the project progress in small iterations. An iteration plan focuses on the iteration that will start next. It provides the detailed information about the time-boxed (2–4 weeks iteration) short-iteration activities and schedule such as additional analysis, the

design, the development in the build phase, and the test aspects, and so on, It also includes date and time of iteration, and shows cases and retrospectives. Daily planning in each iteration is achieved via daily stand-up meetings, in which team members discuss what they did yesterday, what will they do today? Are there any impediments? Agile project, release, iteration, and daily planning enable Agile BPM.

Agile Architecture and Design

Agile design for BPM can kick off by reviewing the existing As-Is process model and identified requirements for the target To-Be process model. Instead of a detailed up-front design, a high-level design for the To-Be process can be developed at the start of the project. This high-level design will then emerge with more details in short releases and iterations. Hence, a high-level design can be built on the identified requirements and objects relevant for the target To-Be situation. A high-level design will set the foundation for the Agile BPM project choices and options that enable the detailed design in each iteration (Design Phase-Product Backlog) wanted and specified by the stakeholders within their expectations. It is important to note here that, instead, a target final To-Be process can be achieved via small Transition states. Each project release and iteration should focus on developing a stable Transition state linked to the overall final To-Be state. Once defined, linked to relevant objects and approved To-Be requirements in the execution Build Phase fall under change control. In the Agile BPM way of thinking and working, this means additions to the product backlog will be made if additional requirements are identified in the ongoing Analyze, Plan, and Design Phases. Iteration Build and Testing enable tracking of build completion and quality against the requirements identified in the Analyze phase. This is crucial to ensure that both Backward and Forward Traceability of requirements have been achieved in the preceding phases so that Value generated through the Phases is not lost. This is highlighted in Figure 5 as follows.

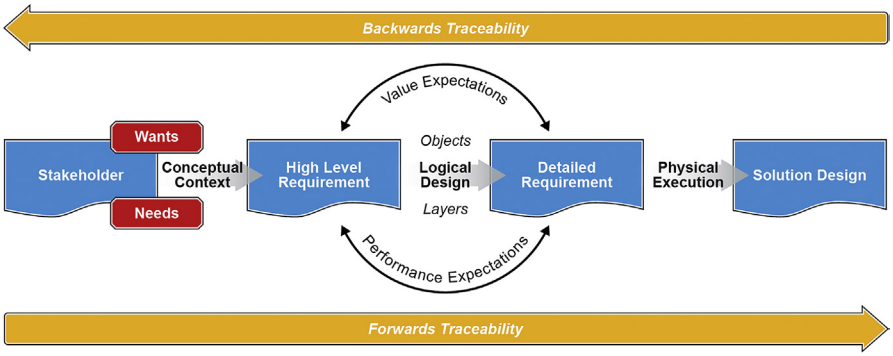

FIGURE 5

Agile BPM backward and forward traceability ensuring value generation.[17]

Agile Build

Traditional ways of working focus on big-bang product or service development in the build phase. Agile ways of working focus on building the product or service minimum marketable or viable features in small iterations based on the just-in-time user stories or requirements. Agile ways of development focus on delivering value early. The focus shifts from documentation to delivering working product or service features right from the beginning. So that Agile BPM initiatives can link strategy, identify value aspects, and focus on the relevant objects, the Agile BPM initiatives in the Build phase use process architecture concepts. There is a misunderstanding that the process architect role and process architecture artifact are not needed in the agile environment. However, various BPM project focused and isolated user stories may overlook the holistic picture of the enterprise architecture and underlying business process and technology assets. To be agile in the build phase, the process user stories need to be connected and classified as described in the Process Architecture chapter.

