Lombardi

WHITE PAPER

What is the difference between Workflow Engines and BPM Suites?

Phil Gilbert
Chief Technology Engineer
Lombardi Software

May 2005

Table of Contents

Introduction

In the 1990's, workflow vendors created quite a bit of confusion in the market when trying to define workflow and how it could best be utilized. Today, that situation is being replicated in the BPM industry. Ask 10 different vendors to define BPM or BPM suites, and you will likely get 10 variations of the definition, even though all vendors use the same basic terminology to explain it. Organizations need help sorting through this confusion in order to discover why a BPM suite is a different, and more important, application for process improvement.

The Workflow Solutions of the '90's

Let's start with a little background. The workflow solutions introduced in the 90's primarily consisted of an engine and a language. Many solutions also included some type of graphical modeling environment, albeit rudimentary. Only a few included a more robust, UML-based modeler; more likely it was a proprietary modeler.

The modeler generated code, normally in the form of some proprietary workflow language, although there was some convergence around the main standards like WFML or XPDL. Both offer XML-based primitives in which process patterns are used to describe workflows. The engine interpreted the code and executed the workflows. Changes or de-bugging typically occurred in the code, as opposed to inside the modeler.

At execution (run-time) many elements of managing tokens (representing tasks) were built into the engines, like queuing and prioritization. Therefore, the aspects of controlling a workflow's behavior could be programmed more easily using these engines and tools, because the basic notions of what kinds of objects flow through a process — such as tasks, or tokens representing tasks — were built-in.

These engines also looked at which resources were involved and consumed during the execution of these tasks. So concepts like organizations and/or systems were built-in and the engine kept track of things like 'how many tasks are currently in user X's inbox.'

Because these workflow systems primarily assisted the developer in areas where "tasks" were assigned, and "workflow patterns" defined the movement of those tasks, it made sense that most uses of these engines were in areas where other programming activities were minimized. That is, when developing automated solutions for processes where workflow and task handling were the primary requirements, workflow engines offered the highest leverage. For example, in imaging and document management systems, automating processes were not about the business data flowing through the process because it was unstructured to begin with. If you simply attached a pointer to the document or image, the rest was handled by the workflow engine.

In processes where a lot of structured data existed, there was extensive custom coding required to build database tables or object models. It also required a lot of integration if there was structured business data, because the systems-of-record had to be touched by the process application. So while the workflow engine provided benefits, it was less beneficial as a percent of the total code that needed to be written in these situations. Additionally, workflow engines typically made a major flawed assumption in these situations; that they knew about the various resources and that they maintained all states of a process. But when you are interfacing with other systems and those systems perform parts of the process on their own, the state is passed back and forth. Working around this required additional programming.

Furthermore, if the requirements were less focused on workflow and more on tying multiple systems together, it was often much easier to code the application without a workflow engine. Otherwise, the mixed paradigms (and languages) often presented skill-set barriers…not to mention difficulty in the subsequent understanding of a program by new people.

Where Are We By the End of the ’90’s?

- Workflow engines had successfully understood the patterns involved in communication between mobile systems (Petri-nets, pi calculus and other maths). Languages had been formed that could provide high-level access to these patterns.
- Unstructured information could be easily attached to tasks managed by a workflow engine.
- Run-time execution reflected the utilization and consumption of resources, allowing for more advanced understanding of these costs.

But problems had developed:

- These languages were splintered and they were new. Any given solution also required knowledge of other languages (SQL, C++, C#, Java, Perl, HTML, etc). If a language was known, there was some leverage but there was also a shortage of people who knew these languages. It was a specialty, and one that was expensive.
- Workflow engines only addressed the workflow pattern and task parts of any given process solution. The other parts of any solution — user interfaces, system interfaces, the object model and persistence layer for those objects — weren't addressed.

The Perfect Storm

In addition to what was happening in the pure workflow world, there were some other disparate events helping shape the market:

- The Internet, enabled by standards at the lowest levels, was driving a new urgency toward the standardization of system interfaces. As the technical ability to do cost-effective e-commerce occurred, the human factor of all that integration became the largest inhibiting factor. Standards for system interfaces were needed quickly and these arrived first in the form of J2EE and .NET and then more recently with Web services.
- Business people were becoming more energized around process excellence, spurred on by the digital successes of companies like Dell and GE who reduced process cost structures while increasing value. And because the focus was coming from business leaders like Michael Dell and Jack Welch, the business-centric nature of the problem, and the solution, became a driving force. Business people were being measured by the performance of their processes and they wanted to take more ownership in the definition, development, understanding and change of these processes. The end result was a need for increased alignment between the business and IT.
- The first wave of lower and middle-management business people were becoming technology-savvy. In many ways, this trend is continuing to drive the future of enterprise computing as both IT and business take responsibility for the definition, development and change of enterprise applications.

