Does IBM BPM = agile development?
The subtle differences between iterative business process management and agile development practices

Scott Glen
February 04, 2016

Learn some of the subtle differences in the approaches used to deliver IBM® Business Process Manager (BPM) processes and agile applications. This tutorial draws on experiences gained from more than a decade of IBM BPM customer engagements from across Europe, the Middle East and Africa.

Introduction

The typical IBM BPM development methodology follows an iterative and flexible path, often described as being agile. However there are subtle differences between the IBM BPM approach and those defined by agile methodologies, such as Scrum or XP. This article highlights key aspects of each approach and explains where IBM BPM projects can benefit by adopting a more bespoke methodology.

The agile way

Agile methodologies began to surface in earnest during the mid-1990s as a reaction to a historical mass of troubled monolithic waterfall projects. These failures were often attributed to issues with scope and requirements management, giving rise to the following problems:

- poorly understood
- prone to change
- complex in scope

For large projects, it can take several years to fully analyze requirements and begin to form a solution design. However, by then, those very requirements can often be invalid, because the business has already reacted to a changing commercial environment. The term "analysis paralysis" was coined to describe projects that were continuously locked in a repeating pattern of analysis and design and that failed to deliver anything of tangible value.

Rather than try to "boil the ocean" by building the entire solution in one delivery, the notion of a more flexible approach surfaced, which among other things could achieve the following points:
• Deliver solutions through continuous and incremental growth.
• Regularly interlock with project sponsors, analysts and end users.
• Avoid the need to conduct a full requirements analysis.
• Welcome changing requirements, even in the latter stages of solution delivery.

The last point is key: it coined the name *agile*, namely the ability to be flexible and to absorb changing or unknown requirements as a normal part of the software development lifecycle.

**Scope management**

There are typically three interrelated variables in a software projects: scope, resources, and time, as shown in Figure 1. Adjusting any one will have a direct impact on the others.

**Figure 1. Project dimensions**

Traditional delivery methodologies try to allocate additional resources or extend the project time scales to deliver the agreed scope, which is seen as an immovable dimension. However, the agile development process is all about being able to accommodate changes in scope, or to deliver a solution without necessarily having a detailed understanding of the scope in the first place. With an agile approach, scope is no longer an immutable factor. Projects can avoid a prolonged period of analysis and to begin to deliver incremental value to the business with a minimal amount of initial investment.

**Agile practices**

The following range of innovative working practices were often associated with the agile development process:

• **Pair programming**: Developers work in pairs, typically one typing and one doing the design thinking "on demand."
• **Test-driven development**: Before writing a module of code, developers first write the tests and embed them into the codebase.
• **Automated, nightly builds**: The entire source code is compiled nightly, and regressions tests are automatically executed to verify the code’s integrity.
• **User stories**: Requirements are captured through interviewing analysts and lead users and are recorded in lightweight stories.
• **Planet "Post-It"**: Team room walls are covered with Post-It notes, defining the stories to be completed by week and phase. These notes are regularly rearranged based on their relative importance and complexity.
• **Daily "Stand Ups"**: Developers and architects conduct "on-the-fly" design sessions in daily stand-up meetings.
**Refactoring**

One by-product of the agile approach is the need for refactoring. As each subsequent user story is implemented, it might affect the existing codebase, causing modules to be rewritten to accommodate the new functionality. This refactoring stems from the desire to avoid an extended analysis and design phase and to start delivering code quickly to provide accelerated business value.

Refactoring in this way should not be viewed negatively. While it might seem inefficient from a programming perspective, it is a valid price to pay for the flexibility that it brings in enabling the project to absorb evolving requirements.

**Sprints**

A sprint is a development cycle that is found in agile methodologies, including the Scrum approach. Scrum-based sprints are time boxed development cycles, of a fixed duration (typically between 2 and 4 weeks), during which a scrum team commits to delivering specific user stories. Sprints are designed to deliver complete areas of functionality, known as potentially shippable increments (sometimes just called PSI), which could be released to the business if required.

