HOW TO COMBINE AGILE INTO THE BPM WORK

The term “Agile” has attracted signification attention across industry and academia[1]. There are a number of ways in which agile concepts can be applied to various enterprise disciplines and industry verticals such as agile software development, agile project management, agile supply chain, agile manufacturing, agile service management, agile process management[2] and the list goes on. These agile principles are mostly applied to various enterprise concepts from 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 to establish an agile BPM capability? This chapter provides precise and practical answers to these fundamental questions.

What Is Agile?

While the basic “agile” term comes from the Latin word *agilis* and means to drive, do, and see. The basic meaning of agile means to move quickly, lightly, and easily. In 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 its 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 it 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 as 1957[3] before falling out of favour for the heavyweight waterfall method. In more modern times the agile movement began to come back when, in 1974, a paper by E. A. Edmonds introduced an

[1] Larman, Craig (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley. p. 27.
Ambler, Scott (12 April 2002). Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process. John Wiley & Sons. pp. 12
Boehm, B.; R. Turner (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Wesley.
Sliger, Michele; Broderick, Stacia (2008). The Software Project Manager's Bridge to Agility. Addison-Wesley.
Rakitin, Steven R. (2001). "Manifesto Elicits Cynicism: Reader's letter to the editor by Steven R. Rakitin". IEEE
Geoffrey Wiseman (July 18, 2007). "Do Agile Methods Require Documentation?” InfoQ.
Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. VTT Publications 478
"Guide to Agile Practices", the Agile Alliance.
Aydin, M.N., Harmsen, F., Slooten, K. v., & Stagwee, R. A. (2004). An Agile Information Systems Development Method in use. Turk J Elec Engin, 12(2), 127-138

[2]

[3] 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.

adaptive software development process[4]. 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 (EVO), 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, micromanaged, and having overly incremental approaches to development. Proponents of these newer, lightweight agile methods contend that they are returning to development practices that were present early in the history of software development[5]. Compared to traditional software engineering, agile development is mainly targeted at complex systems and projects with dynamic, “undeterministic”, and non-linear characteristics, where accurate estimates, stable plans, and predictions are often hard to get in early stages, and big up-front designs and arrangements will probably cause a lot of waste, i.e. not economically sound. These basic arguments and precious industry experiences learned from years of successes and failures have helped shape agile's flavour of adaptive, iterative, and evolutionary development.[6]

Early implementations of agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development (1997), and Dynamic Systems Development Method (DSDM) (1995). After the Agile Manifesto[7] was published in 2001[8] these have since been referred to collectively as “agile methodologies”,

While 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 which 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. There are three main types of frictions which are applied to the strategy linkage, organization, processes, and the technical agility. In this way, agile is referred to as a mind-set, change, flexibility, non-functional requirement (link to strategy and

[4] Edmonds, E. A. (1974). "A Process for the Development of Software for Nontechnical Users as an Adaptive System". General Systems 19: 215–18.
[5] 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.
[6] Larman, Craig (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley. p. 27. ISBN 978-0-13-111155-4.
[7] Agile Manifesto 2001,’ Manifesto for Agile Software Development’, http://agilemanifesto.org/.
[8] Ibid

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 shows the conceptual relationship between the 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 which are employed when agile is practiced. This is followed by the agile principles which guide how agile is applied, and then finally are the agile practices which form the basis for work within an agile setting.

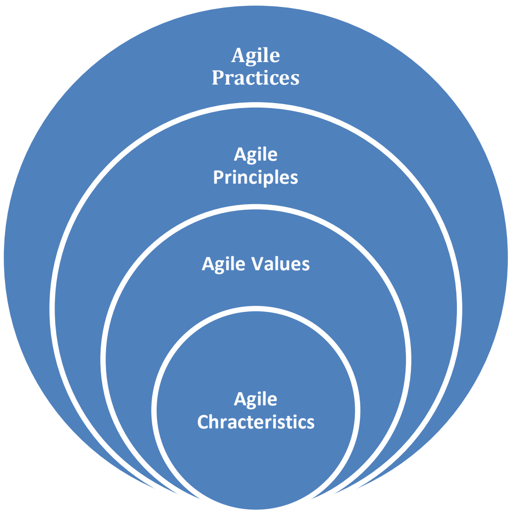


Figure 1: What is agile?

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

“Agility is a persistent behaviour 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." [9]

“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 timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)”.[10]

[9] Qumer, A. & Henderson-Sellers, B. 2008, 'A framework to support the evaluation, adoption and improvement of agile methods in practice', *Journal of Systems and Software*, vol. 81, no. 11, pp. 1899-1919.
[10] Ibid

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 in 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.[11]

Agile Values

Similarly, the six agile values provide fundamental statements that describe agile preferences. The agile values 1-4 were provided by the Agile Manifesto (2001). The fifth agile value "keeping the process agile" was provided by Koch in 2005.[12] The sixth value of "keeping the process cost effective" was provided by Qumer and Henderson-Sellers[13].

