BPM 2.0

IT | Redux

March 2005

Ismael Ghalimi
CEO Intalio

ghalimi@intalio.com

Six years ago, I wrote the first white paper on BPMS. It was one of the seminal publications that helped define the concept for BPM and start a new industry. The three letter acronym, which we borrowed from musicians, became an instant sensation, successful beyond any expectations we could have had at the time. Too successful some would say.

Today, the BPM moniker is used to describe anything from legacy workflow products to business rule engines, flowchart diagramming tools, Java code generators, or even business process reengineering consultancy services. This confusion, perpetuated by software vendors and industry analysts alike, serves two main purposes: it allows any vendor who can show boxes and arrows in its product to keep selling its gear, while letting any analyst who can compile a list of the aforementioned vendors to sell its luminary services to herds of utterly confused end users.

Customers I talk to are asking for a change. They've tried the first version of BPM, did not find what they were looking for, and are wondering if there is anything else worth trying out. The good news: there is, I call it BPM 2.0—a term originally coined by my good friend Bruce Silver, and it is available now. The bad news: the definition I give for BPM 2.0 is a radical one, it leaves no place to hide, and most vendors won't like it. But guess what? I am more interested in making customers happy than letting other vendors sleep well at night, especially when they happen to be my competitors. So here we go, welcome to BPM 2.0!

BPM 1.0	BPM 2.0
Marketed to Business Analysts	Used by Process Analysts
Starting with a Process Modeling Tool	Starting with a Complete BPMS
Multiple Tools from Multiple Vendors	One Single Tool in Eclipse
Usable by J2EE Experts Only	Loved by ABAP, PHP and VB Folks
BPEL, BPML, WSFL, XLANG, XPDL	BPEL
ARIS, HIM, UML, Proprietary Notations	BPMN
BPEL Editor	BPMN Designer
Writing Code Behind the Boxes	Zero Code
Writing Deployment Descriptor Files	One Click Deploy
Implementing Application Connectors	Generating Web Services on-the-fly
Generating Java Code	Interpreting BPEL Code Natively
Web 1.0 User Interface	Web 2.0 User Interface
Bring your own Rule Engine	Rule Engine Included
Bring your own BAM	Real-Time BAM Included
Ad hoc Process Simulation	Native Process Simulation
Continuous Process Improvement	Dynamic Process Optimization
Closed Source Process Engine	Open Source Process Engine
$250,000 Entry Fee	Get Started Today, Free of Charge

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

Used by Process Analysts

Let's start by debunking the biggest lie about BPM, which is that business analysts could use a BPM tool to model & deploy an executable business process. Even though that might be true for the simplest document-centric workflow processes, such as when a business analyst specifies the reviewing process of issuing press releases on a website, it breaks as soon as the process involves transactions with any kind of back office system, for two main reasons: first, last time I checked, no IT guy is ever going to open a port onto the corporate ERP system for a business analyst to mess with; second, the said business analyst does not want to be the one the CEO calls in the middle of the night if the aforementioned ERP system cannot record new purchase orders. Conclusion: BPM 2.0 is not for non-technical business analysts. Never should have been, never will, and nobody should care. Instead, BPM 2.0 is for process analysts who are articulate enough to talk to business folks, yet technical enough to understand the difference between a do-while loop and a for-each statement. We will not bridge the business-IT divide by empowering business analysts to get rid of IT people. Instead, we'll just let more technical process analysts understand business requirements and implement them directly into the process, while leveraging existing IT systems. Neither top-down nor bottoms-up, it's a middle-out approach, and it's the only one that makes the gap any narrower.

