Appian

Selecting a Low-Code Platform

RULES IN CONTEXT TO LOW-CODE

OVERVIEW

Appian is a low-code platform leveraging BPM. Because of this, Appian software straddles a number of analyst quadrants. The three most prolific: low-code, BPM and case management. While Appian has its roots in BPM, client enterprises came to realize that BPM alone was not enough to automate and manage work. Enterprises need access to data independent of process rather than embedded within the process. A seemingly slight nuance yet one with significant impact.

In response to this need, Appian pivoted and became a low-code (data first) platform, leveraging BPM—the only BPM company to successfully make this change.

Appian's data first paradigm is a game changer for enterprises, allowing them to approach work automation and work management in a more efficient way. Starting interactions with data independent of process decreases the number of exceptions that need to be accommodated in a process. Why? Because exceptions are driven by variations in facts—or data—that change after the launch of a process.

Rather than following the traditional BPM approach of modeling out processes to accommodate for every data variation, Appian:

- Provides access to real time data and changes in circumstance so that the user can choose the right process based on the facts—at that time
- This simple paradigm shift allows organizations to model out shorter running processes that are easier to change, test and maintain

Approach work automation and work management in a more efficient way...

Because traditional BPM does not accommodate data independent of process, variations in process data are handled in rules. Processes become overly complex with layers of rules upon rules. Rule layering—using BPM—adds significant overhead to the total cost of ownership (TCO) of the process and adds tremendous challenge to change.

- Consider the amount of testing that is required when changing one rule in the layer. The entire layer and various permeations have to be factored in
- Consider the weight on a BPM application that has to churn through all those rules each time a process is running

For enterprises that require a rules engine, we encourage the use of a modern rules engine rather than leveraging a BPM engine that houses rules capabilities. Appian can integrate and leverage as needed.

The balance of this document illustrates the challenges with the traditional BPM approach to rules and the accommodation of rules by Appian's low-code platform, leveraging BPM.

RULES DEVELOPMENT: TRADITIONAL BPM

The domain of rules leveraged by business is often a mix of many different categories of rules. The purpose for each category varies greatly. For example:

Calculation Rules
(calculation of price or discount)

Assessment Rules
(risk assessment or opportunity assessment)

Logistic Rules
(route determination or alternative delivery determination)

Process Rules
(next task decision or assignment rules)

Decision Rules
(decide on additional approval)

Etc.
(UI rules, Integration Rules, Document Rules, Data Rules, etc.)

Traditional BPM technologies approach the 'rules'-domain generically and purport that they can do it all. This approach often causes rule layering and entanglement, which then leads to increased complexity. Increased complexity affects time to market (T2M) and TCO for several factors: changes, testing, and ongoing maintenance.

To distract from complexity, some BPM providers talk about the simplicity of rules management borrowing a layer cake analogy. The challenge with the analogy is that rules layering and entanglement, at the enterprise level, can cause dependencies between functions and operations within an enterprise that never existed before. Making the de-unification challenging, if not impossible—rendering the software somewhat permanent even when the company wants to decommission.

Rule layering and entanglement is one challenge with this broad brushed approach of traditional BPM technologies, the other is duplicate rules. Leading industry analysts report that one of the most heard complaints regarding BPM is the problem of duplicate rules, including:

- **The ability to put in in new policies.** When you are not aware of the duplicate rules, it is difficult to put in a new policy in your organization, as it is not practiced everywhere
- **Difficulty with upgrades.** During the upgrade, it is discovered that there is much more to do than expected, especially since lots of rules are just customizations and need to be redone during an upgrade

The example below is intended to bring more clarity to the challenge associated with duplicate rules.

Let us start with two rules and evaluate whether they are duplicate:

Rule 1:
If today – start date < 1 year then true else false

Rule 2:
If today – start date < 1 year then true else false

Let us assume that Rule 1 was created on 1/1/2015 as a part of the on-boarding project and is called 'Determine Junior Worker'. One year later on 1/1/2016, HR starts a project and implements a new hire program. The second rule is used as 'Determine New Hire'.

Considerations:

How will BPM determine that these are indeed duplicate rules?
When a second rule is created with exactly the same name, it is easy for BPM—as it is for Appian—to determine the duplicate name, but that doesn't say anything regarding the content of the rule. If a second rule is created with the different name as mentioned above (Determine Junior Worker), BPM needs to somehow determine that the content of the rule is the same.

That can work if the rule is defined exactly the same, but if the second rule was defined as:

- **If current date – employee contract date < 1 year then true else false**

– OR –

- **If today—employee contract date < 1 year then new hire else know employee**

There are numerous possibilities to define a rule that might be a duplicate rule but is simply not recognized as such. BPM does not have the magic to recognize a duplicate rule, nor has any other technology. There might be some basic recognition, but that is different than enforcing re-use.