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

Agile Principles

Agile Software development is based on twelve guiding principles which are set out in the Agile Manifesto[14]:

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.

[11] Gill, A.Q. 2013, '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)*, ed Shim, J.P et al., AIS, USA.
[12] Koch, A.S. 2005, 'Agile Software Development: Evaluating the Methods for Your Organization', Artech House, Inc, London.
[13] Quner, A., Henderson-Sellers, B. and McBride, T. 2007, Agility Adoption and Improvement Model, EMCIS 2007.
[14] Agile Manifesto 2001,' Manifesto for Agile Software Development', http://agilemanifesto.org/.

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 behaviour accordingly

As the twelve guiding principles make clear, are 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

There are a number of agile methods (e.g. XP, Scrum, Lean). These methods provide concrete agile practices which adhere to the agile characteristics, values, and principles. While 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, and Continuous Integration etc. and 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 which may include the full range of product development from requirements, design, modelling, coding, testing, project management, process, quality, and so on. The result is that we learn two things, first the differences indicate that there is no such thing as a standard single agile method that may be applied or adopted off-the-shelf and second, that it may be argued that best practices from different agile methods can be combined to create a situation-specific agile method. What is important to note here that the key to being agile is to focus on harnessing agile characteristics, values, and principles underlying the specific agile practices.