Starting with a Complete BPMS

Because BPM was originally marketed to business analysts, vendors thought that it would be a good idea to start with the only tool business analyst could use, namely some kind of flowchart diagramming tool. Problem is, that's exactly what customers did, but they did nothing else beyond that, for a very simple reason: once a business analyst has diagrammed a process with a tool that does not enforce any rule that would make the process executable, there is no way to make that process executable later. All that work goes to waste and the business analyst feels she's been cheated. If you want to try this BPM 2.0 thing out, my advice is "don't buy a process modeling tool". Instead, get yourself a complete BPMS and start building executable processes from day one. Some would call it Agile Development for business processes. I call it BPM that works.

One Single Tool in Eclipse

Not long ago, going from a process map to deployed code used to take up to seven tools: one for modeling the process at a business level, an other to describe technical details, a third to build connectors to external systems, a fourth to map data in and out, a fifth to specify business rules, a sixth to design workflow user interfaces, and a seventh to deploy all the code on a collection of proprietary runtimes components. They all required different kinds of expertise, ran in different environments, used different languages, lost information when going from one to the other, and made the whole thing so complex than no end-user could actually use them without the outrageously expensive consulting services of a software vendor that got all the pieces through multiple acquisitions rather than building them from scratch with a very clear architecture in mind. BPM 2.0 puts everything you need within one tool, and that tool sits on top of Eclipse. With Oracle and

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

SAP now supporting Eclipse, no other integrated development environment—beside Microsoft's Visual Studio—will matter anymore, so get along with it and demand that your BPM 2.0 tool natively runs in Eclipse.

Loved by ABAP, PHP and VB Folks

BPM products of the first generation required expertise with J2EE, or even worse, proprietary scripting languages, as if the world needed yet an other programming language. As much as I like J2EE for what it gives me as a software vendor, it's a freakishly complex set of specifications that are out of reach for most IT people. Object-oriented programming is extremely powerful, but like it or not, most programmers do not understand it, or at least would rather use something simpler like PHP or Visual Basic. There are 2 million Java programmers out there, and I reckon that only a fraction of them can write EJB components. Even less know how to combine EJB with JMS, Servlets and Message-Driven Beans. Compare that to the 3 million PHP coders and the 8 million VB developers out there, and you'll start getting the picture. BPM 2.0 is not for the J2EE gurus, or at least not limited to them. BPM 2.0 targets process analysts who can read a BPMN diagram, understand the tree-view representation of an XML Schema, and drag-and-drop form widgets onto a canvas. If you think you can do that without too much effort, then BPM 2.0 will work for you.

BPEL

In the early days, there was no standard for executable processes. XLANG was on the drawing board, WSFL did not exist, and the workflow guys had produced nothing more than a set of utterly useless interfaces. Then BPMI.org released the BPML specification, which forced Microsoft and IBM to abandon XLANG and WSFL respectively. For political reasons, Microsoft and IBM decided to write their own specification, rather than adopting BPML, which led to the release of WS-BPEL, a year after the first commercial implementation of the BPML specification was deployed into production. Three years of sterile public discussions followed, until BPMI.org merged with the OMG and finally decided to drop BPML in favor of BPEL. In short, no real standard was available for the last six years, which contributed to slowing down the adoption of BPM. Things are a little bit different today. BPEL has won, BPEL 2.0 is a good enough specification for customers to build mission-critical processes with, and all the big vendors have adopted it. BPM 2.0 works because of BPEL, much like relational databases work because of SQL. Process analysts should not really care about it, for they won't have to write a single line of BPEL code if they pick the right tool, but BPEL is like the DNA of your process, it's the standard that everything else gets built around. So if you're being told that BPEL does not matter, or that the product you're considering buying will support BPEL next year, don't be fooled: you're about to buy a very expensive piece of proprietary software that you will have to get rid of sooner than you think, so don't make the same mistake that early adopters made, and go for the standard, today.

