On the Maturity of Open Source BPM Systems

**Petia Wohed[1], Arthur H.M. ter Hofstede[2], Nick Russell[3],
Birger Andersson[1], and Wil M.P. van der Aalst[2;3]**

1 Stockholm University/The Royal Institute of Technology {petia,ba}@dsv.su.se
2 Queensland University of Technology, Australia a.terhofstede@qut.edu.au
3 Eindhoven University of Technology {N.C.Russell,w.m.p.v.d.aalst}@tue.nl

Introduction

Two major trends in the area of information systems development are (1) the growing interest in Workflow and Business Process Management (BPM) systems and (2) the increasing acceptance and use of open source software. This development raises the following questions:

1. *What are the prominent representatives of current open source Workflow and BPM systems?*
2. *What is the maturity level of these systems?*

Seeking answers to these questions, we have performed a study into the maturity of open source BPM systems. First, an overview of the existing open source systems was made, and three systems – jBPM, OpenWFE, and Enhydra Shark – were selected for further analysis. Then a detailed analysis of the selected systems was performed. The results from this analysis were documented, and the developers were invited to comment. Based on the feedback from the developers, the results were revisited and the study finalized [5]. In this article, we summarize the study and reflect on the area of open source BPM systems.

The study

There is a large number of open source workflow and BPM offerings. For instance, two separate enumerations (see [3] and [4]) contain more than 30 references each. After an extensive literature review, we selected three from these systems, which we state are good representatives for our study. These are jBPM, OpenWFE, and Enhydra Shark, and they are also the systems pointed out by Paul Harmon in his article from 31st of July 2007 [1] as leading in the area.

A well-established way of determining the functionality of a BPM offering is by conducting a patterns-based analysis. The Workflow Patterns Initiative[1] started in the late nineties led to a collection of patterns that, among others, have been used to assess the *control-flow* specification, *data* manipulation, and *resource* handling capabilities of a process language. Since the creation of the original control-flow patterns, more than twenty systems, (proposed) standards, and languages have been subjected to a patterns-based evaluation. These analyses provide a detailed comparative insight, which may serve as the basis for tool selection or adaptations.[2]

We used the workflow patterns framework not only because it is a powerful analysis tool, but also because many evaluations of proprietary tools have already been performed with it. This enables comparison, and here we will also compare the results from our study with the results from

---

[1] www.workflowpatterns.com
[2] It is worthwhile pointing out that a patterns-based analysis is concerned with suitability rather than expressive power. The latter is a formal notion that deals with the issue of whether a language can express certain requirements at all, while the former notion deals with the issue of how easy it is to capture such requirements. Clearly, the former notion is the more important one as, e.g., Java can capture all computing requirements, but nonetheless a big number of languages – e.g., jBPM (Java for Business Process Management) – are continuously created raising the expectations from these languages with respect to their suitability for capturing a diversity of business process scenarios!

corresponding studies performed on Staffware[3], IBM's WebSphere MQ, and Oracle BPEL Process Manager[4].
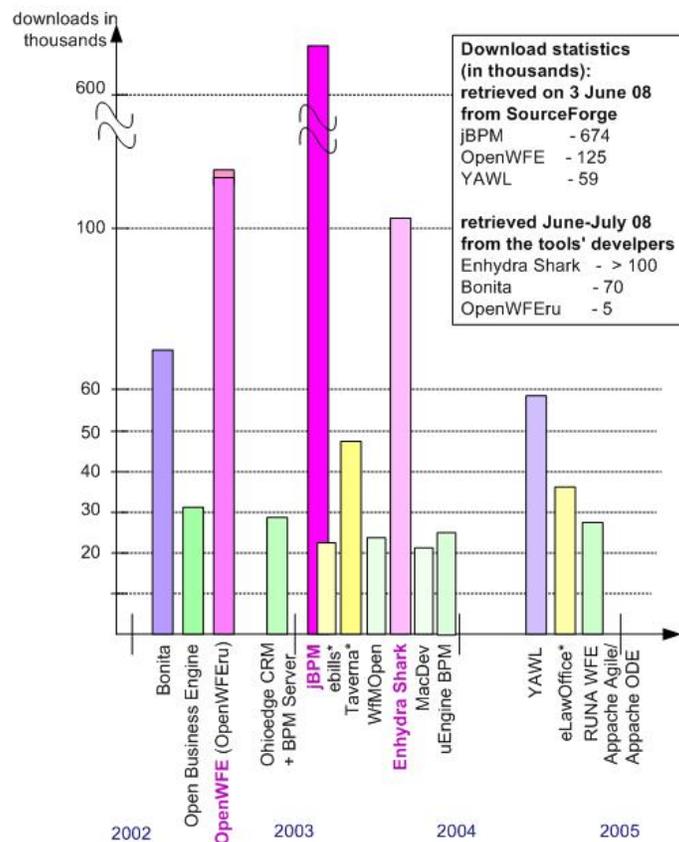