Enter BPM

As a technology category, BPM suites exist to enable companies to improve processes. Period. To do this, many elements are required. A BPM suite must:

- Make it as easy as possible for everyone involved in improving a process to understand the process and to collaborate on its improvement.
- Make it easy as possible to leverage technology in the parts of the process where technology can add value.

To accomplish this, a variety of technologies were used - often in fundamentally different ways than their initial design — so that business people could actively collaborate around process solutions.

- **Workflow engines**, and research into workflow patterns, long ago proved out the various ways in which communications occur between disparate systems. Workflow patterns form the basis for how work travels through a process.
- A **highly graphical modeling notation** like BPMN is an easy metaphor for business people to grasp, as opposed to BPEL, which is just code. In the same way that graphical word processors spread the ability to create electronic documents to the masses, a graphical view of workflows has been chosen as the best way to spread process development to the business and technical people responsible for designing new process-based applications.
- **Round-trip engineering** is a requirement in BPM, so that business programmers and process programmers can continue to be involved alongside technical programmers in the evolution of process solutions after the first version is deployed. In the best BPM Suites, this means that the execution engine directly manipulates the model at run-time. Changes to the process execution behavior should be reflected immediately in the graphical model. This is a major technical advance in the state of workflow/process technology and only a few vendors are capable of this.
- **Standardization of system interfaces** have been adopted allowing a more modular approach. This is true not only in the definition of interfaces, but also in the replacement and modification of them in the context of a drag-and-drop interface. Now

the system interface can be co-mingled with the graphical process elements of the workflow and similarly, human user-interfaces are now embedded into the process drawings, or process models.

- **A presentation layer**, or user interface such as forms, are almost always required in human-facing processes, so most BPM suites offer some way to graphically develop a form. The more advanced systems provide a way for the business user to graphically define the form data that is needed, while allowing the technologist to alter the look and feel. In addition, the business person should be able to continually alter the form, independent of the changes the technologist made. This is also a requirement at the modeling level.

- Because every process application has structured business data flowing through it, BPM suites have built-in the ability to **graphically define business objects** (or process variables), and the persistence of many, if not all, of the business objects defined for the process application is "free."

- Since you are going to use a process-centric method to describe the application, it makes sense that the suite should anticipate data requirements for **process-specific reporting**. Things like timing intervals - or the latencies between process steps — can be easily defined in the graphical model without detailed knowledge of databases or SQL required to persist or access them.

- And, finally, along with SOA (service oriented architectures) providing many Web-based services able to perform parts or, in combination, all of processes that are orchestrated and reported on inside the BPM suite, the other systemic aspect to processes is events. Advanced BPM systems treat human and system inputs and outputs to processes on par with external business events. And, once again, the **definition and correlation of external events** is handled graphically, in the modeling environment.

Conclusion

As you can see, the driving factor behind BPM suites wasn't workflow — any more than it was rapid application development of graphical forms, or systems interfaces. BPM is a response to business requirements. The suite of technologies used in delivering a thoughtful BPM suite includes the best of what came before, including workflow.

But using a workflow engine didn’t (and won’t) give you the benefits that a BPM suite will provide because the packaging of the various technologies is itself an advance. The way you develop an application using a BPM suite is superior to the way in which you build an application using a workflow engine. The reason is because the understanding of the problem has evolved, and therefore the integration of the various technologies required to solve the problem has evolved.

It used to be that spell-checking was a stand-alone application; something you did to a document after it was typed. The spell-checking engine became embedded in the word processor not because its functionality was inherently different, but because the usage of the spell-checker was placed in the context of how and when it was used. Printed documents came out the same, but productivity was enhanced because the spell-check experience was placed in the context of other technology, and the overall usefulness and usability of each technology was enhanced.

BPM is about “how” you build your application more than it is about “what” you are building. The applications being built with BPM suites are no different than what could be built using many other combinations of technology, including the “code and engine” approach. The difference is that the BPM suite offers a better, more scalable approach to building these solutions.

The advance is in how accessible the interfaces are to the developer, whether it’s an IT programmer building Oracle interfaces, the business programmer building user interfaces, or the process programmer designing the workflow. The advance is in how collaborative these people can be, how modular the components of the application are, and the skill sets and prior knowledge required to make changes in the middle of a process.

The advance is in whether using tested enterprise-class technologies like workflow, we can make the same productivity advances in enterprise application development that we’ve made in the last 30 years in personal application development.

About Lombardi Software

Lombardi Software is a leading provider of business process management (BPM) software and services which provide ongoing visibility and control of business processes and increase the speed with which organizations can make critical business decisions. Global 2000 organizations such as Dell, Hasbro, Pfizer, Wells Fargo, Whole Foods, and Universal Music Group leverage Lombardi's proven BPM platform and best practices to improve operating performance across their enterprise. Lombardi is listed as a Leader in the 2004 Forrester Wave and as a Visionary in Gartner Group's Magic Quadrant.

Visit Lombardi at: http://www.lombardisoftware.com.