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

BPMN

As for the process execution language, early BPM products sported many different notations, with cute little shapes and fancy colors. Some were very workflow centric, like HIM, others more technically oriented, like UML Activity Diagrams, but most were totally proprietary, incomplete, and incompatible with each other. BPMI.org set out to fix this, but this time around learned from its early mistakes and made sure that IBM was involved as early in the development process as possible. A fine gentleman by the name of Stephen White did his magic and developed BPMN, which quickly established itself as the standard notation for modeling executable business processes. BPMN supports both the orchestration of web service and the execution of human workflow tasks, while enabling the choreography of multiple business processes through the swimlane metaphor. BPM 2.0 works because one can go from BPMN to BPEL without having to write code. BPMN is not perfect and should learn a couple of tricks from HIM, but its support for compensating transactions, unsolicited events, complex loops and multiple swimlanes is what makes it unique, effective and irreplaceable.

BPMN Designer

Early BPM products supporting the BPEL specification offered a BPEL editor as primary development tool, instead of a real business process design tool. The problem with this approach is that BPEL is a very complex specification, especially from a data management standpoint. Furthermore, BPEL's heavy reliance on complex web services specifications requires developers to manually synchronize multiple BPEL and WSDL files in order to deploy an end-to-end process. BPEL editors do not make this exercise much simpler. They also produce process models that do not have a coarse-enough granularity for them to be shown to a business audience. A higher-level notation is needed, it's called BPMN, and BPM 2.0 must take advantage of it for a wide-enough audience of process analysts to get the productivity they need for their BPM projects.

Zero Code

BPMN and BPEL make for an extremely powerful combination because they allow one to go from picture to code without having to actually write the code. Let's face it, a lot of work went into the development of these two specifications, and this work benefited from an unprecedented amount of collective experience that no single vendor could ever match on its own. What that means is that most BPM products that are based on proprietary notations and execution languages actually require the writing of quite a bit of code in order to make processes executable. Double click on the neat-looking boxes and arrows, and code written in Java or proprietary languages will show its face. There is nothing fundamentally wrong about code, but it just so happens that writing and maintaining code is harder and more expensive than writing and maintaining none at all. BPM 2.0 makes it possible to implement the most complex processes without having to write code. The Dutch Government did just that for a process that has a quarter of a million activities, so if it worked for them at such a scale, it should work for many other organizations.

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

One Click Deploy

By their very nature, business processes are prone to change, and most of us got interested by this new BPM thingy because of the promise that it would make change a little bit easier. I can even remember one of the early pure play BPM startups using the tagline “Go ahead! Change!” to emphasize that very point. Well, this is all nice and fancy, but if one has to write multiple deployment descriptor files and configure various web service interfaces to deploy a process, the ability to change the process as you go remains a pipe dream. BPM 2.0 advocates a radical ‘One-Click-Deploy’ approach to solve this problem. Once your process is valid, with all data mappings completed, business rules defined, and workflow parameters set, just click on a button and get the process deployed on your runtime environment, without any additional work. There is absolutely no reason why it should be any more complex than that.

Generating Web Services On-the-Fly

Most BPM solutions are workflow systems in disguise, and as such, they do not really support integration with back office systems. Distributed transactions and reliable messaging are foreign concepts for such tools. For the rest of them, integration with enterprise applications has been forever tainted by what could be considered as the biggest scam in the history of enterprise software, the idea that you need custom connectors to integrate with enterprise applications such as PeopleSoft or SAP. In the late nineties, EAI vendors made a fortune selling connectors: you want to enter a purchase order into this version of SAP R/3? Buy connector X, for a cool $25,000 per CPU. You want to get the list of employees from that version of SAP R/3? Buy connector Y, for an other $25,000 per CPU. In reality, one can write a generic connector for all versions of SAP R/3, back to SAP R/3 3.1i, that exposes all 200,000 BAPIs, IDOCs and RFCs as web services, on-the-fly, for both standard and custom transactions, without writing a single line of code. The same can be done for Oracle, PeopleSoft, Siebel, and most enterprise applications out there. If it’s possible, BPM 2.0 should take advantage of it, and unless one takes some masochistic pleasure in developing one-time connectors for APIs that might be obsolete the next day, nobody should have to write custom connectors anymore.