Figure 1. Open Source BPMSs - history

The open source systems

Figure 1 shows a snapshot (from 3rd of July 2008) of the most downloaded workflow and BPM systems distributed through a code repository. The systems in the figure are ordered historically with respect to month and year of their registration. The code repositories searched for this snapshot were SourceForge, RubyForge, ObjectWebForge, Tigris.org, BountySource, BerliOS, JavaForge, and GNU Savannah[5]. The selection criteria for a system to be captured in the figure were (1) at least one of the keywords "workflow" and "BPM" appears (at the time point of the search) in the project name or description; (2) the project is about a Workflow or BPM system (and not, for instance, about a document management system utilizing a BPM solution); (3) the project is active, i.e., at least one file upload has been made during the last two years (i.e., since June 2006); and (4) the number of downloads is larger than 20 thousand. Most of the systems in Figure 1 are distributed through SourceForge where download statistics are publicly available.

---

[3] Staffware was since bought by Tibco and used as a base for Tibco's i-Process Suite. Around January 2008 Tibco's i-Process Suite migrated to BPMN as a graphical notation (supporting a sub-set of BPMN symbols) and XPDL 2.0 as a process definition language.

[4] As can be read from its name, Oracle BPEL Process Manager is based on BPEL4WS, i.e., Business Process Execution Language for Web Services language, which was defined in 2002 as a merge of IBM's WSFL and Microsoft's XLANG languages.

[5] Hence, systems distributed directly from vendors' sites, such as Intalio's BPM system are not shown in this figure.

For Enhydra Shark, Bonita, and OpenWFEru (i.e., OpenWFE's successor), distributed through ObjectWebForge or RubyForge, such statistic was retrieved from the developers of the systems.
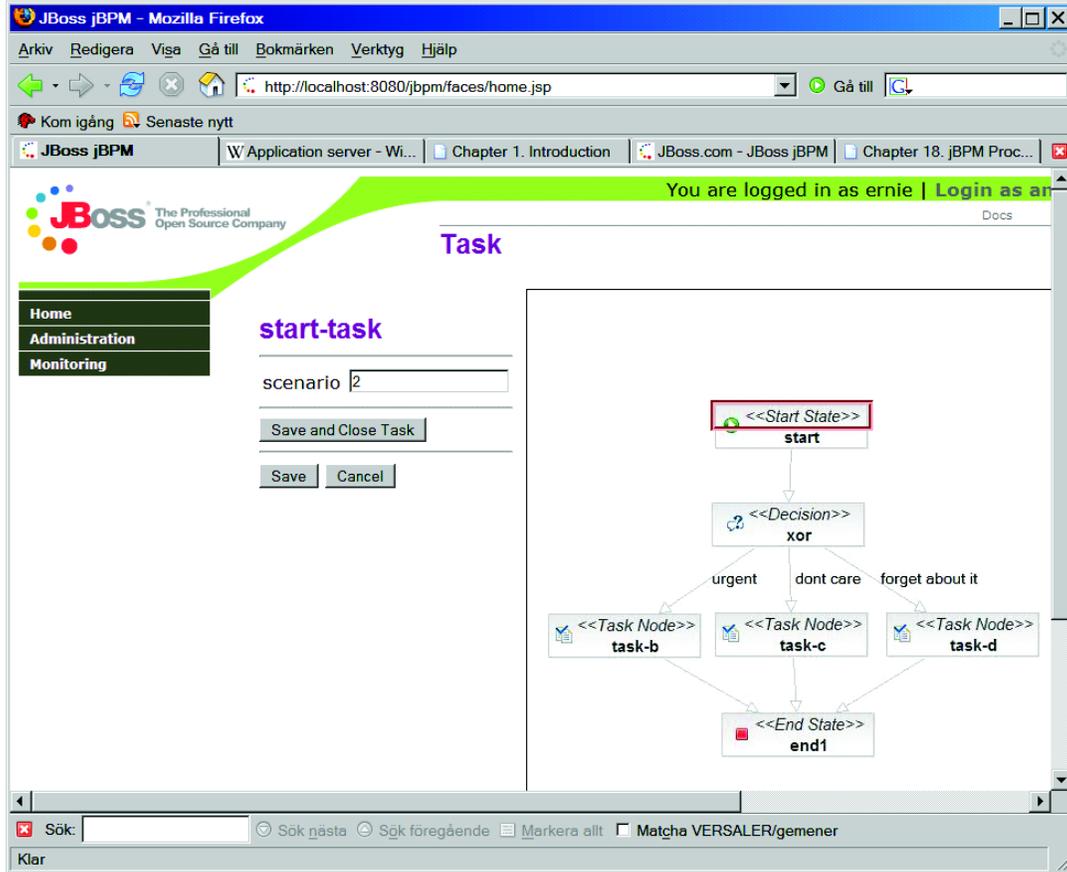