The impact of the process user stories needs to be considered through the lens of the relevant process architectural categorizations and classifications. Agile process architecture integrated with user stories would not only result in better identification of value, performance, rules, monitoring, organizational change management, risk, and implementation strategy, but it would also provide a shared vision of the enterprise to guide the agile teams working in the distributed Agile BPM development environments. Hence, agile ways of working require process architecture principles. This enables the project architecture, at the beginning of the project, to be linked to the holistic enterprise architecture, and then the details of the architecture will evolve as the BPM project progresses in small iterations. In the specific BPM work, we suggest applying Service-oriented Architecture (SOA) and Process-oriented Architecture (POA) principles. A repeatable process or a part of the process can be developed as a "service". A business process or workflow can be managed through the choreography and orchestration of services. As Qumer and Henderson Sellers[18] point out, the agile service-oriented process is developed in small iterations.

The agile build, test, and deploy phases use the To- Be design identified in the Design phase as the defined Product backlog to work with in the Build phase. The Agile BPM way of thinking and working in these phases uses the standard agile activities:

1. Defining the Product Backlog
2. Sprint Planning Meeting
3. Defining the Sprint Backlog
4. Sprint
5. Interrogating and Testing
6. Demo Release
7. Client Feedback Meeting
8. Retrospective
9. Refactoring
10. System Changes
11. System Testing

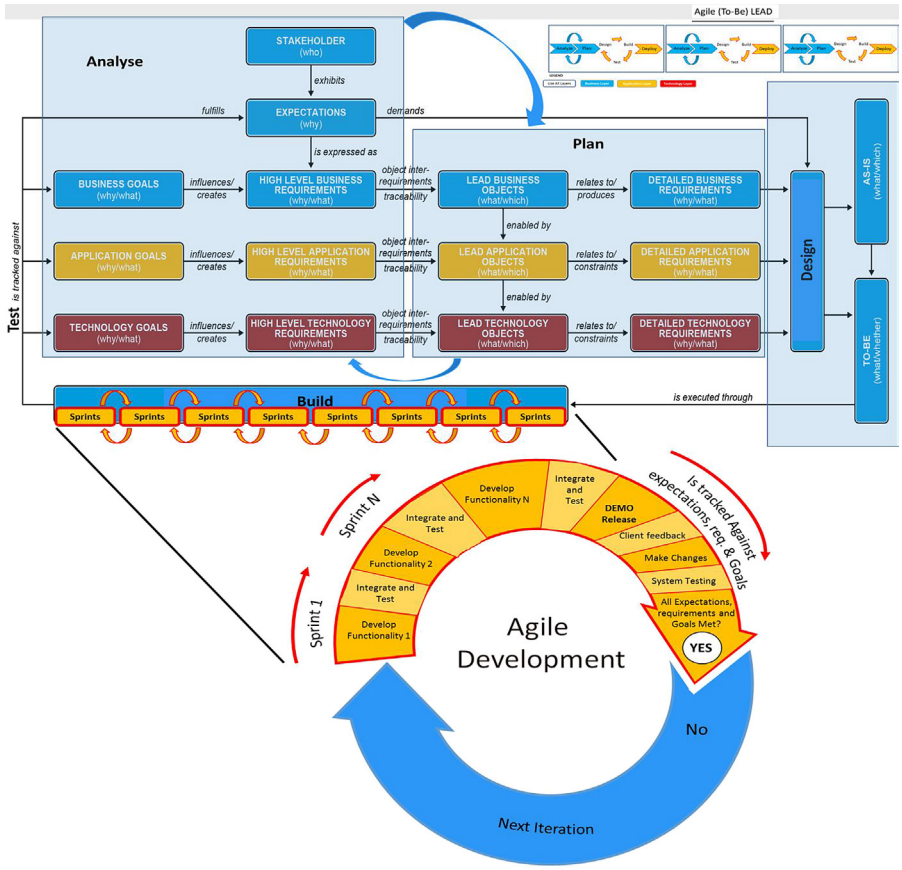

FIGURE 6

The Agile BPM build phase.[19]

12. Decision Point: Are expectations, requirements, and goals of Application and Technology Completed? If YES, then Deploy. If NO, then a new iteration is started.

The build process is presented in context in Figure 6, showing how each set of "build" sprints draws on the design to be executed.