Interpreting BPEL Code Natively

Early implementations of the BPEL and BPML specifications relied on Java code generation: you write the code in BPEL, and a code generator automatically translate it into a set of Java classes that are deployed on a Java Virtual Machine or a J2EE Application Server. Good news: it’s a relatively easy way for a software vendors to get into the BPEL game. Bad news: it does not really work. Much like Oracle’s database does not generate C code to execute a given SQL query, a good BPMS should not have to generate Java code to execute a BPEL process. Java code generation is bad because it makes the deployment of processes more complex than it should be, it creates discontinuity from the process semantic that makes debugging and monitoring an order of magnitude more difficult than with native interpretation, and ultimately, it slows everything down. Things get even worse when such implementations rely on the EJB component model for the persistence of process data, especially when Entity Beans with

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

Container-Managed Persistence are being used. For it to work at a large scale—the aforementioned Dutch Government is running 250 million concurrent process instances that take up to five years to complete on a 4-CPU box—BPM 2.0 must natively interpret the BPEL 2.0 code, ideally through just-in-time compilation into process bytecode that closely maps to the Pi-Calculus semantic, and forgo the use of any EJB component for persistence, relying instead on straight database connectivity. If you need performance and scalability, you should go for such a model.

Web 2.0 User Interface

BPM 1.0 solutions relied on Web 1.0 user interfaces, namely standard email clients and web portals serving task lists and forms produced using plain HTML. BPM 2.0 must take advantage of Web 2.0 and Office 2.0 technologies, such as AJAX for dynamic tasks lists and complex forms supporting client-side data validation, RSS feeds for process events and user task lists, weblogs and wikis for process documentation, web-based calendars for task scheduling, and REST APIs for supporting the most creative mashups. Somehow, BPM has yet to be perceived as a 'cool' technology, and Web 2.0 might be all that is needed to move the needle enough to get a more mainstream audience excited by it.

Rule Engine Included

Up until now, BPM solutions would fall into two camps: you either had a glorified rule engine presented as a generic BPM solution, or you had a generic BPM solution that failed to support the execution of complex business rules natively. As a result, most customers who deployed a BPM solution of the later kind had to look for a rule engine from a third-party vendor, even though they did not really need a full-fledged rule engine to being with. BPM 2.0 makes the rule engine a requirement, so that it can be leveraged by the BPM vendor itself in places where it makes sense, such as decision branching, message routing, late-stage service binding, or contextual user interfaces. As James Taylor puts it, it is no longer OK for the BPM vendors to have nothing in the way of a rule engine—they must either build something comparable or, more likely, OEM something from one of the business rules leaders. BPM 2.0 makes the Business Rule Management System (BRMS) part of the BPMS, so that only one platform has to be managed and the lifecycle of rule-driven processes can be streamlined. To a large extent, the BPMS becomes the killer application that rule engine vendors have been waiting for up until now.

Real-Time BAM Included

Much like the data management industry had separate vendors for data processing (database vendors) and data analytics (business intelligence vendors), the business process management industry has featured separate vendors for BPM and BAM. It might take time for companies to actually merge, but products won't wait for this to happen before merging on their own. With BPM 2.0, BAM is part of the overall solution, day one, instead of being an optional afterthought. You need BAM, because doing BPM without it is like driving a car with your eyes wide shut. BAM is one of those godsends that BPM makes possible, and I cannot think of a reason why anyone should not take advantage of it today.