Figure 2. JBoss JBPM - execute task window

It should be noted that three of the systems in the figure, Taverna, ebill, and eLowOffice (marked with an asterisk), are domain specific[6], hence, fall outside our focus on generic workflow and process management systems. The remaining of the systems in the figure can clearly be divided into three distinct groups: systems with more than 100 thousand downloads, i.e., jBPM, OpenWFE, and Enhydra shark; systems with around 60-70 thousand downloads, i.e., Bonita and YAWL; and systems with around 20 and 30 thousand downloads. As we are closely involved with the development of YAWL, YAWL was kept out of the study, as was Bonita, which was the only other representative in this group. Our attention therefore turned to the three systems from the first group: jBPM, OpenWFE, and Enhydra Shark.

jBPM[7] is a Java based workflow management system provided by JBoss and available on SourceForge. Version 3.1.4 of it was evaluated. The offering is distributed with an LGPL license. The components JBoss jBPM graphical process designer, JBoss JBPM core component, and JBoss JBPM console web application were used during our work. Figure 2 shows a screenshot from JBoss JBPM console and the interface of the tool during the execution of a task.

OpenWFE[8] is a workflow management system, written in Java. Version 1.7.3 was studied through the components: OpenWFE Engine, OpenWFE Webclient, and an independent XML

---

[6] Taverna is a scientific workflow management system, ebills is a system for supporting electronic payment workflows, and eLow is tailored to support low firms.

[7] www.jboss.com/products/jbpm

[8] www.openwfe.org

    3

editor. Figure 3 shows a screenshot from the webclient and the execution of a task. The development of OpenWFE has now migrated from Java to Ruby with the new implementation referred to as OpenWFEru. OpenWFE is distributed through SourceForge, while OpenWFEru is distributed through RubyForge. The Java version is distributed with an LGPL license, and the Ruby version is distributed with a BSD license.

Enhydra Shark[9] is a Java workflow engine offering from Together Teamlösungen, distributed through ObjectWebForge. We worked with the following components: Shark TWS-community-2.0-1 (TWS - Together Workflow Server) and the editor JaWE TWE-community-2.2-1 (TWE - Together Workflow Editor). The evaluation was done through the TWS Admin client. Together Teamlösungen provides also a closed-source version of the tool, i.e., Together Workflow Editor Professional and Together Workflow Server Professional (which have not been included in our study). The open source version of the offering is distributed with an LGPL license. Figure 4 shows a screenshot from the editor and exemplifies the modeling notation used in the tool.

Figure 3. OpenWFE - execute task window

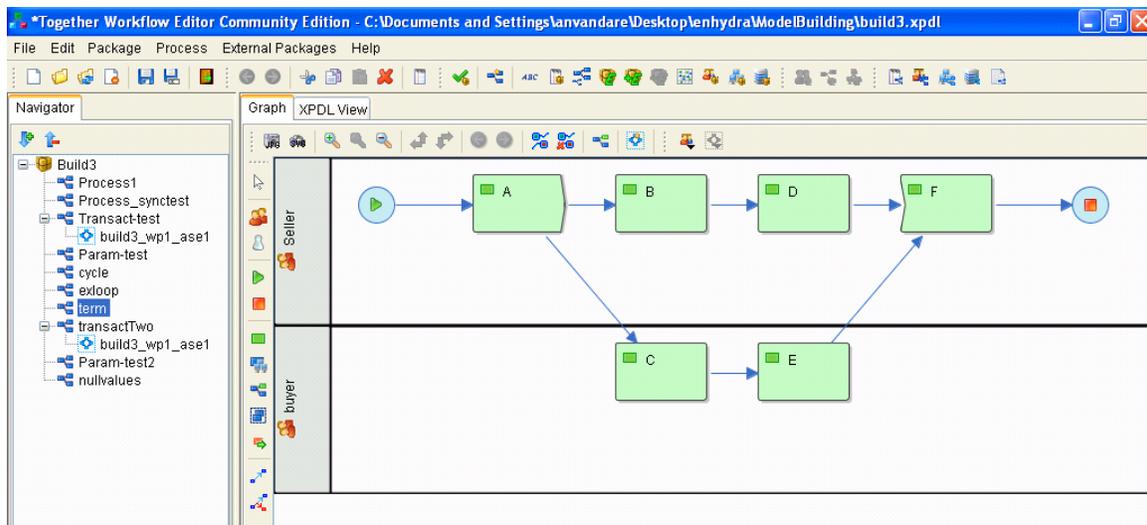[9] www.enhydra.org/workflow/shark/index.html

Figure 4. Enhydra Shark - Together Workflow Editor (TWE)

The results

The study was carried out as follows. Solutions for every pattern, 126 all together, were sought in each of the tools. If a solution for a pattern was identified, then this solution was deployed and tested, and additional feedback from the developers was considered. Tables 1, 2, and 3 summarize the results for the control-flow, data, and resource perspectives, correspondingly.