Agile Testing

Although traditional ways of working around testing first do the testing once the whole product or service is developed, agile ways of working focus on testing the product or service minimum marketable or viable features in small iterations while the development is in progress. Agile ways of development focus on automating the testing practices, such as automated unit testing, acceptance testing, integration testing, and so on, Right from the beginning, the focus shifts from testing documentation to actually testing the

working product or service features against requirements. This is done not only to ensure that the working product or service meets the customer expectation but also to identify any risks or blockers. In each case, whether within the business, application, or technology layer, the way in which 'testing' the requirements is performed is to simulate and compare the As-Is and target behaviors so as to expose possible defects for gaps between that which is desired and what is to be provided. This is achievable as requirements and designs are now addressed through structured models, in which every design object can be traced to its requirement and the various goals it executes. Testing in terms of tracking may occur, not just at the application and technology layers, but within the business layer and within the work system that binds the project choices and options to the specific To-Be solution design that enables the innovation and transformation expressed, for example, demanded by stakeholder expectations. These can also be 'tested' against the specific goals that relate to the requirements as well as the full 'testing' of solution design into the "work system". This in turn leads to the ability to pragmatically consider design options to fulfill and thereby meet expectations and to verify the quality of the product prior to Deployment.

It is important and now possible to resolve any test-related issues during the relevant iteration. If an issue is not resolved in a given time-boxed iteration, then do not extend the duration of the iteration, rather move the issue to the next iteration, and record and prioritize it on the product or test backlog. A user story related to a product or service feature is considered done when it has passed all the acceptance tests. If a minor issue arises, then it is fine to let the user story pass and fix the issue in the next iteration.

Agile Deployment

The process models, end-to-end flows, and or process changes can be deployed into production either after each iteration or at the end of release. An individual product or service release deployment can be combined with other releases for different products. Organizations may have their own local release cycle. Agile BPM ways of deployment requires tracking the testing and changes. For these reasons, Agile BPM in the deployment phase focuses on collaborative and communication-oriented shared responsibility, accountability, and business value-oriented change and governance. As discussed earlier, as within Agile BPM, the requirements for the product or service features are the responsibility and accountability of the owning stakeholder. Senior management is responsible and accountable for funding, empowering, and supporting the managers. For instance, traditional BPM project governance uses a gated approach to release and monitor the fixed up-front project funding and outcomes. Agile BPM, however, decomposes the project into short releases. Project funding is released based on each successful release of a project. Therefore, agile managers and agile teams are mainly responsible and accountable for the delivery of a valuable quality product or service features to the customer. Therefore, in reality, the empowered agile managers, the Agile BPM team, and customer collaborate for the value co-creation. It is important to note that Agile BPM consequently requires empowering the BPM CoE managers and the downstream business managers.

Agile Terminology

Even though agile principles can be applied to Enterprise Architecture, Agile is not an Enterprise Architecture discipline, and hence no direct Objects or Meta Objects apply. However, as shown in the above text, Agile does bring a set of new concepts that are critical to being able to comprehend any discussions on what Agile is, how it works, and how it can be applied within BPM. Having standardized terms provides a structural way of thinking and enables having common terminology in the execution of Agile BPM. It enables the organization of terms around the viewpoints associated with Agile BPM (see chapter What BPM can learn from Enterprise Architecture).

As such, terminology is used with various existing delivery frameworks, methods, and approaches that exist within the BPM CoE and the Project Management Offices (PMO); it is vital when developing such a set of standardized terminology that it be 100% vendor neutral and agnostic from various vendor solutions.

Although the terms are based on a collection of best and leading practices around how to work with Agile BPM within an organization, we do not claim that these terms are all-comprehensive, but rather want to use the terminology that is most common in agile circles and apply to the Agile BPM work (Table 3).

