

BPMS IMPLEMENTATION WITH AN AGILE METHODOLOGY

Amentra, a Red Hat Company
February 5, 2009

Despite significant hype about business users being able to automate processes by dragging and dropping nodes within an easy-to-use interface and without IT involvement, the reality of business process management remains quite different. Business Process Management Suite (BPMS) implementations typically involve many integration and machine-level dependencies in addition to human tasks. Therefore, despite the best intentions of the tool designers, such efforts often become software development projects that require an understanding of the underlying methodologies.

Given the true nature of BPMS projects, many of the best practices and methodologies that the software engineering industry has developed over the last few decades can (and should) be applied to BPMS implementations. Modern software engineering favors agile methodologies, which naturally tend to bring business and technical groups closer together. The purpose of this article is to explore how the core tenets of agile methodologies can be applied to BPMS projects.

AGILE OVERVIEW

At Amentra, we take a pragmatic approach to the software development process. Through years of experience designing and developing custom software in our systems development and service-oriented architecture (SOA) practices, Amentra has learned that our customers benefit from an approach that integrates the best features of several prominent methodologies. Within the realm of Agile techniques, Scrum and eXtreme Programming (XP) each have strong support. We accept tenets of both and even blend more structured documentation into our process for organizations that must satisfy regulatory requirements. When referring to “agile development” in this article, we mean a general approach that combines these techniques and philosophies, not a specific methodology such as Scrum or XP.

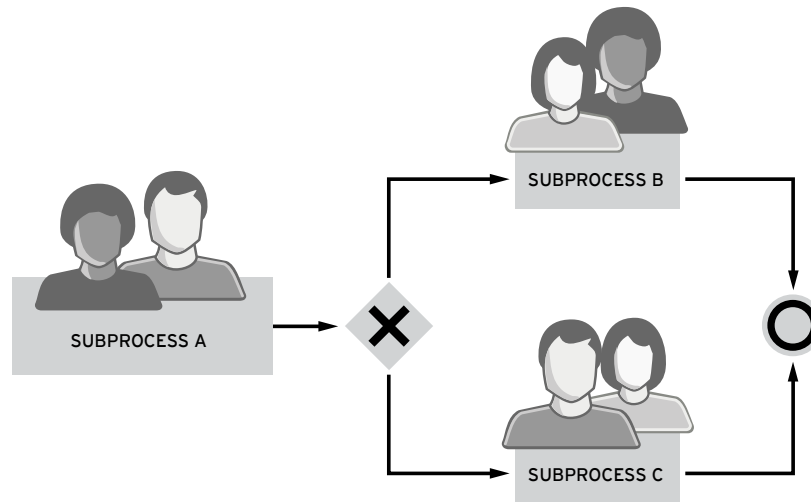
The core concept of all agile methodologies is development in quick iterations to produce working software. The process should favor producing software over documentation and should also favor immediate collection of feedback during the development cycle over. Post-implementation feedback, though valuable, often arrives months too late. During an agile implementation, techniques such as test-driven development and pair-programming enable responsiveness to changing requirements, a focus on quality, and a team approach to development.

TEAM STRUCTURE

To derive the most benefit from agile development, organizations should structure their teams to incorporate both business and technical participants. For production-quality processes, resources with technical knowledge need to be involved. There are many examples of BPMS components that are best utilized by people with technical knowledge. For instance, consider the algebraic nature of business rules, the Boolean logic of gateways (because the BPMN term “XOR” does not resonate with most business users), the frequent necessity of some true code for integrations, and the inevitable front-end hacks for data validation.

We will call the team members who specialize in the technical details “technologists.” The business side will be represented by resources who understand (or can ask the right questions to understand) the business process. We will call these team members “analysts,” even though they may in fact be business users. We call collaborative work between a pair comprising an analyst and a technologist “pair modeling,” a term derived from the XP concept of pair programming.