The *control-flow patterns* systemize different scenarios for how activities are ordered in a process. The *data patterns* capture different mechanisms for dealing with and distributing data relevant for workflow and business process management systems. The *resource patterns* outline different scenarios of how work is distributed among the resources/actors in a process.

A '+' in the tables indicates that direct support for a pattern is present, a '+/–' indicates that direct support is present but in a limited way, and a '–' indicates that there is no support for a pattern. Overall, one can conclude that the range of constructs supported by the three systems is somewhat limited, although OpenWFE tends to offer a considerably broader range of features than jBPM and Enhydra Shark.

From a control-flow standpoint, jBPM and Enhydra Shark support a relatively limited set of control-flow operators, offering little support for patterns other than those related to basic control-flow (see Table 1). OpenWFE offers broader support for variants of the partial join and discriminator constructs and also for controlled task concurrency (i.e., multiple instance tasks).

From a data perspective, all three offerings support a limited range of data element bindings and rely heavily on case-level data elements. While simplistic, the data passing strategies employed in the three systems are reasonably effective and include consideration of important issues, such as inline data manipulation, as data elements are being passed. There are, however, limited capabilities for handling external data interaction without programmatic extensions. It is noticeable that jBPM relies heavily on the use of Java for coding data-related issues and thus its overall level of direct support for the data patterns is relatively low. OpenWFE provides a more comprehensive support in this perspective and supports a wider (but still limited) range of features. In Enhydra Shark, external data communication is meant to be supported through predefined Tool Agents; however, some of these (e.g., the MailToolAgent) did not work in the evaluated open-source version of the offering.

A notable shortcoming in all three offerings is the minimalistic support for the data perspective to influence other aspects of workflow operation, especially the control-flow perspective – e.g., no (or limited) postconditions, trigger support, and limited data-based routing support. Another concern is the shortcomings when dealing with data manipulation activities occurring in parallel.

(I.e., data are lost either because parallel updates on it are ignored, or because some of the updates are given overwriting priority). When parallel work items are operating on the same data, jBPM copies back the corresponding values in the order of work items' completion (overwriting the values of earlier completed instances); OpenWFE copies back the corresponding values according to the specified strategy, i.e., First, Last, etc. (overwriting data when the Last strategy is applied and ignoring new data when the First strategy is applied); and Enhydra Shark does not copy back the variable values (hence, also losing data).

Basic Control–flow	A	B	C	1	2	3	Termination	A	B	C	1	2	3
1.Sequence	+	+	+	+	+	+	11.Implicit Termination	+	+	+	+	+	+
2.Parallel Split	+	+	+	+	+	+	43.Explicit Termination	-	-	-	-	-	-
3.Synchronization	+	+	+	+	+	+	Multiple Instances						
4. Exclusive Choice	+	+	+	+	+	+	12.MI without Synchronization	+	-	+	+	+	+
5.Simple Merge	+	+	+	+	+	+	13.MI with a pri. Design Time Knl	+	-	+	-	+	-
Advanced Synchronization							14.MI with a pri. Runtime Knl.	+	-	+	-	+	-
6.Multiple Choice	-	+	+	-	+/-	+	15.MI without a pri. Runtime Knl.	-	-	+/-	-	-	-
7.Str Synchronizing Merge	-	+	+	-	-	-	27.Complete MI Activity	-	-	-	-	-	-
8.Multiple Merge	-	-	-	+	-	-	34.Static Partial Join for MI	-	-	-	-	+	-
9.Structured Discriminator	-	-	-	-	+	-	35.Static Canc. Partial Join for MI	-	-	-	-	+	-
28.Blocking Discriminator	-	-	-	-	-	-	36.Dynamic Partial Join for MI	-	-	-	-	-	-
29.Cancelling Discriminator	-	-	-	-	+	-	State-Based						
30.Structured Partial Join	-	-	-	-	+	-	16.Deferred Choice	-	-	+	+	-	-
31.Blocking Partial Join	-	-	-	-	-	-	39.Critical Section	-	-	+	-	-	-
32.Cancelling Partial Join	-	-	-	-	+	-	17.Interleaved Parallel Routing	-	-	-	-	+/-	-
33.Generalized AND-Join	-	-	-	+	-	-	40.Interleaved Routing	-	-	-	-	+	-
37.Local Sync. Merge	-	+	+	-	+/-	-	18.Milestone	-	-	+/-	-	-	-
38.General Sync. Merge	-	-	-	-	-	-	Cancellation						
41.Thread Merge	-	-	+/-	+/-	-	-	19.Cancel Activity	+	-	+/-	+	-	-
42.Thread Split	-	-	+/-	+/-	-	-	20.Cancel Case	-	-	+	-	+/-	+
Iteration							25.Cancel Region	-	-	+/-	-	-	-
10.Arbitrary Cycles	+	-	-	+	+	+	26.Cancel MI Activity	+	-	+	-	-	-
21.Structured Loop	-	+	+	-	+	-	Trigger						
22.Recursion	+	+	-	-	+	+	23.Transient Trigger	+	-	-	+	+	-
							24.Persistent Trigger	-	-	+	-	-	-