Table 3 ***The Most Common Terms that Would Be Used Within Agile BPM***[20]

Term	Definition
Agile coach	A Person responsible for supporting and improving the capability of an organization to deliver in an agile way.
Agile driver and forces (external/internal)	Pressures that arise from outside or inside a system triggering agile approaches.
Backlog	A Prioritized list of requirements that are waiting to be worked on.
Bug	An error, flaw, mistake, failure, or fault in the process models, process rules, or process design that produces an incorrect or unexpected result, or causes it to behave in unintended ways.
Burndown chart	A Visual representation that shows work remaining over time.
Burnup chart	A Visual representation that shows work completed over time.
Business capability	An abstraction that represents the abilities and the quality of being capable, intellectually (logical) and or physically. Agile enterprise developments must be able to specify the aptitude that may be developed for the enterprise and how it will perform a particular function, process, or service.
Business change	Changes in the way an organization functions brought about through a project or other initiative.
Business resource/actor	A Specific person, system or organization internal or external that is part of or afected by the agile development to the enterprise. This can include that the agile development will influence or impact the resource/actor-defined functions and activities.

Table 3 ***The Most Common Terms that Would Be Used Within Agile BPM —Cont'd***

Term	Definition
Business service	Agile concepts applied to business concepts will impact the change and development of business services. In terms of the externally visible ("logical") deed, or effort performed to satisfy a need or to fulfil a demand, meaningful to the environment.
Business workflow	A Business workflow involved in the agile development, impacting and/or changing the stream, sequence, course, succession, series, progression, as well as order for the movement of information or material from one business function, business service, business activity (worksite) to another.
Continuous integration	When individual process models are combined in, for example, an entire end-to-end process flow and tested as soon as they are produced.
Cross-functional team	A Group of people with different skills and expertise working toward a common goal.
Defect trend	A Report that shows a rolling average of the number of problems (bugs) the team has opened, resolved, and closed.
Definition of "done"	An increment of a product that is ready for continual use by the end user. Can also be referred to as "done, done."
Deployment	All of the activities that make the process models ready for use and implementation.
Elaborate	When the delivery team adds detail to high-level business requirements.
Function	These are sometimes called epic stories or epics. Functions represent large sets of functionality, for example, Accounts receivable, Accounts payable, month-end close, etc.
Functionality	The behaviors that are specified to achieve.
Information radiator	A large, highly visible display that gives a picture of progress and key issues relating to an area of work.
Iteration	A Short time period in which a team is focused on delivering an increment of a product that is useable.
Kanban board	A visual board in which columns represent a state in which a user story can reside, for example, planned, blueprinting, realization, testing, done. Stories are arranged on the Kanban board and moved from one column to another as progress is being made. Many teams build physical Kanban boards by using tape and Post-it notes. Digital Kanban boards are another alternative.
Lean	Techniques to streamline processes and eliminate any activities that do not add value to the user.
Non-functional requirements	Describe how the process models or BPM projects should operate, as opposed to functional requirements that describe how it should behave. Typical examples would be: wished behavior, process security, accessibility, usability, availability, response times, etc.

Continued

Table 3 ***The Most Common Terms that Would Be Used Within Agile BPM —Cont'd***

Term	Definition
Owner	The person who is ultimately responsible for prioritization and acceptance of delivered features on a given process or project.
Performance expectations	Although for the most tagged and classified as non-functional requirements, the performance expectations are more as they specify the desire for the manner in which, or the efficiency with which, something reacts or fulfils its intended purpose as anticipated by a specific stakeholder. It will give an input to non-functional requirements, however the performance expectations will also be used in the early validation and thereby be the baseline against performance testing.
Release	Each release is associated with some type of go-live in which a number of processes or BPM projects are moved to roll out/production. For example, a "big bang" BPM program could have just a single large release. A more phased approach could lead to many releases within a single program.
Release plans	A Plan that sets out the order in which user requirements will be released into live service.
Retrospective	A Retrospective is a focused session in which your team looks back at how the current agile approach is working and which areas can be improved. Many agile teams conduct retrospectives at set intervals of time (every 8 weeks or at the end of every sprint).
Rework	Components of a project that will need to be revisited to correct bugs or altered to meet new requirements.
Show and tell	When the delivery team demonstrates how the product or service works at the end of each iteration to elicit feedback.
Sprint	A Sprint is typically a predetermined period of time (2 weeks, 4 weeks, 6 weeks, etc.) within which a set of identified user stories needs to be complete. Alternatives to sprints are to use a Kanban variation of agile project management methodology.
Stand-up	A Short meeting conducted standing up to report progress, share impediments, and make commitments.
Task	We generally try to avoid tracking detailed tasks, but sometimes we need to breakdown a single user story into multiple tasks and assign those to different people. For example, this can be handy for tracking specific process design and development tasks. Each task belongs to one and only one user story. It is the lowest-level entity that we track.
Technical debt	Poor process design in overall process architecture. The consequence of this is that more time is needed later on in the project to resolve process issues
Testing	A set of actions undertaken to assess whether a process or process model behaves as expected.