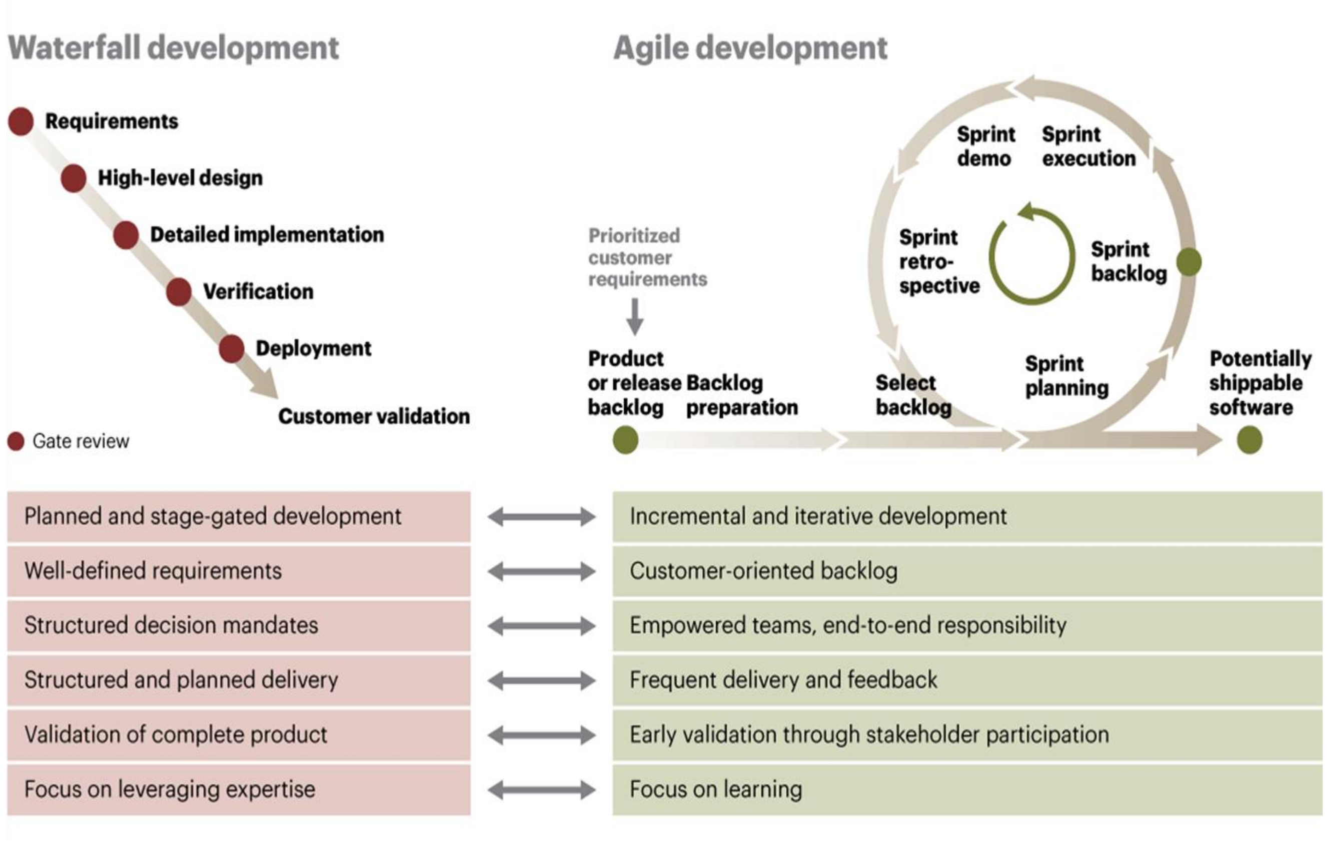
Agile vs Traditional Ways of Working

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 indicated in Figure 2 below. 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, 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, before beginning any kind of development there needs to be a clear plan and a clear vision. Because the Waterfall method requires upfront, extensive planning, this 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, following the development plan without a problem. 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 where 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 minimum marketable features. An agile project is organised into small releases, where each release has multiple iterations. Within each iteration just enough work is pulled off the stack, planned, analyszed, designed, developed, tested, integrated and then can be deployed in the production or production like environment such staging environment. During and at the end of 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 (a.k.a 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 that consists of Analysis and Planning stages, followed by a rapid Design, Build, Test cycle all of which then ends with deployment.



Planned and stage-gated development	Incremental and iterative development
Well-defined requirements	Customer-oriented backlog
Structured decision mandates	Empowered teams, end-to-end responsibility
Structured and planned delivery	Frequent delivery and feedback
Validation of complete product	Early validation through stakeholder participation
Focus on leveraging expertise	Focus on learning

Figure 2: Agile versus Traditional Waterfall

Experience with the agile approach has shown that it can be especially beneficial in situations where 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 lifecycle of each development model. Below each of the portrayal of each type of life cycle are the key properties of each method and how they relate to the equivalent property of the alternative method.

Agile BPM

While agile is not a silver bullet which can be applied to all problems, however it does provide ways of working that could be suitable to the circumstances where frequently changing business and customer requirements or other conditions of uncertainty force the organisation 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. We must however keep in mind that agility of a process is not in and of itself Agile BPM and that in order to incorporate agility into BPM actually requires a fundamental shift in the strategy, operations, and tactics of the way BPM works and how modelling is carried out in an organisation. 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 firm defined terminology
4. A concept to develop agile capabilities into 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, organisation, 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 that will have value to customers and community. In contrast to 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 organisations 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 recognising the risks and failure early so as 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 there are resistance or frictions in adapting a new way of thinking and working with agile concepts. For an organization adopting Agile BPM concepts, there are numerous possible challenges. 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 non-agile organization e.g. 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 favour.
- **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 organisations 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 knowledgebase is in place in the BPM CoE this friction will generally start to diminish.

These frictions limit the ability of an organisation to maximise the use of Agile BPM in an optimised Way of Working, Modelling, and Governing. Table 1: Indicates the typical friction factors across organisational areas.

	Static	**Political**	**Dynamic**	**Knowledge**
Organisational	We are unique	Agile BPM Vs non Agile BPM	Not getting Value for Agile yet	Waterfall Vs Agile
Process	Why to 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.

Table 1: Example of friction factors across organisational areas.

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

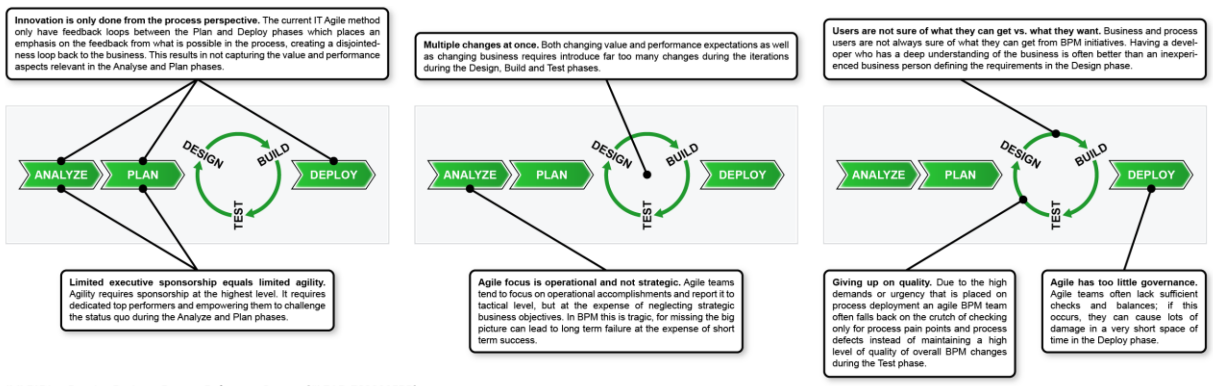
- **Innovation is only done from the process perspective.** As it is in the IT world, the current view of agile is very much defined by the Software/ Application and underlying technology perspective; this gives rise to a degree of vagueness of requirements, especially within the business layer. We also see this in the process community, where 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 that 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 placing for a BPM project an emphasis on the feedback from what is possible in the process, 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 Analyse 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, etc. in the required 2-4 weeks sprint cycle. Agile BPM therefore needs to build a better requirement and execution approach into the overall approach.