Table 1. Support for the Control-flow Patterns in A–Staffware 10, B–WebSphere MQ 3.4, C–Oracle BPEL PM 10.1.2, 1–JBOSS jBPM 3.1.4, 2–OpenWFE 1.7.3, and 3–Enhydra Shark 2.0

Data Visibility	A	B	C	1	2	3	Data Interaction-External (cont.)	A	B	C	1	2	3
1. Task Data	-	+/-	+/-	+/-	-	+/-	21. Env. to Case–Push	+/-	+/-	-	-	-	-
2. Block Data	+	+	-	-	+	+	22. Case to Env.–Pull	-	-	-	-	-	-
3. Scope Data	-	-	+	-	+/-	-	23. Workflow to Env.–Push	-	+/-	-	-	-	-
4. MI Data	+/-	+	+/-	-	+	+	24. Env. to Process–Pull	+/-	-	-	-	-	-
5. Case Data	+/-	+	+	+	+	+	25. Env. to Process–Push	-	+/-	-	-	-	-
6. Folder Data	-	-	-	-	-	-	26. Process to Env.–Pull	+	+	-	-	-	-
7. Global Data	+	+	+	-	+	-	Data Transfer						
8. Environment Data	+	+/-	+	+/-	+	+/-	27. by Value–Incoming	-	+	+	-	-	+/-
Data Interaction-Internal							28. by Value–Outgoing	-	+	+	-	-	+/-
9. Task to Task	+	+	+	+	+	+	29. Copy In/Copy Out	-	-	+	+	+	+
10. Block to Subpr. Dec.	+	+	-	-	+	+	30. by Reference–Unlocked	+	-	+	-	-	-
11. Subpr. Dec. to Block	+	+	-	-	+	+	31. by Reference–Locked	-	-	-	-	+	-
12. to MI Task	-	-	+/-	-	+	-	32. Data Transf.–Input	+/-	-	-	+	+	+
13. from MI Task	-	-	+/-	-	-	-	33. Data Transf.–Output	+/-	-	-	+	+	+
14. Case to Case	+/-	+/-	-	+/-	+/-	+/-	Data-based Routing						
Data Interaction-External							34. Task Precond.–Data Exist.	+	-	-	-	+	-
15. Task to Env.–Push	+	+/-	+	+/-	+	+	35. Task Precond.–Data Value	+	-	+	-	+	-
16. Env. to Task–Pull	+	+/-	+	+/-	+	+	36. Task Postcond.–Data Exist.	+/-	+	-	-	-	-
17. Env. to Task–Push	+/-	+/-	+	-	-	-	37. Task Postcond.–Data Val.	+/-	+	-	-	-	+/-
18. Task to Env.–Pull	+/-	+/-	+	-	-	-	38. Event-based Task Trigger	+	+/-	+	-	-	-
19. Case to Env.–Push	-	-	-	-	-	-	39. Data-based Task Trigger	-	-	-	-	-	-
20. Env. to Case–Pull	-	-	-	-	-	-	40. Data-based Routing	+/-	+	+	+/-	+/-	+

Table 2. Support for the Data Patterns in A–Staffware 9, B–WebSphere MQ 3.4, C–Oracle BPEL PM 10.1.2, 1–JBOSS jBPM 3.1.4, 2–OpenWFE 1.7.3, and 3–Enhydra Shark 2.0