Table 3 ***The Most Common Terms that Would Be Used Within Agile BPM —Cont'd***

Term	Definition
Test coverage	The proportion of a process model that has been assessed.
Time box	A fixed time frame, usually to undertake an intense increment of work
User story	Each user story describes a particular business requirement and is assigned to a single function. In many ways, a user story is the next level down in terms of detail after a function. The standard question approach is followed: "WHO is needed, WHAT we do, WHY we do it". User stories are business-centric, not technology-centric. They do not capture HOW something will be accomplished (that comes later).
Value expectation	Tagging and classifying value expectations as nonfunctional requirements is a part of specifying the anticipated benefits that are of worth, importance, and significance to a specific stakeholder. It will give an input to nonfunctional requirements, however the value expectations will also be used in the customer orientation feedback loop, relating back to the specific stakeholders value expectations.
Value proposition?	A key principle of Agile is its recognition that during a project the customers can change their minds about what they want and need (often-called requirements churn), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, Agile BPM concepts need to adopt an empirical approach, accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to respond to changing value and/or performance expectations and thereby emerging requirements and create specific value proposition to the new need/want.
Velocity	The rate at which a team completes work.

Building Agile Capabilities in the BPM CoE

The establishment of an agile capability in an existing BPM CoE is a challenging task. This is partly because the CoE and teams have a different way of thinking, working, modeling, and implementing. It would, therefore, be appropriate to gradually establish agile capability by introducing agile roles, practices, and tools. One way to do is to use the Agility Adoption and Improvement Model (AAIM). This model (Figure 7) was first developed and published in 2007[21] and then was updated in 2010 as "AAIM Version 2.0".[22] AAIM V2.0 has been developed based on the intensive research in agile adoption at a large scale. The AAIM can be used as a roadmap or guide for agile transformation. Organizations or teams can adopt and improve agile environment to achieve specific agile level(s).

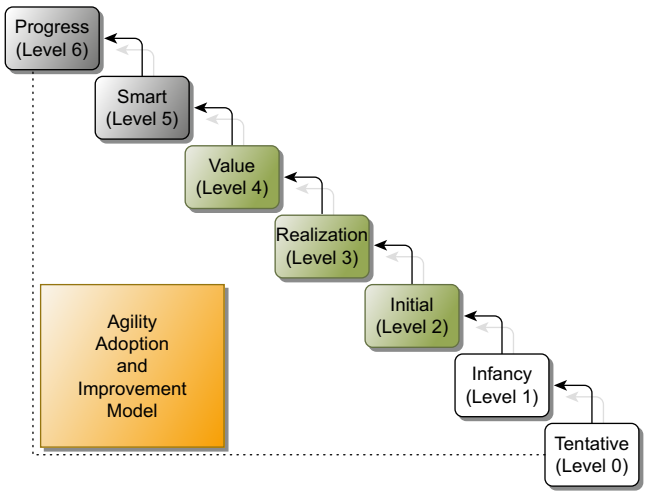

FIGURE 7

The agility adoption and improvement model V2.0.[23]

