BPM: Structured vs. Unstructured

Austin Rosenfeld

When most of us start analysis for a BPM project, we begin by gathering information about the starting and stopping events of the process, the participants, and maybe the reporting needs. We then map out a long sequence of steps that form the process. At the end, we have a BPMN diagram that shows the end-to-end process, accounting for all possible paths, exceptions, events, etc., and we declare the process diagramming "complete." At this point, a common criticism of the BPM paradigm comes into play: real-world processes are executed in a world with little structure, imperfect information, and unforeseen exceptions, so therefore no BPM diagram is ever complete. How, then, can we proceed to work with the IT department and implement a system that will replace the old filing cabinets, emails, and spreadsheets, leaving process participants with no remnants of their previous process execution environment that could handle exceptions outside the context of a rigid set of system-enforced rules? There can be no more "we never encountered this situation before; we need to circumvent the normal flow of things." A task form has been assigned and demands to be completed, with required data fields, before the executing process can move forward down the predefined path. This Article will frame this problem as one of the continuum of process structure and present a technical solution that modern BPM suites support.

At one end of the continuum, we find processes with very loose structure. The hotel concierge handles a stream of independent requests that have a relatively unconstrained domain of content, operating in a predictable manner by following a loosely structured process. His or her process comprises the following steps: accept a request, think of sources to consult, consult those sources, aggregate information, and deliver a response. Because the activities a concierge undertakes are knowledge work, the consultation of sources could involve an unpredictable level of breadth or depth. We would not model this process at the level where we present the user with a finite list of known sources, force the selection of exactly one, and then attempt to automate the search process within those sources (especially before the advent of the all-knowing Google). Instead, we would model this process as loosely structured and leave a lot of discretion to the participants.

At the other end of the continuum, we find highly structured manufacturing processes. In this domain, the inputs and outputs are clear, most process instances follow the exact same path, and we expect to churn through a high volume (producing millions of nearly identical widgets) with very few exceptions. The exceptions we do encounter are fairly predictable in advance: shortage or irregularity of an input, lack of storage capacity for an output, or violation of an environmental assumption (weather, pressure, etc.). We can model the process so that at every step, the user has the option of indicating that one of a prepared list of exceptions occurred and we can implement specific handling logic for each exception.

In the business world, most processes fall somewhere in between the concierge and manufacturing examples, lying on the interior of the structure continuum. Many processes even contain several phases, each of which may be at a different point on the structure continuum and may be more or less likely to be subject to unpredictable exceptions. Many business processes also have multiple levels of structure, such as case management processes, which often allow a finite number of activities to be performed within the context of a process but place few or no constraints on the order of those activities or whether each activity is mandatory in any given instance. The list of activities is structured, but the interactions that the process supports are less so. In BPMN terms, processes of this nature call for ad-hoc tasks, a concept whose very name indicates a lack of structure.

To model a process that has varying degrees of structure, we need a toolkit that supports the nature of the process. BPM and BPMN excel at the more structured aspects, supporting linear flow, predictable events, gateways to make decisions with finite and foreseeable outcomes, and validation to apply structure to data. When process flow devolves into knowledge work or an unforeseen exception occurs, what began as a structured process may suddenly turn into case management. Ideally, the "case" is managed in a standardized way within the system and does not become a support ticket with the IT department to remodel the process.

From a technical standpoint, the design of a BPM system has now gone beyond the flow that a BPMN diagram can express. To deliver value to its users, the system needs to support the pattern of ad-hoc case management instances within standard workflow. Specifically, this functionality should be designed with the following features:

- The context of the executing process is available to the people resolving the exception.
- The possible resolutions to the exception are unconstrained, though the system may make suggestions.
- The solution is recorded as a part of the process audit history and remains visible throughout the remaining life of the process.
- The process can continue normal operations once the exception is resolved; the process does not have to be abandoned or restarted from the beginning.

Ideally, the BPM suite that is used to implement the system will have out-of-the-box support for solving the problem of varying structure in the manner described. The simplest metaphor for this solution is a Facebook-style "all-objects-and-events-are-discussion-threads" approach to presentation. Process participants and observers can follow the history of process execution and collaborate via comments to resolve exceptions, all within the context of the original executing process. People handling the exception can perform knowledge work outside the confines of the system but keep other interested parties and the audit trail updated with comments that can even contain links or references to activities performed in other systems. If your BPM suite does not support that metaphor, then dashboards should be designed to provide similar flexibility, allowing users to start sub-processes that mimic discussion thread functionality.

In conclusion, real-world processes vary in their levels of structure. Even within a given process, the level of structure can vary from phase to phase, and events and exceptions can occur that make the ideal structure for the process significantly less rigid. We should design BPM systems to account for the various points on the structure continuum that a process can occupy, especially since BPM suites support design patterns that solve this problem well.

Author

Austin Rosenfeld is the founder of Macedon Consulting, Inc. Previously, he was a product architect at Appian and ran the BPM consulting practice at Amentra. For more information on how BPM suites can deliver flexible solutions for your organization, contact austin.rosenfeld@macedonconsulting.com

BPTrends Linkedin Discussion Group
We recently created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**