Creation Patterns	A	B	C	1	2	3	Pull Patterns, continuation	A	B	C	1	2	3
1.Direct Allocation	+	+	+	+	-	+	24. Sys.-Determ. WL Mng.	+	-	-	-	-	-
2. Role-Based Allocation	+	+	+	-	+	+	25. Rrs.-Determ. WL Mng.	+	+	+	-	-	-
3. Deferred Allocation	+	+	+	+	+	+	26. Selection Autonomy	+	+	+	+	+	+
4. Authorization	-	-	-	-	-	-	Detour Patterns						
5. Separation of Duties	-	+	-	-	-	-	27. Delegation	+	+	+	-	-	-
6. Case Handling	-	-	+	-	-	-	28. Escalation	+	+	+	-	+	-
7. Retain Familiar	-	+	+	+	-	-	29. Deallocation	-	-	+	-	+	+
8. Capability-based Alloc.	-	-	-	-	-	-	30. Stateful Reallocation	+/-	+	+	-	+	-
9. History-based Alloc.	-	-	+/-	-	-	-	31. Stateless Reallocation	-	-	-	-	-	-
10. Organizational Alloc.	+/-	+	+/-	-	-	-	32. Suspension/Resumption	+/-	+/-	+	+	-	-
11. Automatic Execution	+	-	+	+	+	+	33. Skip	-	+	+	-	-	-
Push Patterns							34. Redo	-	-	-	-	+/-	-
12. Distr. by Offer-Single Rsr.	-	-	+	-	-	+	35. Pre-Do	-	-	-	-	-	-
13. Distr. by Offer-Multiple Rsr.	+	+	+	-	+	+	Auto-start Patterns						
14. Distr. by Alloc.-Single Rsr.	+	+	+	+	-	-	36. Comm. on Creation	-	-	-	-	-	-
15. Random Allocation	-	-	+/-	-	-	-	37. Comm. on Allocation	-	+	-	-	-	+
16. Round Robin Alloc.	-	-	+/-	-	-	-	38. Piled Execution	-	-	-	-	-	-
17. Shortest Queue	-	-	+/-	-	-	-	39. Chained Execution	-	-	-	-	-	-
18. Early Distribution	-	-	-	-	-	-	Visibility Patterns						
19. Distribution on Enablement	+	+	+	+	+	+	40. Config. Unalloc. WI Vis.	-	-	-	-	+/-	-
20. Late Distribution	-	-	-	-	-	-	41. Config. Alloc. WI Vis.	-	-	-	-	+/-	-
Pull Patterns							Multiple Resource Patterns						
21. Rsr.-Init. Allocation	-	-	-	-	-	-	42. Simultaneous Execution	+	+	+	-	-	-
22. Rrs.-Init. Exec.-Alloc. WI	+	+	+	+	-	-	43. Additional Resources	-	-	+	-	-	-
23. Rsr.-Init. Exec.-Offered WI	+	+	+	-	+	+							

Table 3. Support for the Resource Patterns in A–Staffware 9, B–WebSphere MQ 3.4, C–Oracle BPEL PM 10.1.2, 1–JBOSS jBPM 3.1.4, 2–OpenWFE 1.7.3, and 3–Enhydra Shark 2.0

For the resource perspective and the results listed in Table 3, it can be concluded that only simple notions of work distribution are supported and typically only one paradigm exists for work item routing in each offering. There is no support for any form of work distribution based on organizational criteria, resource capabilities, or execution history. All three offerings provide relatively simple facilities for work item management; e.g., (for two of them) there is no ability to configure work lists at resource or system level, no notion of concurrent work item execution, and no facilities for optimizing a work item throughput (e.g., automated work item commencement, chained execution). One area where OpenWFE demonstrates noticeably better facilities is in terms of the range of detour patterns (e.g., deallocation, reallocation) that it supports.

Comparison with closed source systems

We used the workflow patterns framework not only because it is a powerful analysis tool, but also because many pattern-based evaluations of proprietary tools were already performed. Therefore, our results can be compared with the results from similar studies. In Tables 1- 3 the results from the evaluations of three proprietary offerings taken from [2] are shown: Staffware[10], IBM's WebSphere MQ, and Oracle BPEL Process Manager[11].

When it comes to comparing the state-of-the-art in open source workflow systems to that in proprietary systems, the results in Tables 1- 3 show that none of the offerings stands out as being clearly superior to the others, although it can be argued that Oracle BPEL PM demonstrates a

---

[10] Staffware was since bought by Tibco and used as a base for Tibco's i-Process Suite. Around January 2008 Tibco's i-Process Suite migrated to BPMN as a graphical notation (supporting a sub-set of BPMN symbols) and XPDL 2.0 as a process definition language.

[11] As can be read from its name, Oracle BPEL Process Manager is based on BPEL4WS, i.e. Business Process Execution Language for Web Services language, which was defined in 2002 as a merge of IBM's WSFL and Microsoft's XLANG languages

marginally wider range of features, while Enhydra Shark and jBPM clearly lag behind in terms of overall patterns support. Oracle BPEL PM and OpenWFE tend to demonstrate broader pattern support in their corresponding tool classes (i.e., open-source vs. proprietary), especially in the control-flow perspective. Moreover, it can also be observed that the proprietary tools are generally better equipped in the resource perspective and better able to support interaction with the external environment, whereas the open-source systems essentially rely on their users having programming experience (e.g., Java) to achieve the required integration with other systems. In the data perspective, jBPM, relying heavily on Java coding, lags clearly behind the other offerings.