AGILITY ADOPTION AND IMPROVEMENT MODEL

The AAIM is structured into white, green, and black blocks and six levels (from 0 to level 6—Tentative to Progress). The colors indicate the levels, in which the white blocks are those levels at which initial experience in critical aspects of agility is garnered, whereas the green blocks are the levels at which the agile practices are established and entrenched, and the black blocks show those levels at which agile disciplines become universal. Each agile level has a name and specifies the lean agile principles to follow to achieve the particular level. Continuous improvement is integral to each level. The achieved lean agile level shows the lean agile maturity of an organization or team. The following section discusses the AAIM in the context of Agile BPM capability establishment.

- Tentative (Level 0)
  This level focuses on establishing an experimental environment whereby experience can be gained by BPM teams with some of the agile roles, practices, and tools. Based on this initial experience, BPM teams can communicate the perceived advantages of agile ways of working to the senior management to seek their support to begin with the further systematic establishment of the Agile BPM capability.
- Infancy (Level 1)
  This level focuses on adopting a basic elementary set of agile principles, roles, practices, and tools to support the iterative and incremental test-driven BPM development (evolutionary environment). This level provides the foundation for the further establishment of the Agile BPM capability.

- Initial (Level 2)
  This level focuses on establishing a collaborative BPM environment by adopting agile principles, roles, practices, and tools to support active communication and collaboration among the team members and internal and external shareholders.
- Realization (Level 3)
  This level focuses on establishing a simple result-focused Agile BPM capability by adopting agile principles, roles, practices, and tools to support the production of the executable BPM artifacts with minimal or reduced documentation. This is an advanced agile level, and the teams who are not accustomed to working with less documentation would find it challenging. This level could only be achieved if a well-established communication-oriented culture exists in the organization (e.g., established at level 2).
- Value (Level 4)
  This level focuses on establishing self-organizing BPM teams by adopting agile principles, roles, practices, and tools. Critical here is the fact that self-organization requires working knowledge and experience of Agile.
- Smart (Level 5)
  This level focuses on establishing a knowledge-focused Agile BPM capability by adopting agile principles, roles, practices, and tools to support knowledge management and innovation beyond the scope of an individual Agile BPM project and team.
- Progress (Level 6)
  This level focuses on establishing a continuous improvement by sustaining and continuously improving Agile BPM capability.

Some lessons learned around the Agility Adoption and Improvement journey include the realization that no single agile method or approach is a silver bullet. An organization should not focus too much on mechanically adopting only one agile framework end to end, such as Scrum or XP, but rather should focus on establishing and harvesting an agile mind set, values, principles, thinking, practices, roles, tools, and culture by using some kind of road map or adoption model of progress. The essential lesson is to let agile teams Assess, Tailor, Adopt, and Improve their own agile method suitable to their needs, context, or project, and focus on more "Facilitating and Guiding" teams with appropriate "Reward and Incentive Program" in their agile transformation journey while avoiding the imposition of "Agile" on teams.

CONCLUSION

Traditional BPM ways of working focus on detailed up-front planning, requirements analysis, process analysis, process design, process implementation, and continuous improvement to adjust changes. In other words, traditional takes a waterfall approach to BPM. Here, the assumption is that all the requirements for the process work are fixed, known, or complete. A lot of time and resources are spent up front

for achieving this illusion of a fixed or apparently complete list of requirements and plans, without actually delivering a single feature of a working product or service. By the time requirements are completely defined, signed off, and developed, business focus and market competition may have already been changed in response to an always-changing business environment, or changing performance or value expectations. Organizations need to be agile in response to such changing business environments and expectations. We have, therefore, focused in this chapter on the question of why we need to be agile, when, and how Agile BPM could be applied, as well as how to establish an agile capability. Applying the Agile BPM way of thinking and working will ensure that the BPM CoE teams work in a faster way and apply Kaizen principles of continuous improvement directly in their way of working. In that, they learn from what they do and how, Agile adapts and reshapes their manner of delivering the project and involves stakeholders in a new way, for example, as coparticipants in the process.