FIGURE 1: AGILE TEAMS WORKING ON SUBPROCESSES

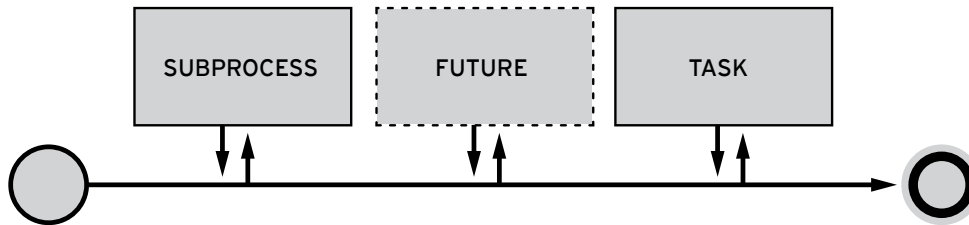


The overall process implementation team is made up of a fair number of people. In addition to project managers and quality assurance representatives, several people on the team will fill technologist and analyst roles. The analysts are each responsible for a segment of the overall process, and the technologists have usually been chosen in similar numbers so that such segments can be automated in parallel. While many methodologies favor democratic interaction among the larger group throughout the process, we favor breaking into smaller teams. A pair of resources containing one technologist and one analyst should work on each process segment. This pair can solve problems together and quickly develop working processes, which they can then demonstrate to the larger group for feedback. We have found that the ideal mix is to spend about half the day working together as a pair and half the day working individually. When working individually, the business analyst can follow up with other resources to define more details of the process, and the technologist can work on technical details of the implementation, such as integration and advanced user-interface configuration.

ITERATIONS

An iteration--the basic unit of work on an agile project--can be thought of as a mini-release, a piece of software that is developed end-to-end over a short period of time. In a BPMS implementation, iterations generally comprise one or more subprocesses. A key goal of agile, iterative development is that everything that is developed during an iteration works at the end of the iteration and can be used by end users. If a subprocess is to be partially completed, it should be in a state where the main flows are functional, but some optional features have been deferred until a subsequent iteration. These optional features, along with feature requests that are added during an iteration by business analysts and other stakeholders, should be prioritized and tracked in a repository or document called a backlog.

FIGURE 2: A PROCESS, SOME OF WHICH IS IN THIS ITERATION, SOME IN THE FUTURE



At the beginning of each iteration, the project manager defines the scope by choosing the subprocesses and items from the backlog that the team believes to be implementable during the iteration. Entire aspects of the BPM implementation, such as role definition or reporting, should not be deferred to subsequent implementations. The goal of this approach is to give the users a feel for what their interactions with the system are going to be so that they can provide feedback. These users should include both the participants of the process and the consumers of the dashboards that report on the process.

DOCUMENTATION

In agile terms, the process model is the documentation. Because the analyst and technologist work on the model together, the analyst is not required to create external documentation that the technologist will implement. Similarly, most BPMS modelers have a print or export feature that allows the model to be shown to end users to give them context for what the process does. Intuitive names for nodes, conditional transitions, and subprocesses provide a level of clarity that allows the model to be self-documenting. Occasional annotation to explain a complex part of the process is encouraged, but relying on notes often indicates that the process can (and should) be simplified.

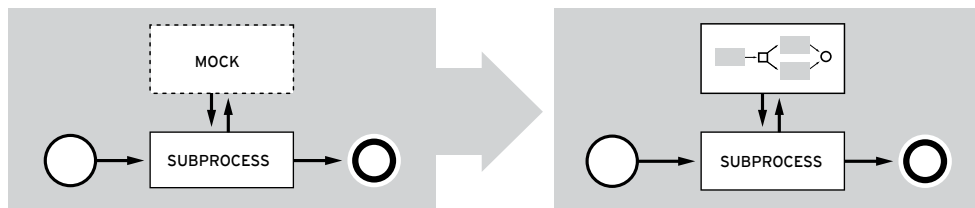
MODULARITY

Larger processes should be decomposed into smaller subprocesses. These building blocks are often easier for analysts and business users to conceptualize, and the practice mirrors practices common to software design methodologies that promote modularity and reuse. In the BPM world, these smaller units also are less likely to suffer from conflicts when multiple developers edit processes, making the maintenance of process artifacts less error-prone. Small processes can also be built and tested quickly, with fewer paths to consider in each process. In an agile methodology, this structure facilitates interaction between the technologist and the analyst by allowing them to quickly build working software together. Most BPMSes couple a runtime execution environment with the modeler that supports this interaction pattern.

MOCKING AND TESTING

A common implementation technique in agile development is mocking, an aspect of unit testing that allows the developer to assume the systems he or she is interacting with will work properly. This allows the focus to remain on making the newly developed code work. In the BPM world, we use a similar tactic to speed up development while the analyst and technologist are pair modeling. When the model requires an integration task, the pair can stub the task out with a script task that simply sets the desired process variables. Later, the technologist can replace the script task with an integration task that handles the underlying technical aspects of the process. This may even involve SOA implementation.