Overall, one can conclude that the open source systems are geared more towards developers than towards business analysts. If one is proficient with Java, jBPM may be a good choice. If not, choosing jBPM is less advisable. Similarly, while OpenWFE has a powerful language for workflow specification in terms of its support for the workflow patterns, we postulate that non-programmers will find it difficult to understand. Finally, Enhydra Shark's minimalistic support for the workflow patterns may require complicated work-arounds for capturing nontrivial business scenarios.

Finally, we should note that there are a couple of limitations with our study. First, while the patterns framework is a powerful analysis tool, it does not cover all the aspects of interest for a workflow or business process management system. Aspects such as performance capabilities, version management, model analysis capabilities, and administration facilities are some examples of aspects not covered by the framework. Secondly, the way in which the studied open source offerings were selected provides a limitation. While systematic, it leaves out a third category of offerings, namely, offerings which were initially developed as proprietary products and lately released as open source, but are still available only through their vendors' web sites, e.g., Intalio BPMS.

Reflections

The use of open source software for the deployment of BPM solutions opens up a series of new opportunities for both open source developers and the user community more generally. Unlike other application domains, the BPM area is relatively new and not polarized by the efforts of any particular mainstream software developer. Indeed, as recognized by industry analysts, it is a sector that is in a significant state of flux as new entrants identify and capitalize on opportunities in different phases of the BPM lifecycle. The dynamic nature of the field brings with it both challenges and opportunities for software developers and users alike. In this section we consider these issues in more depth.

Implications for Open Source Developers

The rapidly evolving nature of the BPM domain makes it a fertile area for open source developers to identify and exploit innovative niche opportunities. As a relatively immature software domain, it remains an area that is not yet dominated by any particular vendor or standards initiative. Hence, there is significant opportunity for the rapid, grassroots approach to software development championed by the open source community. In order for open source developers to succeed in the field, however, there are both new challenges and considerations of which they need to be mindful.

A new technology landscape for BPM is emerging.   As the BPM marketplace continues to mature, it is becoming increasingly evident that it is composed of a series of distinct application areas based around individual segments of the process lifecycle. Areas such as business process modeling, process enactment, process mining, and business activity monitoring (among many other possibilities), all constitute valid focus areas for BPM tool suppliers; however, the inherent complexity of each of these individual areas means that it is extremely unlikely that a single solution – commercial or open source – is able to adequately meet the functional demands of more than one BPM application area. As it is generally the case that open source initiatives are

resource constrained, developers should carefully consider the scope of the solution that they intend to deliver.

Effective business solutions are made up of many parts. As process thinking becomes more pervasive, process technology is being increasingly applied to more complex business problems. These solutions cross organizational and technology boundaries in ways that have not been previously considered, and it is now increasingly likely that one single process technology is not able to fulfill the needs of all aspects of a business process. With this in mind, not only should open source developers focus on delivering a comprehensive solution in a specific BPM application area, they should consider their offering as just one part of the overall BPM landscape and actively examine how it can effectively interoperate with similarly positioned tools (open or closed source) in other related BPM application areas.

Standards are a double-edged sword. Traditionally, IT standards have been proposed as a basis for achieving increased interoperability between distinct technological offerings. However, in recent years, many standards initiatives seem increasingly to hamper the entry of new solutions providers. The standards are both informally defined and subject to ongoing revision, meaning developers continually need to update their software (sometimes markedly) in order to continue to be compliant as the standard(s) evolve. Moreover, the fundamental intention of many standards (i.e., a single accepted way of working in a given technical domain) is not even pursued by the parties behind them. As an example, Microsoft is a major contributor to the BPEL standard, but only partially supports it in their BizTalk offering, and in other initiatives, such as the Windows Workflow Foundation (part of Windows Vista), actively promotes alternate ways of implementing some BPEL constructs. While in BPEL the <flow> operator can be used to model graph-based process models, this construct is not available in the Workflow Foundation. Instead there is a graph-based construct that only allows for sequential behavior and a (structured) parallel composition operator. Since the large software companies do not take their own standardization efforts seriously but still expect other parties to support them, it seems that smaller competitors (e.g., open source initiatives) can only be slowed down in their development efforts by paying too much attention to such standards. With this in mind, while open source developers need to be mindful of industry standards, they should not view them as a panacea that will necessarily increase the usage and relevance of their offering.

Open source offers a variety of new distribution possibilities. One of the significant changes engendered by the open source movement is recognition that the value of a software offering lies not simply in having open access to the associated source code, but more specifically in possessing the knowledge associated with its implementation and operation. This shift in value recognition opens up a variety of new ways in which open source offerings can be distributed and deployed. Open source developers should be mindful of these alternatives and recognize that it is no longer necessary to deliver software directly to the end user, but that alternate distribution opportunities exist, such as embedding their offering in other products (possibly offered by other vendors), licensing its deployment, and distributing it as a companion to other suitably aligned products.