- **Users are not sure of what they can get vs 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.”[15]
- “Having a developer who has a deep understanding of the business is often better than an inexperience business person with no understanding of how technology can enable work about “requirements”. This also touches upon the challenge we previously discussed where 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, comes to 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. I.e. 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 space of time. Agile BPM must interlink with BPM Governance (see chapter BPM Governance)
- **Giving up on quality:** Due to 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 Agile figure below which for many organisations represents the current agile method with its failures and problems.

In figure 3 we see the Agile development method laid out to show the Analyse 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 point in the method.

[15] Source: Business Week, May 12 1998


Innovation is only done from the process perspective. The current IT Agile method only have feedback loops between the Plan and Deploy phases which places an emphasis on the feedback from what is possible in the process, creating a disjointed-ness loop back to the business. This results in not capturing the value and performance aspects relevant in the Analyse and Plan phases.

Multiple changes at once. Both changing value and performance expectations as well as changing business requires introduce far too many changes during the iterations during the Design, Build and Test phases.

Users are not sure of what they can get vs. what they want. Business and process users are not always sure of what they can get from BPM initiatives. Having a developer who has a deep understanding of the business is often better than an inexperienced business person defining the requirements in the Design phase.

Limited executive sponsorship equals limited agility. Agility requires sponsorship at the highest level. It requires dedicated top performers and empowering them to challenge the status quo during the Analyze and Plan phases.

Agile focus is operational and not strategic. Agile teams tend to focus on operational accomplishments and report it to tactical level, but at the expense of neglecting strategic business objectives. In BPM this is tragic, for missing the big picture can lead to long term failure at the expense of short term success.

Giving up on quality. Due to 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 and process defects instead of maintaining a high level of quality of overall BPM changes during the Test phase.

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 space of time in the Deploy phase.

(c)LEADing Practice Business Process Reference Content [#LEAD-ES20005BP]

Figure 3: Agile weakness point indicators.[16]

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, and 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 which considers all the layers of the enterprise i.e. business, application, and technology, thus allowing the use of these 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 below.

Agile Analysis

Agile analysis in the context of Agile BPM, suggests to actively collaborate 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 upfront. The key difference (compared to traditional process analysis) is that the Agile BPM focuses on the relevant value and performance drivers and analyses where and how they can be executed. A process in scope can be identified, modelled, analyzed and decomposed into sub-processes for the Agile BPM project. Within each sub-process, a set of requirements are documented at the 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 sub-process are estimated and prioritised. The prioritised process requirements are organised 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 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.

[16] LEADing Practice Business Process Reference Content [#LEAD-ES20005BP]