Alternative development approaches, such as Kanban teams, provide an even more flexible and demand-driven approach to implementing stories, which can form part of continuous delivery cycle.

Whatever the approach, the creating production-quality code that can be released incrementally is a key objective.

**The process-driven approach**

The methodology commonly associated with IBM BPM development has successfully adopted the following agile practices:

- Iterative implementation
- Business and end user alignment
- Reduction of up-front analysis and design
- User story-driven requirements capture

However, there are a number of subtle differences and variations in the way IBM BPM processes and agile applications are delivered.

**Strong requirements**

By its very nature, IBM BPM almost always focuses on a well-defined specification: namely, the process itself. The requirements are typically well understood by the business and are often a response to inefficient and suboptimal operations.

A process owner is typically responsible for the end-to-end process and is often supported by a number of business subject matter experts who understand the finer details of its operation. Above
all else, processes, by definition, do not change that often. They might use rules to introduce dynamic behavior, but structurally they are often stable. Therefore, while playbacks are used to embrace business feedback, do not expect wholesale change in the process during development.

**Business expectations**

When discussing agility in IBM BPM, it is not primarily about the ability to react to changing or unknown requirements. Of course, each project attempts to absorb feedback between iterations. However, this feedback almost never introduces wholesale change. It typically refines or clarifies existing requirements. The agility derived from IBM BPM should be *business agility*; delivering flexible, future-proof processes that grow incrementally and continuously to provide a new level operational responsiveness.

**Processes or applications**

Business process management is not about building "applications," which is one of the most common anti-patterns observed in IBM engagements with clients. Processes are typically uni-directional and task-based. They start, do something, typically involve a variety of roles, and finish. Applications can also be task-based, but they often have an interaction pattern where one user role repeatedly returns to a central screen to perform similar work activities. A customer relationship management (CRM) system is an application, a flight booking system is an application, and Order to Cash is a process. A good use case for IBM BPM is one where it orchestrates multiple applications and involves multiple roles.

Yes, IBM BPM can be used to build applications – but it was not designed as a rapid application development (RAD) platform or as a fourth-generation programming language (4GL). Care should be taken when using IBM BPM in this way, because such projects often fall into the realm of the 20/80 rule: the last 20% of requirements take 80% of the effort.

**Solution design**

Because IBM Process Designer in IBM BPM is so intuitive, there is often a temptation to circumvent all design activities and proceed straight to the keyboard and begin process implementation on day 1. Experience has taught that you can't skip design activities. There is still a need for distinct process analysis, modelling, and solution design activities. During this initial phase of the project (known as the "blueprint" or "Iteration 0") the following activities are conducted:

- **Process analysis and modelling**: The delivery team and the line of business need to collaborate to understand the project's requirements and graphically model the process. Ideally this activity should be led by a dedicated business analyst from the project team who is familiar with process modelling and optimization techniques, and who can use tools such as Blueworks Live.
- **Define user stories**: The process model alone cannot comprehensively capture all the requirements. A series of user stories must be written to define how the various roles interact with the process.
• **Construct wireframe UIs**: Simple user interface mockups can be used to enhance playbacks and help immerse business sponsors in the new process.

• **Business metrics**: How will we validate that the IBM BPM implementation has actually delivered business value? Key Performance Indicators (KPIs) must be captured with the business sponsors to provide insight into future operations.

• **Business objects**: While there is no need to model every aspect of the business data contained in the solution at this stage, you must ensure that the key objects have been identified and that they can support the desired business metrics. Refactoring business objects at a later date, especially those used in complex service interfaces, can be a time consuming experience.

• **Solution outline**: Solution architecture is often overlooked, but a degree of macro design is required to capture the technical aspects of the solution, including non-functional requirements, and to identify and validate points of integration with supporting operational systems.

• **Playback 0**: The scope, process, design and plan are all verified with key business sponsors.