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

Native Process Simulation

With BPM solution of the first generation, process simulation was supported by ad hoc process simulators that were based on primitive finite state machine simulating the execution of processes to be deployed on a separate runtime. Good news: such an ad hoc simulator is relatively straightforward to implement. Bad news: it does not reflect the true nature of the target runtime environment, cannot accurately simulate load testing, and has no visibility onto process data, which represents a good half of the process' semantics when using process execution languages such as BPEL. While appropriate for addressing the needs of business analysts who have no interest in the actual execution of the processes they model, such simulators are totally useless to the people who own the overall process lifecycle. Learning from this experience, BPM 2.0 advocates a different approach for simulation, whereby the process engine is used as process simulator. According to such an approach, simulated processes are stubbed out from external systems, which are emulated by the process engine itself. In order to simulate a process, the process development environment automatically deploys a collection of process instances and randomly generate seed variables in order to support simulation models such as Monte Carlo. The process engine executes the set of simulated process instances and lets the BAM infrastructure aggregate the results to be displayed back to the process analyst, who gets access to business-level key performance indicators, as well as system level performance metrics. This approach ensures that the semantics of the simulated process remains 100% accurate with respect to the semantics of the process to be deployed. Furthermore, by combining business indicators with system metrics, it guarantees that business and IT get equal representation in the evaluation of processes to be deployed within a mission-critical production environment. At the end of the day, ad hoc process simulators are nothing more than cute toys for business analysts, while native process simulators are accurate metrology instruments that BPM practitioners can safely rely on.

Dynamic Process Optimization

Adepts of the Business Process Reengineering school promoted the concept of continuous process improvement, and early BPM vendors made sure to support this methodology with their products. Problem was, changing the process once deployed into production turned out to be more difficult than most had initially expected, especially when software code had to be re-written for changes to be applied. Furthermore, if making a change to a process meant going through the entire process lifecycle all over again, from modeling to simulation and deployment, most ideas for process improvements remained just that, ideas. BPM 2.0 marks a departure from the concept of continuous process improvement, and promotes a more dynamic process optimization model, whereby key process elements can be optimized on the fly, without having to re-deploy the entire process. Through the use of native BPEL interpretation, reusable process interfaces, externalized business rules, late-stage binding, and instance-level exception handling, running process instances can be optimized in real time, without requiring advanced technical skills. Such an approach allows the Dutch government to make regular changes to processes that can take up to five years to complete. If anyone had any doubt that BPM could be used to support long-running transactions, such doubts should be put to rest by now.

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.

IT | Redux

by Ismael Ghalimi

March 2005

Open Source Process Engine

Your process engine will quickly become the most critical piece of your IT infrastructure. As such, you need the best insurance policy money can buy for it, and if you can get it for free, even better. This is the main reason why your BPMS should be architected around an Open Source process engine. Whatever should happen to your BPM vendor of choice, a community will remain to support you. The web, which one could have called Internet 2.0, was literally built on top of the Open Source Apache web server, which still runs a cool 67% of all web servers connected to the Internet today. The same will be true for BPM 2.0, and the leading Open Source BPEL server will also become the leading BPEL server.

Get Started Today, Free of Charge

Last but not least: BPM is too good of a technology for it to be kept out of reach from most of its potential users because of high acquisition costs. Until now, a fully featured enterprise-class BPMS would cost north of $250,000. If you wanted to add support for business rules, BAM and a couple of enterprise applications, it would cost you close to $500,000. I tend to think that such a high price tag is the largest single contributor to the slow adoption rate for BPM that we witnessed over the past six years. BPM 2.0 will change this by making enterprise-class BPMS products free of charge when deployed through certain configurations. Hybrid business models blending Open Source and commercial software will ensure that a handful of vendors enjoy the success they need for supporting customers over the long run. In the end, customers, system integrators and software vendors alike will all benefit from the upcoming explosion of the BPM market. Things are starting to get exciting, so don't wait any longer and join the party today!

Ismael Ghalimi is a passionate entrepreneur and fervent industry observer, founder and CEO of Intalio, initiator of BPMI.org, and author of IT|Redux. Ismael is a professional scuba diver, student pilot, fanatic American V-Twin enthusiast, and avid LinkedIn networker. Ismael can be contacted at ismael@itredux.com.

Copyright © 2005 Ismael Ghalimi. All Rights Reserved.