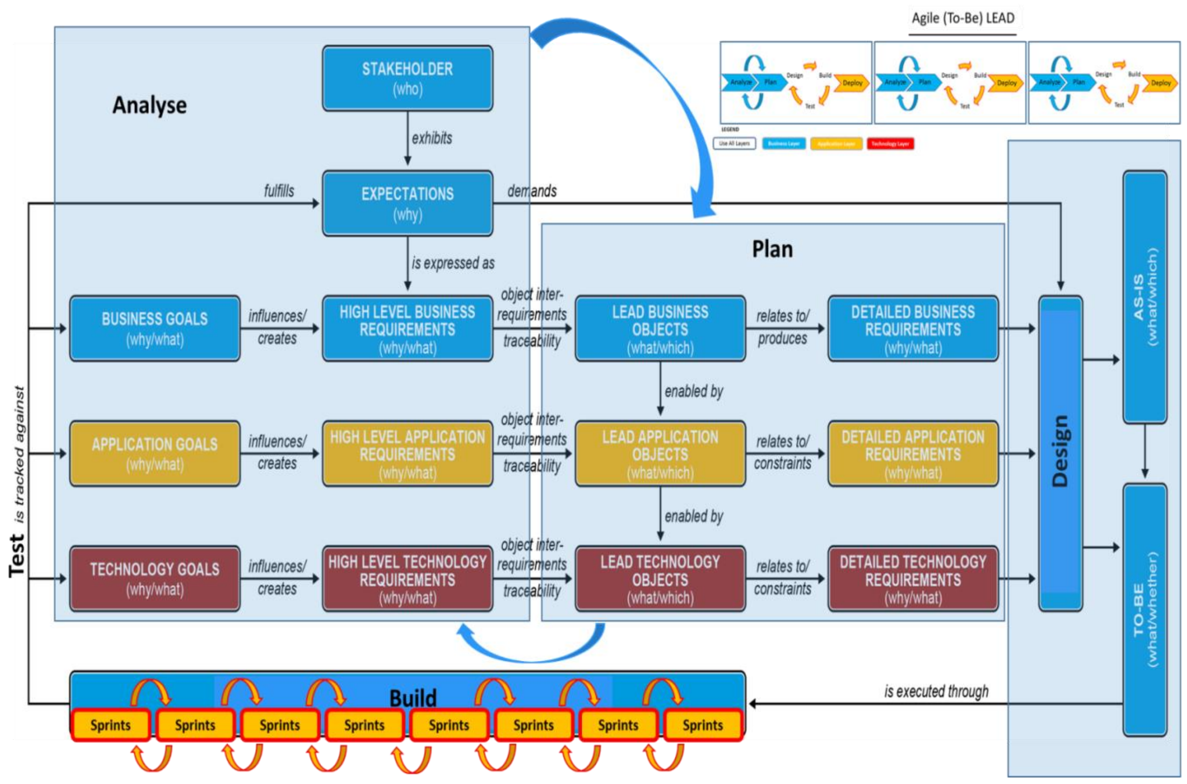

Agile (To-Be) LEAD
Analyse
STAKEHOLDER (who)
exhibits
fulfills
EXPECTATIONS (why)
demands
is expressed as
BUSINESS GOALS (why/what)
influences/ creates
HIGH LEVEL BUSINESS REQUIREMENTS (why/what)
object inter-requirements traceability
Plan
LEAD BUSINESS OBJECTS (what/which)
relates to/ produces
DETAILED BUSINESS REQUIREMENTS (why/what)
enabled by
APPLICATION GOALS (why/what)
HIGH LEVEL APPLICATION REQUIREMENTS (why/what)
LEAD APPLICATION OBJECTS (what/which)
relates to/ constraints
DETAILED APPLICATION REQUIREMENTS (why/what)
TECHNOLOGY GOALS (why/what)
HIGH LEVEL TECHNOLOGY REQUIREMENTS (why/what)
LEAD TECHNOLOGY OBJECTS (what/which)
DETAILED TECHNOLOGY REQUIREMENTS (why/what)
Design
AS-IS (what/which)
TO-BE (what/whether)
Test is tracked against
Build
Sprints
is executed through

Figure 4: Details of Agile BPM Way of Working.[17]

Agile Planning

Traditional ways of BPM planning focus on the detailed upfront 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, 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 the project progresses in small releases. Table 2 illustrates an example of such a template/artefact, used to relate the captured components relevant in the plan phase aspects i.e. Stakeholder/Process Owner, relevant objects and the high level (non-functional) requirements and detailed (functional) requirements. Such a template typically is in a form of a map, which can start as a simple row and when information is added produces a catalogue of rows. Since a release plan only focuses on the release in hand and the first 2 or 3 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

[17] LEADing Practice Agile Reference Content #LEAD-ES30006ES

phase can be used and tracked against in the planning phase.

Requirement #	Who/Whom specification e.g. Stakeholder/Owner	Where specification e.g. Layer, Objects, Area (process, service, data, infrastructure) etc	What specification: High Level Requirements	What specification: Detailed Requirements
#				
#				
#				
#				
#				
#				

Table 1: Example of an Agile BPM Template developed in the analysis and planning phase

Table 2 is the template of a map which captures requirements in a high level and detailed form and both indicates 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 iteration. 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, and the development in the build phase and the test aspects etc. It also includes date and time of iteration show cases and retrospectives. Daily planning in each iteration is achieved via daily stand-up meetings, where team members discuss what did they do 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 upfront design, a high-level design for the To-Be process can be developed in 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 details 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 Analyse, Plan and Design Phases. Iteration Build and Testing enable tracking of build completion and quality against the requirements identified in the Analyse 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 below.