### Story points

On large engagements it is not uncommon for the project management team to work with a group of unfamiliar development resources, each of whom has a different level of skill and experience that can dramatically affect their ability to deliver code. In such cases it can be beneficial for the design team to use an arbitrary unit of measurement when estimating the complexity of work packages. After the work has been assigned to named resources, the resource-agnostic estimations can then be converted into developer-specific figures.

Most agile projects use some variation of story points as a way to assess the complexity of their user stories. Story points are abstract numerical values that define the complexity of the story when compared to the other stories under development, but they do not directly reflect the number of days required to deliver the story. Developers also have their own velocity, a numerical value that reflects their ability to deliver code.

To calculate the duration in days that it will take a specific developer to deliver a specific story, simply divide the story points by the developer’s own velocity. Therefore if you have a story that has been allocated 8 points, and assigned to a junior developer with a velocity of 2, then it might take the developer $8 \div 2 = 4$ days to deliver the story. A more senior developer with a velocity of 4 would code the same story within half the time.

Story points can work well where you have a dedicated design team who is responsible for estimating all stories and who does not yet know the capabilities of the development resources. However, story points do introduce the following challenges:

• **Duration**: Business sponsors like to deal in actualities. If business sponsors ask “When will the project be delivered?” they want a calendar date, not a value in story points.

• **Estimation**: Using story points brings an additional level of estimation into the project planning. Not only do you estimate each story, but you also have to estimate each developer’s skill level. The most important metric – namely the number of days to deliver a work package
– is therefore a product of two estimations, each of which will have some factor of uncertainty, potentially leading to a greater degree of inaccuracy in the overall estimate.

By their nature, IBM BPM projects should cultivate a close working relationship with the associated line of business. Projects aim to deliver business value early and often, which helps sell the IBM BPM vision and builds confidence in the team’s delivery capability across the enterprise. This approach has the following affect on the anatomy of a typical IBM BPM project:

- **Process-centric project:** By implementing the core process flow you implicitly deliver a large percentage of the solution, meaning that you typically do not have hundreds of user stories left to manage, estimate, and develop across your iterations.

- **Solution acceleration:** The model-driven approach of the IBM BPM product helps accelerate the solution delivery. Preach short development cycles and incremental deployment, with releases often occurring somewhere between 10 to 16 weeks. Work packages are therefore smaller and easier to estimate accurately.

- **Team composition:** IBM BPM projects typically have small, close-knit teams, with high visibility of their technical capabilities. The developers often work collaboratively with the project manager and solution architect on story estimations and should have a high degree of confidence in their ability to deliver.

For these reasons, I have some reservations on the value of story points in IBM BPM projects. Introducing an additional level of estimation undoubtedly increases the opportunity for inaccuracies, and at the same time provides no value to business sponsors. It is often much simpler and easier to estimate in "average developer days", taking account of the skill level in the team.

**Project planning**

Within an IBM BPM program there is often an explicit need to provide the business sponsors with an accurate delivery schedule, accompanied by clear expectations around the evolution of the process. However you cannot estimate the project duration with 100% certainty – nor should you want to – because you still want the ability to react to evolving requirements during implementation. The project manager and solution architect should have about 80% confidence in the delivery time scales and resource allocation. Therefore, the plan should not be defined to the n<sup>th</sup> degree, but it should provide the business with confidence that the solution will be delivered broadly on schedule.

Following the Blueprint phase, you must also plan out the iterations in order to efficiently allocate resources. Often members of the IBM BPM team transition in and out as needed, and subject matter experts on topics such as business rules and user experience are not needed full time. Such resources need to be planned accurately, because they are often shared across multiple projects, perhaps as part of an IBM BPM program delivered through a Center of Excellence.

**Stand up meetings**

A key agile feature is the daily stand up meeting, or the scrum meeting. This meeting can be invaluable for the following main reasons:
• **Refactoring agility costs:** By its very nature, an agile approach introduces a varying degrees of rework, therefore it is important that any impact from code refactoring is publicised across the team, and that the consequences discussed openly amongst all interested parties.

