AGILEPOINT

DIFFERENCES
between
BPMS and Low Code Process Platforms

TABLE OF CONTENTS

THE BPMS MALAISE

In a recent low-code vendor landscape report, Forrester analysts Clay Richardson and John Rymer identified five distinct types of low-code platforms, one of which is process-focused low-code platforms. If you read between the lines in this and other Forrester reports, as well as analysis coming from the greater process community, a process-focused low-code platform is really just a synonym for today's new breed of BPMS. This fact poses an interesting question: Why the new moniker? Is there just so much baggage with "BPMS" that vendors are looking for a euphemism, or is there really something so different between the two technologies that a reclassification is in order?

1. TECHNOLOGY OR IDEOLOGY

From a technology standpoint, the evolution from BPMS to low-code sometimes referred to as the "appification" of BPMS—has been in play for several years now, so you'd have to look back quite a ways to see a pronounced difference between an old-school BPMS and a state-of-the-art, process-focused low-code platform. Even the old-guard BPMS incumbents are talking business apps these days, and while there are well defined feature sets designed for BPMS scenarios as well as feature sets designed for low code scenarios, the real gap between the two approaches to software development is probably more ideological than technological.

2. THE LONG AND SHORT OF BPMS

The Business Process Management (BPMS) technology class was founded on a business theory—that an entire organization could be viewed as a collection of discreet and interrelated business processes, and, that by systematically improving each process, an organization could continually improve its operational efficiency. BPMS suites, then, were software platforms designed to automate, monitor, measure, and improve processes. They were all about operational efficiency, which, in a broader context, provided management oversight. Increased oversight, in turn, facilitated governance, risk management, and, ultimately, better compliance with internal rules and regulations and external laws.

But there was, and still is, a downside to BPMS—it sounds great in theory but is extremely difficult to implement on an enterprise-wide scale. The problems with BPMS constitute a tangled web, which encompasses all of the following and more:

- Enterprise processes can be extremely complex.
- Processes often span multiple operational groups within an enterprise.
- Many organizations find it nigh unto impossible to get the necessary stakeholders to agree on optimal process methodologies.
- Most processes are, at best, dynamic and, at worst, downright unstable.
- Building such processes, even with declarative, low-code/no-code tool sets, can take months, even years, during which time the process itself evolves, making the finished application dysfunctional out of the box.
- BPMS initiatives require such broad buy-in and enterprise wide commitment—not just to embark on the journey, but to vigilantly maintain efforts, well . . . forever—that they're really hard to get off the ground.

3. WHEN GOOD PROCESSES GO DARK

Jim Sinur, a one-time Gartner BPMS analyst and current industry thought leader, once wrote an article entitled “Beware of the Dark Process.” The piece, which was published on his Gartner blog, has since been taken down, but the gist of it was this: One of the biggest challenges of BPMS occurs when functional processes go dark, that is, when conditions change in such a way that business logic won’t allow workers to complete the task at hand using the instituted automated process. In such instances, workers have no choice but to revert to their old, non-automated habits. Herein is the onset of darkness—management has lost oversight on the process, which oversight was the real reason it implemented BPMS to begin with.

4. BPMS—THE TREADMILL OF IT

Without having industry data at my finger tips, my bet is that most treadmills (in fact, exercise contraptions in general) are purchased sometime during the first week in January, are put into heavy use for a few weeks, but, by Valentine’s day, have been moved into the garage, where they will collect dust until being listed in the local classifieds for pennies on the dollar. The unfortunate part is that a treadmill is really good for anyone with the discipline to actually use it over a sustained period and will deliver the advertized benefits—lower cholesterol and the ability to fit into last year’s wardrobe.

My point, of course, is that BPMs and treadmills have a lot in common—any organization that will make a broad commitment to BPMS will be better off in the long run. Such enterprises will improve operational efficiency; eliminate costly—sometimes catastrophic—human error; and will have an institutionalized framework for systematic growth. But just as treadmills are, at best, boring and, at worst, flat-out painful, automated processes can likewise become mechanized task-masters.

5. DING DONG, THE PROCESS IS DEAD

So why do knowledge workers often develop a strong dislike of process automation in about the same time it takes to get fed up with a treadmill? Consider this: Perhaps the biggest problem with BPMS is that the actors in any given process may end up spending more time and effort to complete their sub routine than it used to take them doing it the old way. (Anyone who’s ever tried to get a bunch of old-school sales reps to use a CRM knows of what I speak.) Automated processes require strictures, protocols, and new stuff to learn. And from the perspective of those in the trenches, “if it ain’t broke, why fix it?”

