DavidChappell
& Associates

APPLICATION LIFECYCLE MANAGEMENT AS A BUSINESS PROCESS

DAVID CHAPPELL

SPONSORED BY MICROSOFT CORPORATION

COPYRIGHT © 2010 CHAPPELL & ASSOCIATES

Not too long ago, the bond rating agency Moody's disclosed that it had incorrectly assigned a triple A rating to billions of dollars in debt products. In fact, the correct ratings were as much as four levels lower. The error led some of Moody's customers to invest in products that were significantly riskier than they expected. It also led to front-page coverage in the Financial Times and a blot on Moody's reputation.

The incorrect ratings weren't the result of an analyst making poor judgments on these products. Rather, they were caused by a bug in the custom application that Moody's used to model risk. As business processes like this one become more dependent on software, getting that software right gets ever more important. And because the most important software in an organization is typically developed in-house, doing this well has become a foundation for business success.

The set of activities required to create and run custom applications is known as *application lifecycle management (ALM)*. ALM supports many business processes today, and it's also a critical business process in its own right. Any organization that creates custom software should take the ALM process at least as seriously as it does any other important business process. Being better than your competitors at creating custom software can provide a significant competitive advantage.

HOW ALM SUPPORTS BUSINESS PROCESSES

Organizations are defined by their business processes. To understand what an organization does, you need to understand its processes. In a very real sense, improving that organization—making it more responsive to change, more profitable, and more valuable—means improving its business processes.

Supporting those processes with software can help. In a recent Harvard Business Review article[1], Andrew McAfee and Erik Brynjolfsson describe their research into why this is so. One of their key findings is that because software can be easily installed throughout an organization, an improved business process can be quickly replicated in every store, every factory, or every office. Just as important, a software-based process is self-enforcing. Rather than relying on, say, training manuals and hoping that employees follow a new process, the software itself carries out the process in a consistent way. According to McAfee and Brynjolfsson, the ability to embed a better business process in software, then replicate it reliably across an organization, is a primary way to create competitive advantage with information technology.

Yet getting this kind of unique advantage from off-the-shelf software is hard. Almost by definition, packaged software implements well-understood, common processes. And since the same package is available to everybody, differentiating your organization with packaged software isn't easy. Instead, real differentiation comes from unique business processes that are better than those used by your competitors. For unique processes supported by software—which today means nearly all unique processes—custom applications are required. This is exactly why the ALM process is so important, as Figure 1 illustrates.

[1] "Investing in the IT That Makes a Competitive Difference", Harvard Business Review, July-August 2008

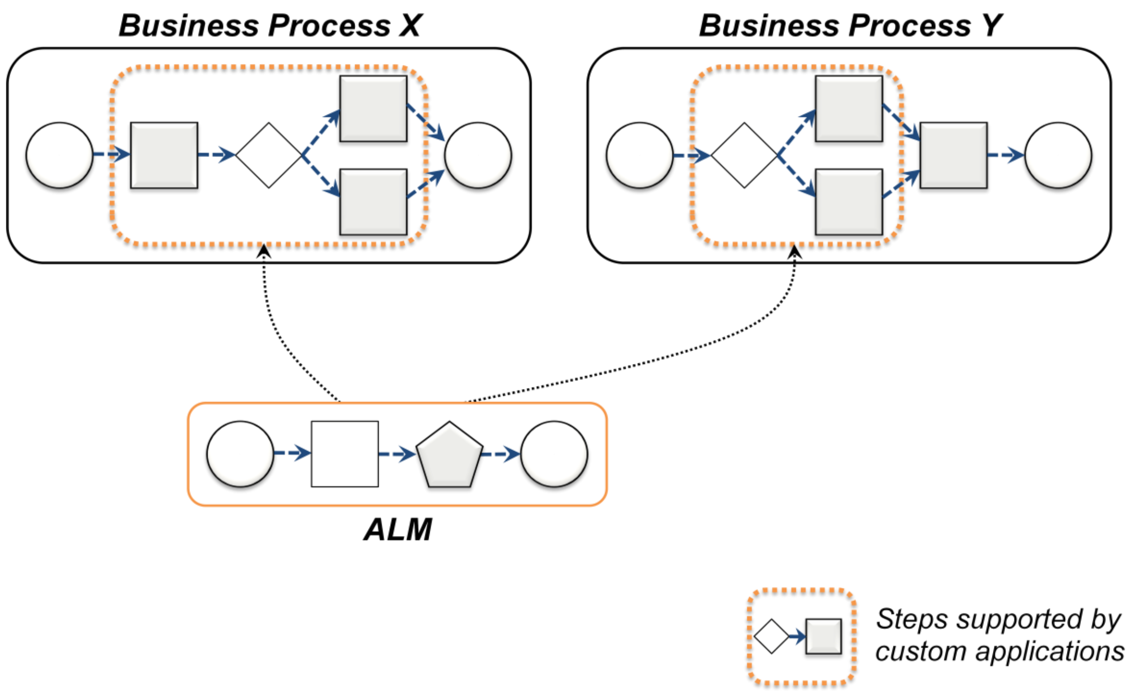

Figure 1: The output of the ALM process—custom applications—provides essential support for other business processes.

As the figure shows, custom applications might underlie all of the steps in a business process, as in process X, or only some of them, as in process Y. In either case, the custom applications that support this process are created, operated, and maintained by the ALM process, as shown here. How fast a new business process can be rolled out and how quickly it can be changed are largely dependent on how fast the applications it relies on can be built and updated. Being good at ALM is a prerequisite for being good at creating and changing business processes.

Being good at ALM is a prerequisite for being good at creating and changing business processes.