• **Design sessions:** The agile approach often foregoes a detailed design phase, preferring to design "on demand" during the daily sessions where the group can exchange technical ideas on how best to approach specific stories.

On IBM BPM projects you still want to accommodate business change where appropriate, but the refactoring and on-the-fly design should be minimized by the initial Blueprint design and planning exercise. As a consequence, you often see stand up meetings degenerate into "round table meetings" where each participant simply stands up and states what they are working on. This meeting does ensure that everyone knows what everyone else is working on – but is this time well spent every day?

The cost of these meetings should not be underestimated, especially when trying to deliver in an accelerated timeframe. As an example, if you envision a typical, small 12-week project, with a team of 8 resources, the accumulated cost of daily meetings could be: 8 people * 0.5 hours per day * 5 days * 12 weeks = 240 hours

Communication is vital, but on small projects where the entire team is normally co-located in one room, project knowledge is typically disseminated implicitly. The team members are aware of each other's successes and challenges, and the solution architect is responsible for bringing specific resources together as needed to address technical issues. Irrespective of when they are on-boarded, each resource should also be aware of the big picture and should be exposed to Playback 0 materials so they have a full understanding of how their work contributes to the overall solution.

Therefore, carefully assess the need for daily meetings and consider the experience, location, and social factors within the team before investing large amounts of project time. Weekly meetings can often provide the best trade off. In the weeks without a playback it can be beneficial to briefly gather the entire team to discuss the overall project status and air any issues.

**Playbacks are great, but they are really nothing new**

Software projects have almost always demonstrated progress to clients, although perhaps not with such a scheduled approach. What is new from the client's perspective is that they don't have to raise a project change request (PCR) if they just want a button moved 3 pixels!

Playbacks, however, are not user acceptance tests, and you should not prepare for them in that way. The playback should be rehearsed, and you should ensure that you can demonstrate incremental growth across the solution. However they are an opportunity to gather vital business feedback and not do (usually) represent a formal sign-off or contractual milestone. Of course, the exception might be the final playback, which can constitute a level of user acceptance and should be treated with more diligence.

In some cases entire project teams download tools and spending several days preparing for each playback with infinite precision. In these cases the business sponsors are almost viewed
with trepidation, as a feared foe, rather than as a welcome partner in the extended team. While this level of preparation can be commendable, it is often not affordable. Based on the previous example, the mathematics are again simple: 8 people * 8 hours * 3 days preparation * 4 playbacks = 864 hours

This example means that about 109 days would be dedicated to preparation and demonstration (and not development!), a luxury that is not often possible. Playbacks are great: embrace them, maximize their benefit, but don’t fear them.

**IBM BPM iterations are not agile sprints**

Both are short-lived development cycles, typically between 2-3 weeks in duration, where the overall solution is incrementally constructed by the implementation of user stories. However at the end of an iteration, the solution is not necessarily designed to be in a state that can be released to the business.

By comparison, agile sprints are aimed at continuously delivering release-ready artifacts, which empowers the business sponsors to say whether they want to go live with what they have so far in case a specific piece of valuable functionality has been delivered. This works for some application deliveries, but IBM BPM is process-centric and so treating IBM BPM iterations as agile sprints could cause the following issues:

- **Releasing half-finished processes:** IBM BPM projects attempt to deliver each end-to-end process as soon as possible, to enable the business to envisage the final solution. Envisioning the solution might initially require the use of stubs and mock services that cannot realistically be released.
- **Increased testing effort:** More releases require more testing, specifically formal user acceptance testing conducted by the organization’s business sponsor, lead user, or analysts.
- **Perpetual change management:** If you release an enterprise-wide process to the business at the end of every 2 or 3 week sprint, then the business sponsors could easily find themselves locked in a cycle of change, continuously reacting to operational enhancements that might actually hinder their ability to perform their function.