FIGURE 3: A MOCK IMPLEMENTATION OF A SUBPROCESS IS CONVERTED INTO A REAL IMPLEMENTATION

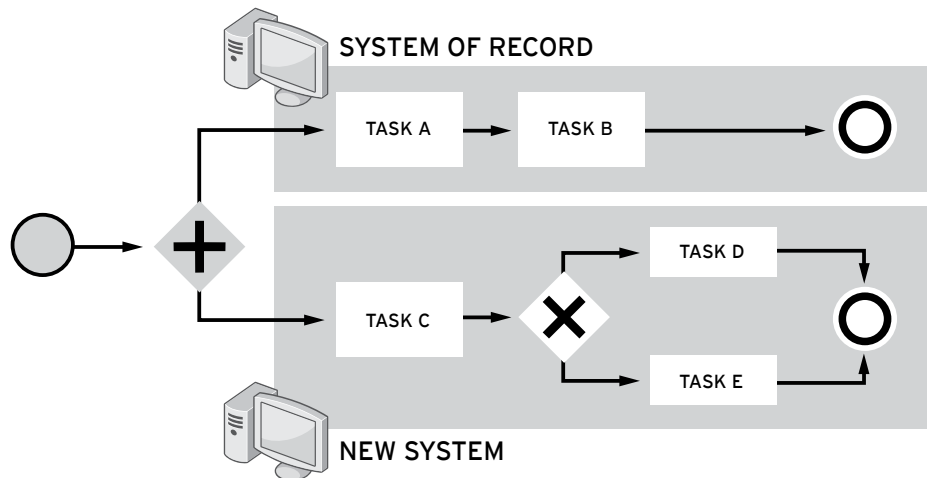


Mocking with a script task allows the pair to focus on the process flow and interactions between nodes rather than the implementation of each node. Implementation is usually the technologist’s individual prerogative. Similarly, the pair can develop screens for human-oriented tasks using bare-bones interfaces. The technologist can work on more detailed styling (and validation) during non-pair time. This practice allows the pair to focus on the data elements of the tasks.

A BPMS design constraint that often interferes with the runtime aspects of pair modeling is the way the execution engine handles role-based assignment. When stepping through a process, the user must be a member of each role in order to see the corresponding tasks. To work around this limitation, we suggest creating a dummy user who is a member of every role. This user will therefore be assigned all tasks, which will make stepping through the process much easier, both for the pair during development and during demonstrations. In order for this tactic to be effective, all assignments must be made to roles instead of individuals. This approach is a best-practice of BPMS implementation in agile or non-agile development methodologies and should be followed consistently.

HYBRID DEPLOYMENTS

FIGURE 4: A HYBRID SYSTEM, SOME IN BPM AND SOME IN THE AS-IS IMPLEMENTATION TECHNOLOGY



If the automated process being implemented during a typical BPMS project is a replacement for an existing system, the two may co-exist through several releases, allowing quick, incremental delivery of BPM results. A common strategy is to keep the existing system’s data store as the system-of-record, writing back data from the BPMS process through SQL nodes as necessary.

As processes replace some modules of the system, BPM role-based task execution and reporting on those processes is enabled. This approach lets the users adjust to the BPM paradigm in small increments, avoiding having to relearn how to do their daily work all at once. It also provides an early opportunity for feedback on user experience aspects of process execution, such as the portal interaction, inbox options and navigation, and task user interface organization. Significant effort can be saved by adapting to feedback after only a small piece of the process has been deployed--instead of waiting until the entire system has been developed. This type of approach is at the heart of iterative, agile methodologies.

CONCLUSION

Because the BPM paradigm emphasizes collaboration between IT and business stakeholders as well as rapid response to change, BPMS implementations are ideal candidates for an agile development methodology. This article has shown a few techniques Amentra uses to implement BPMS processes within the framework of an agile methodology. For more information on how Amentra can help your organization improve its process maturity or maximize the return on investment from the purchase of a BPMS, please email bpm.info@amentra.com.



AMENTRA CORPORATE HEADQUARTERS

Riverside on the James
Suite 701
1001 Haxall Point
Richmond, VA 23219

Phone: (804) 355-9360
Fax: (804) 355-9361