Getting better at ALM can even affect the build vs. buy decision for new software. If an improved ALM process makes creating new software faster, cheaper, and less risky, an organization might choose to build more often and so get more customized applications that add competitive value. In a world that runs on code, getting better at creating that code is unquestionably a good thing.

UNDERSTANDING ALM AS A BUSINESS PROCESS

ALM supports business processes, and ALM is itself a business process. It's common to equate ALM with the software development lifecycle (SDLC), viewing it as solely focused on creating applications. Yet every deployed application must be monitored and managed, which means that operations are also part of an application's lifecycle. Just as important, business processes change, and so the ability to maintain and modify an application after it's deployed is also central to ALM. The reality is that ALM is more than just SDLC.

Still, it's fair to say that improving the SDLC aspects of ALM will yield the most benefit in a typical organization. Getting really good at, say, running software efficiently might give you a cost advantage, but it's not going to catapult your business forward. Being really good at software development—creating innovative applications faster than your competitors—can enable this kind of business leap. Yet especially in its development aspects, ALM isn't like most other business processes. Software development can't be a repetitive process like, say, employee onboarding or order-to-cash, and it's certainly not as regular and predictable as a manufacturing process. While these more traditional processes must deal with occasional exceptions, the basic flow and even how long the process takes to complete is typically much the same each time it's executed.

Creating software is effectively a form of new product development.

ALM's SDLC aspects aren't like this. Instead, creating software is effectively a form of new product development. The process looks at least somewhat different each time, and how long it takes to complete can vary. To manage this variation, modern approaches to SDLC use multiple iterations, as Figure 2 shows.

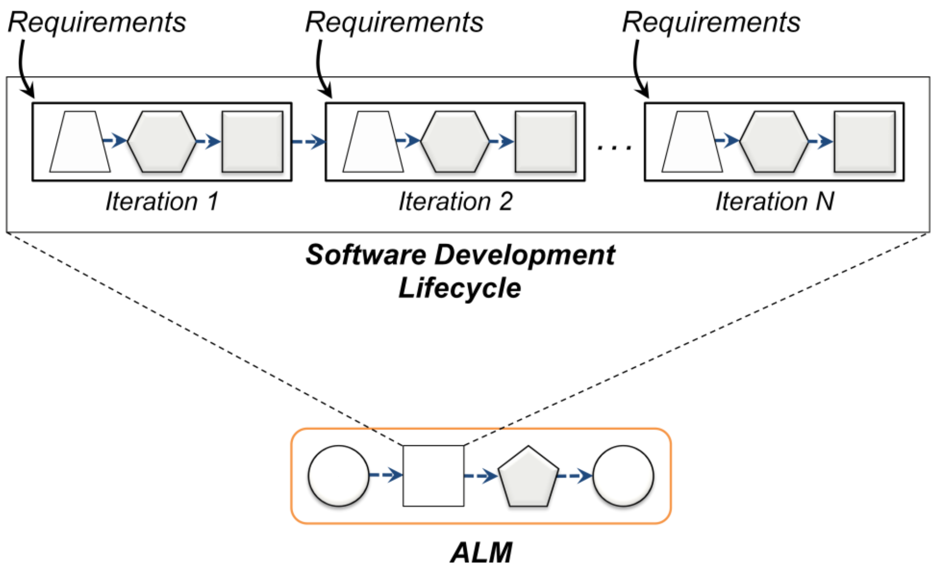

Figure 2: In a modern ALM process, the software development lifecycle is divided into several iterations, with prioritized requirements input to each one.

Rather than rigidly defining all of the requirements up front, then building software that meets these requirements and testing it—the traditional waterfall approach—iterative software development performs each of these steps multiple times. At the start of each iteration, the project's requirements can be changed and re-prioritized, as Figure 2 shows. During each iteration, the development team spends some time on design, some time writing code, and some time testing that code.

While this approach might seem inefficient, it's much better at dealing with the inevitable changes in requirements that crop up during the development process. For example, the people who will use a new application often don't know exactly what they want when development begins. An iterative approach lets the development team show its users how things look at each stage, letting them give feedback that affects the project's direction. Because new requirements are incorporated at the start of each iteration, changes can be made at well-defined points in the process. (Being iterative is also a fundamental characteristic of what are known as *agile* processes, although there’s some disagreement on the precise definition of this term.)

Like other business processes, ALM itself can be supported by software. Just as with any automated process, an organization that creates a first-rate ALM process in one group can use ALM software to help replicate that process quickly across the company. But because ALM is different from many other business processes, the software that supports it doesn’t implement a fixed, unchanging set of steps. Instead, tools for ALM support the fundamentals of the process, such as tracking requirements, maintaining code and documents in a centralized place, and keeping track of project status, e.g., the current number of bugs and the rate at which those bugs are fixed. Just as the right software can make other business processes more effective, the right ALM software can help improve the way an organization creates and uses custom applications.

CONCLUSION

Most business people understand the strategic value of differentiated business processes. Yet the importance of ALM in supporting those processes gets much less attention. In reality, any organization that creates custom software should take the ALM process as seriously as it does any other critical business process. ALM might be even more important, since it creates the underpinnings for the others.

At Moody’s, for example, the business process for rating debt is surely among the most important in the company. Yet this process relies on a custom application, which means it ultimately depends on Moody’s ALM process. Being better at ALM might have helped the company find and fix the bug that led to mis-rating debt before it caused any damage. And Moody’s is hardly alone—nearly every organization has similar stories.

Being better than your competitors at creating and using custom software can bring substantial competitive advantage. Similarly, being worse can put you at a significant disadvantage. If your organization doesn’t see ALM as one of its most important business processes, it’s time to change that view.

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.