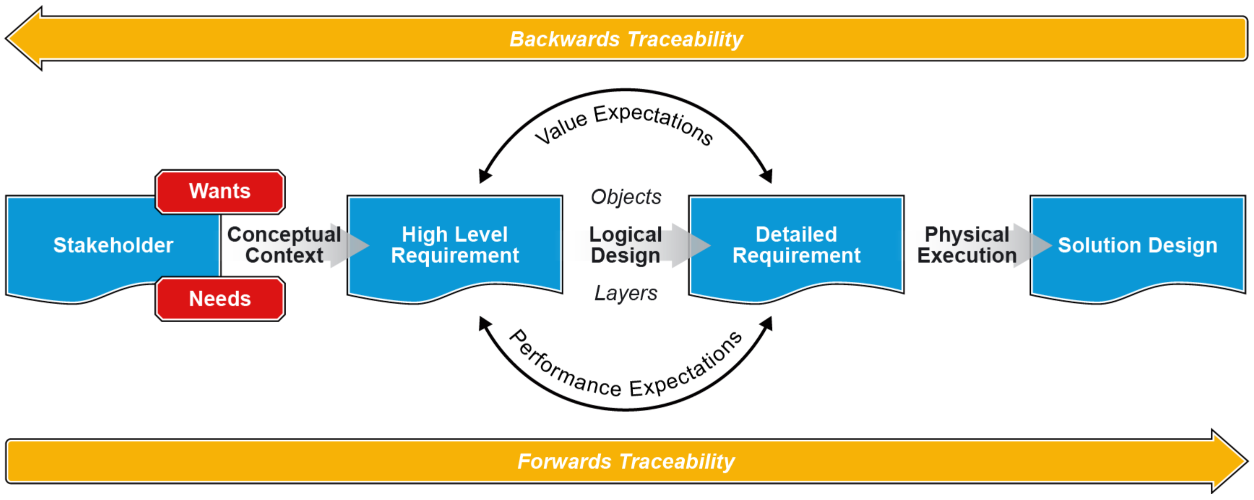


Figure 5: Agile BPM Backward and Forward Traceability ensuring value generation.[18]

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. In order 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 there is no need of the process architect role and process architecture artefact 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 looked into 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

[18] LEADing Practice Agile Reference Content #LEAD-ES30006ES

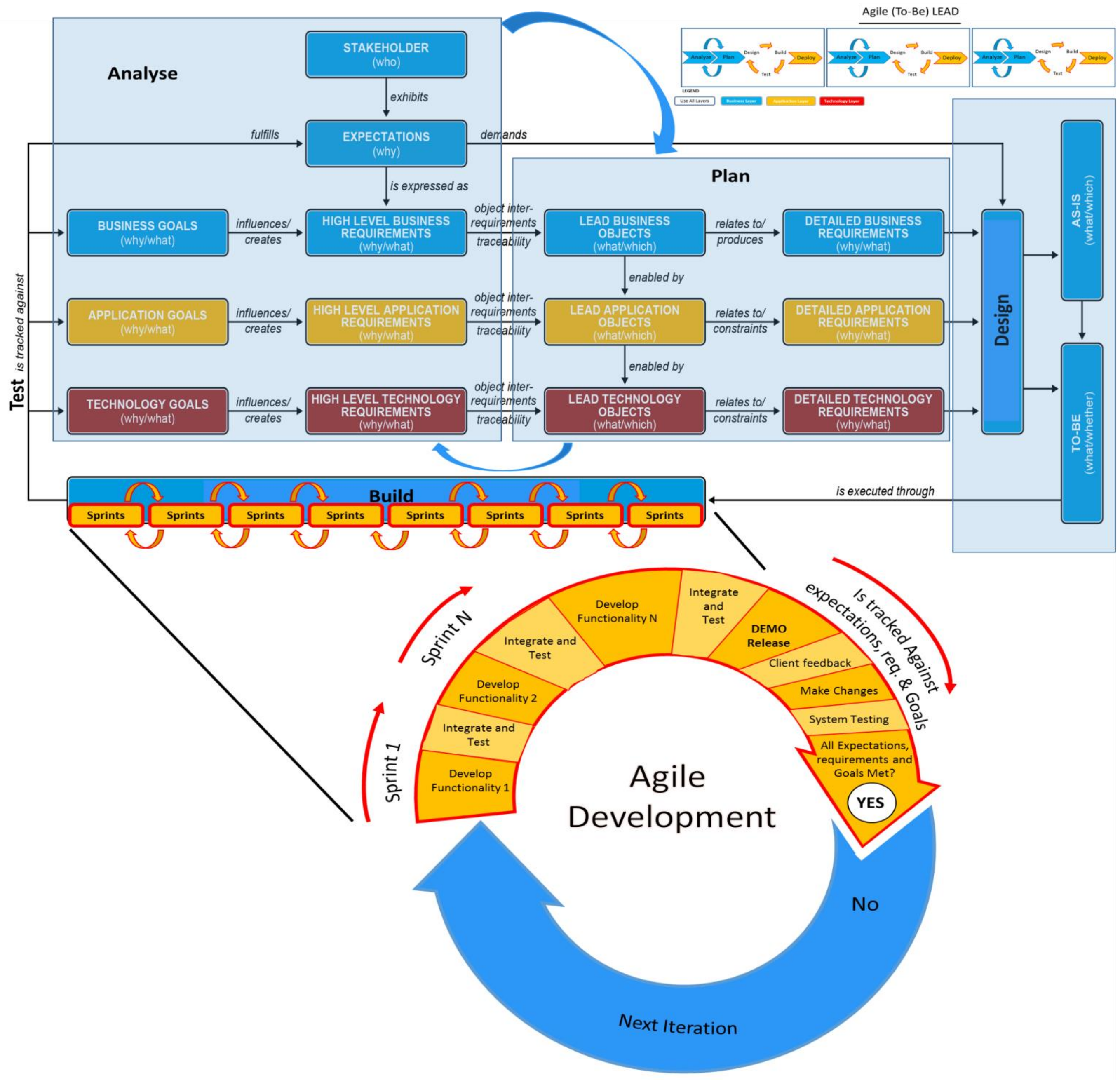
BPM work, we suggest applying Service Oriented Architecture (SOA) and Process Oriented Architecture (POA) principles. A repeatable process or a part of the process can be developed as a “service”. A business process or workflow can be managed through the choreography and orchestration of services. As Qumer and Henderson Sellers[19] 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. Interrogate and Test
6. Demo Release
7. Client Feedback Meeting
8. Retrospective
9. Refactoring
10. System Changes
11. System Testing
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.