Do I want BPM to determine this as duplicate rules and force me to reuse it?

The rules as defined in 1) and 2) are *technically* the same, but are they *functionally* the same? BPM is *technically* capable of determining that two rules are the same (even if they appear different as described above), but BPM is not capable of determining if these rules are *functionally* the same. Let's assume that the HR department is using Rule 1 as well. HR is working with its new program and noticed that some support is needed for new employees in the second year as well. They want the rule changed in:

> **Rule 3:**
> If today—start date < 1 year then new else if today—start date < 2 years then new else not new employee

This is not possible because that change would have consequences for the on-boarding application. A common BPM solution for that is called specialization by adding a layer. The first rule is still called, but now when it gives 'false' as a result, it will also call an additional rule determining if the employee is less than 2 years employed. Since the layer is defined for HR, this second rule is not fired when the first rule is called by on-boarding.

This becomes an issue because on-boarding is not aware of HR using an existing on-boarding rule. Because of re-use enforcement when using BPM, HR must re-use the rule. Now, on-boarding has increasingly more work due to additional requirements while the number of on-boarding cases has expanded significantly. On-boarding decides to change the number of years of experience of a Knowledge Worker to 5 years and add an additional category called 'Experienced Worker'.

They want to change Rule 1 to:

- **If today—start date < 2 year then junior else if today—start date < 5 years then not experienced else experienced knowledge worker…**

Since this cannot be done as HR is also using the rule, BPM uses the rules layer cake analogy to solve the problem, by adding a layer for on-boarding handling the specialization. The consequences are significant:

- Increased complexity by adding two new layers; one for HR and one for on-boarding
- A new dependency between HR and on-boarding that was not there before by having them both use Rule 1 in the underlying layer
- Having three rules (in three layers) that could have been handled easily and without dependencies in two rules

The complexity continues to increase. Imagine what happens when UI, Process, Integration, etc. rules are entangled.
The bottom line is that what traditional BPM technologies call specialization, should have actually been called customization. This problem grows exponentially if you 'specialize' out of the box (OOTB) rules and these OOTB rules are changed in a next release.

Ultimately, this becomes unmanageable, resulting in multiple layers with numerous duplicate rules.

RULES DEVELOPMENT: APPIAN

Appian enables the interoperability of the processes and records (data). Interoperability is another time and space saver allowing you to build less because you are constantly leveraging what you have.

When a component is created with Appian (UI, Integration, Expression Rule, Decision Table, etc.), you can always re-use it in any other component, which is the whole idea behind records. In Appian, you are using data/information from other systems (leveraging, not storing) and re-using the record as much as possible, so that you do not have to go to different sources over and over again. This is partly the reason why Appian implementations lead the industry with respect to speed and accuracy.

Appian makes it much easier to avoid rules entanglement, with a better component structure over the different categories of business rules.

Appian is capable of creating and managing business rules through our Decision Tables, Expression Rules and Expression Modeler (e.g. used in the Process Modeler). Appian can easily create UI's for the business to maintain Business Rules Parameters / Inputs, while also easily integrating with specialized Business Rules Engines. Appian's rules, for example Integration Rules, are clearly identified and easily configured through Wizard and can be reused in Business Rules.

Complex rules, such as large risk assessment rules and models, are best handled by specialized rules or modeling tools. Appian can integrate with these tools and can create UI's to manage parameters, inputs and even models. As Appian orchestrates and supplies consistent ways of working across the organization, interaction with services (like business rules and models) becomes seamless and effortless.

CONCLUSION

Business executives strive for innovation while driving efficiency. Highly effective leaders find the balance of leveraging technologies that orchestrate a myriad of rules and capabilities. This view of work leaves complexity out—allowing for a predictable T2M while controlling TCO. Using specialized services and tools avoids entanglement and dependencies between different parts of an organization, while also alleviating technical challenges during upgrades.

A low-code platform is the optimal technology for companies to address and execute ongoing organizational strategies. Business users often have very complex requirements with respect to business rules capabilities. In addition, they want to develop and employ processes independent from IT, striving to make changes at the pace of customer needs.

What is often called 'extensive rules capabilities' can be a mix of requirements: managing specific business parameters (e.g. discount rates), managing specific business rule (e.g. calculate discount), managing task assignment, managing SLA's, managing emails, etc.—all with the daunting requirement of 'without IT involvement or unexpected dependencies'. When not clarified, these look extensive as a whole, but seem reasonable individually.

Appian's low-code platform leverages BPM. This unique view of how to tackle strategic organizational priorities allows for both business and IT to navigate projects at the pace of its customers. They can start to understand that change is not something that is costly, time consuming and risky, but something that can effectively be done together.