The answer, of course, comes from the top down: Overall efficiency from automation, in many cases, is achieved only from the viewpoint of senior management, for whom the automated process imposes governance on a sometimes global and, often, undisciplined, workforce. And governance mitigates organizational risk by ensuring compliance with rules, regulations, and laws. Put another way, BPMS may actually drive labor costs up, but the company’s bottom line will, nonetheless, improve via litigation averted, penalties avoided, and blunders headed off at the pass.

Just the same, the dissonance between the C-Suite and the rank and file is often impossible to overcome—when processes go dark, those in the trenches are often in no rush to get them fixed. And just as the Munchkins broke out in song the day the Wicked Witch met her end, so also might staff members launch into their own little river dance once the process is gone for good. It’s this lack of grass-roots support that can erode the resolve of even the most committed senior management team, regardless of what unimaginably large sum it has already sunk into its BPM initiative.

THE ESSENCE OF LOW CODE

Despite the downside, it's not all bad with BPM. The fact is there are some things BPMs are really good at. Composing automated workflows is a given, of course. But BPMs offer much more than simple workflow, the ability to manage updates to automated processes mid-execution to name just one. The fact is, though, that enterprise-class, low-code process platforms can do a lot (if not most) of that "BPM" stuff, which brings us back to the fundamental difference between BPMs and low code process platforms—it's really more ideological than technological.

1. MOVING PAST THE BIG FOUR

As was already mentioned, the big four value props of BPMS are efficiency optimization, governance, risk mitigation, and compliance—all compelling to the COO but often just a pain in the neck to everyone else. So what is interesting to people in the trenches? The quick answer is anything that will make their lives easier. And that's where low code hits its stride.

2. DECONSTRUCTING PROCESS

Again, one of the biggest problems with BPM is the force behind it—operational efficiency, which efficiency may only be apparent to senior executives. For individuals in a complex, cross-functional process, automation may make their jobs worse. By eliminating the ultimate goal, though, of overall efficiency and, instead focusing on what's actually problematic for individuals within a functional group, the dynamics of workflow/business app development change dramatically.

3. MAKING LIVES EASIER

Suppose, for example, that someone in some department realizes that a serious bottleneck could be eliminated by creating a simple trigger app—maybe when someone drags something into a SharePoint list, several other events are performed automatically (accounts could be created, data could be updated, notifications could be sent . . . whatever.)

With low-code, not only could such an app be created without C-suite oversight and without deep analysis into how it affects an uber complex, enterprise-wide process, the app could be created without so much as mentioning it to the IT department. In fact, the person who had the idea to begin with—given a certain level of computer literacy—could actually build the app herself.

Therein lies the universal appeal of low code—lightweight, simple, unobtrusive. Build what you need when you need it.

4. LOW CODE IN A NUTSHELL

In a recent webinar Forrester's Clay Richardson, a leading authority on low code, summed up the benefits of low code as follows:

- **On-the-Fly Requirements Discovery**

  Low code platforms are declarative, model-driven environments, enabling software design to be, for lack of a better analogy, something like a Lego® experience—dragging the necessary pieces and parts onto a canvas and configuring them to have specific characteristics. Because the process is so fast, it's often easier to just begin building an app using trial and error, rather than spend weeks or months defining system requirements. This experimentation-approach to system development often helps teams uncover hidden value in apps.

- **Live Trial Business Ideas at Low or No Cost**

  Because apps are so easy to build, it's possible to stage live trials of new apps in days, hours, or even minutes of conception. Associated costs of development and testing are minimal.

- **Deploy and Scale Apps in Minutes**

  Once deployed and tested, extending access to low code apps, depending on the platform in question, would take hardly any time at all.

- **Generate Mobile Apps from Older Apps**

  Teams are able to build new mobile apps without utilizing traditional mobile-app platforms, such as SWIFT, Xcode, or Android. Rather, using a low-code platform, teams are often able to take existing apps and transform them into mobile apps.

- **Expand Development Resources Cost Effectively**

  While enterprise-class low-code platforms are not for computing neophytes, they can be applied effectively by power users. Furthermore, IT teams have learned that junior software developers or developers without formal computer-science training can easily learn low-code development, and produce sophisticated, cross-functional, hybrid business apps that will run on any type of device in a relatively short amount of time.

THE AGE OF THE CUSTOMER

One of the compelling ideological differences between BPMS and low code is a byproduct of the stage of evolution of the IT industry. BPMS was the province of Operations—intended almost entirely for use within the boundaries of an organization. But as Clay Richardson points out, we are now in the "Age of the Customer." And one of the most compelling characteristics of low-code platforms is that they can be used to create systems of engagement. In other words, low-code apps can be used to interconnect an enterprise with its ecosystem—employees, channel partners, and customers.

AGILEPOINT

DIFFERENCES between BPMS and Low Code Process Platforms

To learn more about
AgilePoint NX, the Responsive Application Platform,
visit www.agilepoint.com.

Learn More | Request Trial | Request Demo