[19] Qumer, A. & Henderson-Sellers, B. 2007, '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*, ed H. Fujita and D. Pisanelli, IOS Press, Amsterdam, The Netherlands, pp. 83-92



Agile (To-Be) LEAD
Analyse
STAKEHOLDER (who)
exhibits
EXPECTATIONS (why)
fulfills
demands
is expressed as
BUSINESS GOALS (why/what)
influences/ creates
HIGH LEVEL BUSINESS REQUIREMENTS (why/what)
object inter-requirements traceability
APPLICATION GOALS (why/what)
HIGH LEVEL APPLICATION REQUIREMENTS (why/what)
TECHNOLOGY GOALS (why/what)
HIGH LEVEL TECHNOLOGY REQUIREMENTS (why/what)
Plan
LEAD BUSINESS OBJECTS (what/which)
relates to/ produces
DETAILED BUSINESS REQUIREMENTS (why/what)
enabled by
LEAD APPLICATION OBJECTS (what/which)
relates to/ constraints
DETAILED APPLICATION REQUIREMENTS (why/what)
LEAD TECHNOLOGY OBJECTS (what/which)
DETAILED TECHNOLOGY REQUIREMENTS (why/what)
Design
AS-IS (what/which)
TO-BE (what/whether)
Test is tracked against
Build
Sprints
is executed through
Sprint 1
Sprint N
Develop Functionality 1
Integrate and Test
Develop Functionality 2
Develop Functionality N
DEMO Release
Client feedback
Make Changes
System Testing
All Expectations, requirements and Goals Met?
YES
No
Next Iteration
Agile Development
Is tracked Against expectations, req. & Goals

Figure 6: The Agile BPM build phase.[20]

Agile Testing

While traditional ways of working around testing first does the testing once the whole product or service is developed, agile ways of working focuses on testing the product or service minimum marketable or viable features in small iterations while the development in progress. Agile ways of development focus on automating the testing practices such as automated unit testing, acceptance testing, and integration testing, etc. Right from the beginning the focus shifts from testing documentation to actually testing the working product or service features against requirements so that verification of requirement to ensure conformance of a design and build, whether in terms of business, application, or technology,

[20] LEADing Practice Agile Reference Content #LEAD-ES30006ES

or with respect to the compliance of the objects and design to the requirements in a systematic manner is necessary to provide stakeholders the information about the fulfilment of their expectations and the possible risks within the agile implementation involved. 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 behaviour 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, where 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 e.g. demanded by the 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 fulfil and thereby meet expectations and to verify the quality of the product prior to the 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 prioritise 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 there is a minor issue 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 in the end of release. An individual product or service release deployment can be combined with other releases for different products. Organisations 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 is 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 upfront 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. So 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 there are no direct Objects and Meta Objects that 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 needs to be 100% vendor neutral and agnostic from various vendor solutions.

While 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 the agile circles and apply to the Agile BPM work.

Term	**Definition**
Agile coach	A person responsible for supporting and improving the capability of an organisation to deliver in an agile way.
Agile Driver & Forces (external/internal)	Pressures that arise from outside or inside a system triggering agile approaches.
Backlog	A prioritised 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 organisation functions brought about through a project or other initiative.
Business resource/actor	A specific person, system or organization internal or external that is part or effected by the agile development to the enterprise. This can include that the agile development will influence or impact the resources/actors defined functions and activities.
Business Service	Agile concepts applied to business concepts will impact the change and

Term	Definition
	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	Where 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 towards 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	Where 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 behaviours that are specified to achieve.
Information radiator	A large, highly visible display which 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 where columns represent a state that a user story can be in, 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 which describe how it should behave. Typical examples would be: wished behaviour, process security, accessibility, usability, availability, response times, etc.
Owner	The person who is ultimately responsible for prioritization and acceptance of delivered features on a given process or project.
Performance Expectations	While 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 where 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

Term	Definition
	service
Retrospective	A retrospective is a focused session where 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	Where 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 pre-determined 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 by handy for tracking specific process design and development tasks. Each task belongs to one and only one user story. Generally speaking, 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.
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 non-functional 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 non-functional 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

Table 2: The most common terms that would be used within agile BPM[21]

[21] Taken from the LEADing Practice Agile Reference Content LEAD-ES30006ES

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 based on the fact that the CoE and teams have a different way of thinking, working, modelling and implementation. 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[22] and then was updated in 2010 as "AAIM Version 2.0".[23] The 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). This section discusses the six stages or levels of the AAIM V2.0 followed by the agile business process management.