End Notes

1. Larman Craig., Agile and Iterative Development: A Manager's Guide. Addison-Wesley, (2004) p. 27.
   Ambler Scott., Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. John Wiley & Sons, (12 April 2002) pp. 12.
   Boehm, B. R., Turner. Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Wesley, (2004).
   Sliger, M., Broderick, S., The Software Project Manager's Bridge to Agility. Addison-Wesley, (2008).
   Rakitin, S.R., "Manifesto Elicits Cynicism: Reader's letter to the editor by Steven R. Rakitin", IEEE (2001).
   Geoffrey Wiseman (July 18, 2007). "Do Agile Methods Require Documentation?" InfoQ.
   Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J., Agile Software Development Methods: Review and Analysis. VTT Publications, (2002) 478.
   Guide to Agile Practices, the Agile Alliance.
   Aydin, M.N., Harmsen, F., Slooten, K. V., & Stagwee, R. A., An Agile Information Systems Development Method in Use. Turk J Elec Engin, (2004) 12(2), 127–138.
2. Gerald M., Weinberg, As quoted in Larman, Craig; Basili, Victor R. (June 2003). "Iterative and Incremental Development: A Brief History". Computer 36(6): 47–56, doi:10.1109/MC.2003.1204375, ISSN 0018–9162.
3. Edmonds, E.A., "A Process for the Development of Software for Nontechnical Users as an Adaptive System". General Systems, (1974) 19: 215–218.
4. See note 559 above.
5. Larman, C., Agile and Iterative Development: A Manager's Guide. Addison-Wesley, (2004) p. 27. ISBN 978-0-13-111155-4.
6. Agile Manifesto., Manifesto for Agile Software Development, (2001) http://agilemanifesto.org/.
7. Ibid.

8. Qumer, A., & Henderson-Sellers, B.,"A framework to support the evaluation, adoption and improvement of agile methods in practice", Journal of Systems and Software, (2008) vol. 81, no. 11, pp. 1899–1919.
9. Ibid.
10. Gill, A.Q., "Towards the Development of an Adaptive Enterprise Service System Model", Americas Conference on Information Systems, Chicago, USA, August 2013 in Americas Conference on Information Systems (AMCIS 2013), Shim, J.P et al., ed., AIS, USA (2013).
11. Koch, A.S., "Agile Software Development: Evaluating the Methods for Your Organization", Artech House, Inc, London (2005), pp. 1–272.
12. Quner, A., Henderson-Sellers, B. and McBride, T., (2007). Agility Adoption and Improvement Model, EMCIS 2007.
13. Agile Manifesto (2001), Manifesto for Agile Software Development, http://agilemanifesto.org/.
14. Source: Business Week, (May 12 1998).
15. LEADing Practice Business Process Reference Content [#LEAD-ES20005BP].
16. LEADing Practice Agile Reference Content #LEAD-ES30006ES.
17. Ibid.
18. Qumer, A., & Henderson-Sellers, B., "ASOP: an agile service-oriented process", International Conference on Software Methods and Tools, Rome, Italy, November 2007 in New Trends in Software Methodologies, Tools and Techniques. Proceedings of the sixth SoMeT_07, H. Fujita and D. Pisanelli, ed., IOS Press, Amsterdam, The Netherlands, (2007) pp. 83–92.
19. See note 16 above.
20. Taken from the LEADing Practice Agile Reference Content LEAD-ES30006ES.
21. See note 18 above.
22. Qumer, A., A Framework to Assist in the Assessment and Tailoring of Agile Software Development Methods, PhD Thesis, (2010) UTS.
23. Gill, A.Q., Bunker, D., "SaaS Requirements Engineering for Agile Development" in Xiaofeng Wang, N. Ali, I. Ramos, R. Vidgen ed., Agile and Lean Service-Oriented Development: Foundations, Theory, and Practice, IGI, USA, (2013) pp. 64–93.