Implications for Open Source Users

On the surface, open source BPM solutions seem to offer the end user the answer to their BPM prayers – an innovative, rapidly evolving, and cost-effective source of potential solutions to the wide range of BPM-related issues that they are currently facing. However, despite the promising outlook offered by the open source BPM community, there are some salient considerations for potential end-users of these offerings.

Open source BPM is technology-transfer not product acquisition. The acquisition model for open source technology is inherently different from that of commercial software offerings. The end deliverable is not carefully packaged and well documented software marketed to potential users by savvy pre-sales staff and delivered by an experienced technical team. It's more likely to be source code downloaded from a website. The initiative for open source procurement comes from

the user not the vendor. This has a significant implication: The end-user cannot expect simply to "use" the software as they would with any other commercial tool; rather, they need to "own" the solution at both a conceptual and a technical level. This entails a far greater investment in both upfront and ongoing education in the product's overall construction and operation than would be the case with commercial off-the-shelf (COTS) software. Moreover, end-users should anticipate that the deployment of an open source offering would involve some degree of customization to their specific needs and that they would most likely need to commission or undertake this customization themselves.

Community engagement is vital to long-term success. Successful deployments of open source offerings are characterized by users who have successfully engaged with the software developers and continue to do so on a long-term basis. This is vital as users should not only focus on addressing short-term support issues but also need to provide broader feedback to developers on the realities of using the software and the opportunities for its longer term development. One of the great successes of open source development practices has been the establishment of successful models and associated technology support for distributed software development. These attributes offer users the ability to interact with and support development efforts in ways not previously available to them. They also provide the opportunity for the user to actually guide the long term development plans for the software in a way that is generally not possible with COTS software.

Total cost of ownership can be difficult to establish. By the time that commercial software offerings are taken to market, their overall value has been recognized by the vendor and quantified in terms of their overall purchase price. Moreover, the experiences of other customers give the new user some insight into the total cost of ownership (TCO) for the offering, i.e., the overall cost of acquiring and utilizing the software over a given period of time. The same degree of certainty is difficult to establish for open source offerings. Although upfront purchase costs are likely to be minimal (if there are any at all), there is likely to be a significant education and technology expense as new users seek to understand and adapt the software to their specific needs. The extent of these costs can be difficult to quantify. In addition, actual operational costs can also be difficult to determine, given that it may not be possible to learn from the experiences of other users and that there may not be others who are using the software in the same way.

Early adopters need to ensure they mitigate their risk. In many cases, open source BPM efforts are in the early stages of their overall development lifecycle. While this has the potential advantage that users are able to acquire innovative BPM software much earlier than might ordinarily be the case, it carries with it the concomitant risk that the software may deviate from its original stated objectives or that it may just not meet them at all. Moreover, given the rapid evolution of many open source initiatives, it is possible that these issues might arise over a much shorter timeframe than might normally be the case with COTS software. Potential open source users need to build these considerations into their acquisition plans for such software. Furthermore, they also need to recognize the potential for the situation to arise where, within their own deployment, they choose to deviate from the release sequence adopted by the open source developer.

Licensing considerations need careful attention. One of the salient features of open source software is that many development initiatives in turn rely on other open source software products. As well as introducing some risk into the correct operation of a given offering, along the lines discussed in the previous point, there is also a more subtle issue with regard to the licensing of any associated products. When a user deploys a software offering, in many cases they are actually deploying several, each of which has its own set of licensing conditions. While these may not be especially restrictive if the acquired software is purely used internally within an organization, there can be significant implications if the software or any associated modifications to it need to be distributed on a wider basis.

References

1. Paul Harmon. Exploring BPMS with Free or Open Source Products. BPTrends, available at www.bptrends.com/publicationfiles/advisor200707311\%2Epdf, July 31 2007. last accessed 27 Sep 07.
2. Workflow Patterns Initiative. Workflow Patterns - homepage. available at www.workflowpatterns.com. last accessed 27 Sep 07.
3. Java-Source.net. Open Source Workflow Engines in Java. available at enumerationsatjava-source.net/open-source/workflow-engines. last accessed 06 Jun 08.
4. Manageability. Open Source Workflow Engines Written in Java. available at www.manageability.org/blog/stuff/workflow_in_java. last accessed 06 Jun 08.
5. P. Wohed, B. Andersson, A.H.M. ter Hofstede, N.C. Russell, and W.M.P. van der Aalst. Patterns-based Evaluation of Open Source BPM Systems: The Cases of jBPM, OpenWFE, and Enhydra Shark. BPM Center Report BPM-07-12, BPMcenter.org, 2007.