Agility Adoption and Improvement Model

The AAIM is structured into white, green, and black blocks and 6 levels (from 0 to level 6 – Tentative to Progress). The colours indicate the levels where the white blocks are the levels where initial experience in critical aspects of agility is garnered, whereas the green blocks are the levels where the agile practices are established and entrenched, whereas the black blocks show those levels where 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 organisation or team. The following section discusses the AAIM in the context of agile BPM capability establishment.

[22] Qumer, A. & Henderson-Sellers, B. 2007, '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*, ed H. Fujita and D. Pisanelli, IOS Press, Amsterdam, The Netherlands, pp. 83-92.

[23] Qumer, A. 2010, A Framework to Assist in the Assessment and Tailoring of Agile Software Development Methods, PhD Thesis, UTS.

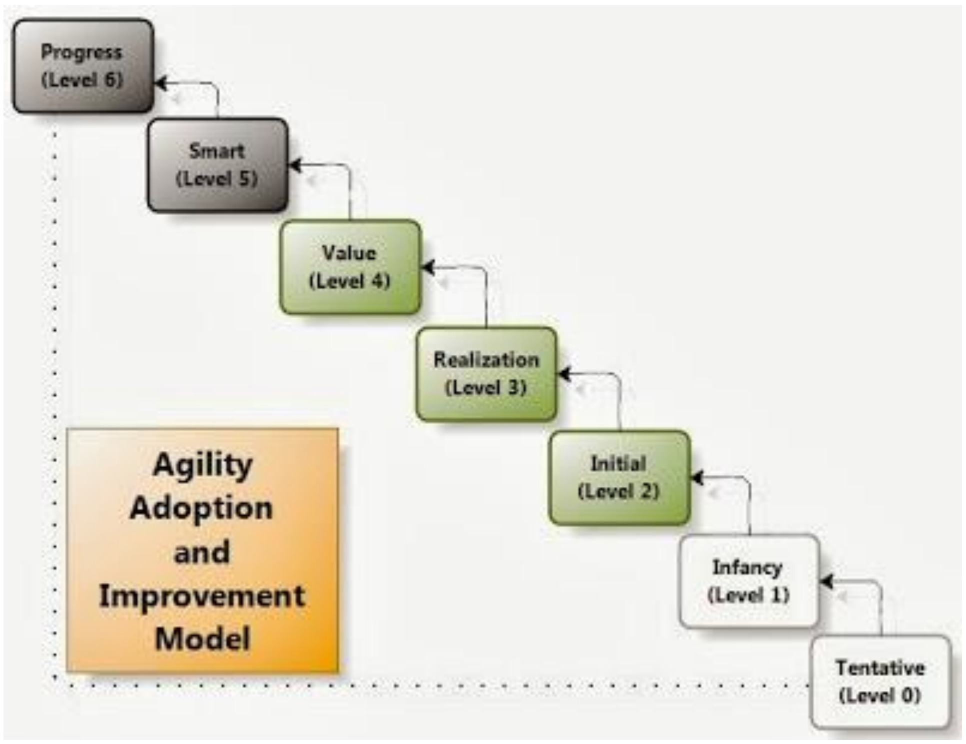


Figure 7: The Agility Adoption and Improvement Model V2.0[24]

- 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, 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 artefacts 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 there is a well-established communication-oriented culture in the organization (e.g. established at level 2).
- Value (Level 4)

[24] Gill, A.Q., Bunker, D. 2013, 'SaaS Requirements Engineering for Agile Development' in Xiaofeng Wang, Nour Ali, Isidro Ramos, Richard Vidgen (eds), *Agile and Lean Service-Oriented Development: Foundations, Theory, and Practice,* IGI, USA, pp. 64-93.

This level focuses on establishing self-organising BPM teams by adopting agile principles, roles, practices, and tools. Critical here is the fact that self-organisation 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, 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 to 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 at the same time to avoid imposing "Agile" on teams

Conclusion

Traditional BPM ways of working focus on detailed upfront planning, requirements analysis, process analysis, process design, process implementation, and continuous improvement to adjust changes. In other words, take a waterfall approach to Business Process Management Here, the assumption is that all the requirements for the process work are fixed or known or complete. A lot of time and resources are spent upfront 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. Organisations need to be agile in response to such changing business environment and expectations. We have therefore focused in this chapter on the question of why we need to be agile and where, 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 applies kaizen principles of continuous improvement directly in their way of working. In that they learn from what they do and how, adapts and re-shapes their manner of delivering the project and involves stakeholders in a new way e.g. as co-participants in the process.