Too much change at a process level only frustrates business sponsors. Therefore, you need to establish a healthy balance that can support the desire for operational improvements without placing an unnecessary impact on the business.

This balance can often be found within a program of continuous process improvement, which facilitates incremental growth over a prolonged period.

**Process optimization**

IBM BPM is a natural implementation partner for a Lean or Six Sigma process improvement program. Lean analysis is typically conducted over a range of processes, attempting to deliver improved process control and measurement, performance visibility and operational efficiencies. Although often conducted as separate exercises, Lean and IBM BPM can be combined within a single iterative delivery lifecycle. See the references section for more details.
Continuous process improvement

Above all, IBM BPM development should be seen as a program of continuous process improvement and not a single delivery vehicle. Applications are typically delivered and usually patched, but they can exist for long periods of time without being upgraded or enhanced. With IBM BPM you want to implement quickly, gather business insight through business monitoring, deliver an improved process, and repeat.

Monitoring is the key aspect here: the business sponsors must be given time to evaluate the newly released process, typically analyzing several months of business metrics and operational statistics before potentially identifying further improvements.

Process implementation roadmap

As IBM BPM technology matures and its adoption becomes more mainstream, an increasingly number of clients embark on IBM BPM Programs. These programs typically involve a small number of teams delivering multiple processes in parallel, often adopting a factory approach with regular deployments throughout the year. The analysis and design is always conducted in-house, as close to the business as possible, but development teams can be located off shore, using outsourced resources or business partners to provide a cost effective way to accelerate IBM BPM delivery.

As the scale of delivery and expenditure increases, it becomes even more important to select the right processes in order to create business value and return on investment in a short timeframe. Achieve these goals through a series of processes assessment workshops, known collectively as Process Inventory, Variation Analysis, and Roadmap (PIVAR), where a number of candidate processes are compared to an agreed set of criteria to establish which derive the most bang per buck. For more information, see the Business Process Management Design Guide Using IBM Business Process Manager IBM Redbooks publication. The output of these workshops can be plotted on an impact-effort graph, assisting in the creation of an IBM BPM process implementation roadmap, as shown in Figure 2.
Figure 2. Process prioritization

Ideally, seek to prioritize those processes that fit into the "magic quadrant" at the top left of the diagram – delivering high impact with low effort. The IBM BPM program therefore provides an ideal vehicle to implement continuous process improvement, enabling the business to schedule incremental improvements in processes and to ensure in advance that operations are ready to receive these enhancements at predetermined points throughout the year. Substantially different from the traditional IT delivery model, yet more disciplined than a typical agile approach, the process prioritization strikes a happy medium between flexibility and predictability.

Conclusion

While the IBM BPM delivery approach borrows heavily from agile methods, the following differences were highlighted:

- **Well-defined requirements**: The process provides a well understood core business requirement that is not prone to change.
- **Solution design**: A degree of up-front solution design is important to provide accurate scoping and planning, minimizing refactoring.
- **Continuous process improvement**: Process development and optimization is an ongoing activity, using business monitoring to identify further refinement opportunities.
- **IBM BPM program**: A scalable delivery approach focuses on scheduled, business-aligned IBM BPM solution deployments.

It is imperative that a suitable delivery methodology is defined, documented, and evangelized before you embark on an IBM BPM project or program. One size rarely fits all, so you might need to customize your existing agile approach or engage with IBM or an IBM Business Partner for a Method Adoption Workshop (MAW) or a Center of Excellence (CoE) engagement to define an
approach that is tailored to the needs of your organization. Whatever path you take, address the factors discussed in this article to maximize the efficiency of your IBM BPM journey.

Acknowledgements

The author would like to thank Paul de Wildt for his review and comments.
Related topics

- Evaluating BPM applications: BPM design reviews and Rubik's Cubes (developerWorks article)
- A business process management approach to enterprise process modernization (developerWorks article)
- BPM and Lean: a powerful combination for process improvement (developerWorks article)
- IBM Business Process Manager Developer Center
- IBM Business Process Manager product page