Open Source Workflow: A Viable Direction for BPM?
Extended Abstract

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

[1] Stockholm University/KTH, Stockholm, Sweden
{petia,ba}@dsv.su.se
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
{n.c.russell,w.m.p.v.d.aalst}@tue.nl
[3] Queensland University of Technology, Brisbane, Australia
a.terhofstede@qut.edu.au

With the growing interest in open source software in general and business process management and workflow systems in particular, it is worthwhile investigating the state of open source workflow management. The plethora of these offerings (recent surveys such as [4,6], each contain more than 30 such systems) triggers the following two obvious questions: (1) how do these systems compare to each other; and (2) how do they compare to their commercial counterparts. To answer these questions we have undertaken a detailed analysis of three of the most widely used open source workflow management systems [1]: jBPM[1], OpenWFE[2], and Enhydra Shark[3]. Another obvious candidate would have been the open-source workflow management system YAWL (www.yawlfoundation.org). However, given the authors' close involvement in the development of YAWL, we did not include it in our evaluation.

This analysis was based on the *workflow patterns* framework [2]. This framework provides a collection of generic constructs which recur in a workflow context. It is divided into control-flow, data, and resource patterns based on the process perspectives outlined in [3]. A patterns-based analysis is guided by explicit evaluation criteria which are identified for each pattern. It aims to investigate the ability of a workflow system to support each of the patterns that have been identified and is based on the premise that each pattern describes a feature that it is *desirable* to support in a business process context. Hence, the workflow patterns framework is not concerned with expressive power, but rather with *suitability* (see e.g. [5]).

We choose to use the workflow patterns as the basis for our investigation because it is a well established framework that is widely used for WFMS evaluations (as evidenced by the numerous references to it). There are already a substantial number of evaluations of contemporary offerings based on the patterns and they provide an effective means of comparing the capabilities of differing systems on a neutral basis. For the purposes of this analysis, the results from some of these earlier evaluations (i.e., Staffware, WebSphere MQ and Oracle BPEL PM) are added to the results from the analysis of open source systems summarized here.

[1] www.jboss.com/products/jbpm
[2] www.openwfe.org
[3] www.enhydra.org/workflow

Z. Bellahsène and M. Léonard (Eds.): CAiSE 2008, LNCS 5074, pp. 583–586, 2008.

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

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	–	–	+	–	–	–

The investigation was undertaken as follows. Solutions for each of the 126 patterns were sought in each of the tools evaluated. Where successfully identified, they were deployed and tested. The initial results were summarised and each of the system vendors/developers was invited to provide feedback on their accuracy. On the basis of these responses, a final set of results were agreed upon and they were comprehensively documented in the form of a technical report [7]. Tables 1- 3 summarise the main findings.

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). OpenWFE offers broader support for variants of the partial join and discriminator constructs and also for controlled task concurrency (i.e. multiple instance tasks).

For the data perspective, all three offerings support a limited range of data element bindings and rely heavily on case-level data elements. However, whilst simplistic, the data passing strategies employed in all three systems are reasonably effective and include consideration of important issues such as inline data manipulation when data elements are being passed. There are limited capabilities for handling external data interaction without utilising programmatic extensions. Another area of concern relates to shortcomings when dealing with parallelism of data manipulation (i.e. data is lost either because parallel updates on it are ignored, or because some of the updates are overwritten).

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

Data Visibility	A	B	C	1	2	3
1. Task Data	–	+/–	+/–	+/–	–	+/–
2. Block Data	+	+	–	–	+	+
3. Scope Data	–	–	+	–	+/–	–
4. MI Data	+/–	+	+/–	–	+	+
5. Case Data	+/–	+	+	+	+	+
6. Folder Data	–	–	–	–	–	–
7. Global Data	+	+	+	–	+	–
8. Environment Data	+	+/–	+	+/–	+	+/–
Data Interaction-Internal						
9. Task to Task	+	+	+	+	+	+
10. Block to Subpr. Dec.	+	+	–	–	+	+
11. Subpr. Dec. to Block	+	+	–	–	+	+
12. to MI Task	–	–	+/–	–	+	–
13. from MI Task	–	–	+/–	–	–	–
14. Case to Case	+/–	+/–	–	+/–	+/–	+/–
Data Interaction-External						
15. Task to Env.–Push	+	+/–	+	+/–	+	+
16. Env. to Task–Pull	+	+/–	+	+/–	+	+
17. Env. to Task–Push	+/–	+/–	+	–	–	–
18. Task to Env.–Pull	+/–	+/–	+	–	–	–
19. Case to Env.–Push	–	–	–	–	–	–
20. Env. to Case–Pull	–	–	–	–	–	–

Data Interaction-External (cont.)	A	B	C	1	2	3
21. Env. to Case–Push	+/–	+/–	–	–	–	–
22. Case to Env.–Pull	–	–	–	–	–	–
23. Workflow to Env.–Push	–	+/–	–	–	–	–
24. Env. to Process–Pull	+/–	–	–	–	–	–
25. Env. to Process–Push	–	+/–	–	–	–	–
26. Process to Env.–Pull	+	+	–	–	–	–
Data Transfer						
27. by Value–Incoming	–	+	+	–	–	+/–
28. by Value–Outgoing	–	+	+	–	–	+/–
29. Copy In/Copy Out	–	–	+	+	+	+
30. by Reference–Unlocked	+	–	+	–	–	–
31. by Reference–Locked	–	–	–	–	+	–
32. Data Transf.–Input	+/–	–	–	+	+	+
33. Data Transf.–Output	+/–	–	–	+	+	+
Data-based Routing						
34. Task Precond.–Data Exist.	+	–	–	–	+	–
35. Task Precond.–Data Value	+	–	+	–	+	–
36. Task Postcond.–Data Exist.	+/–	+	–	–	–	–
37. Task Postcond.–Data Val.	+/–	+	–	–	–	+/–
38. Event-based Task Trigger	+	+/–	+	–	–	–
39. Data-based Task Trigger	–	–	–	–	–	–
40. Data-based Routing	+/–	+	+	+/–	+/–	+

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

Creation Patterns	A	B	C	1	2	3
1. Direct Allocation	+	+	+	+	–	+
2. Role-Based Allocation	+	+	+	–	+	+
3. Deferred Allocation	+	+	+	+	+	+
4. Authorization	–	–	–	–	–	–
5. Separation of Duties	–	+	–	–	–	–
6. Case Handling	–	–	+	–	–	–
7. Retain Familiar	–	+	+	+	–	–
8. Capability-based Alloc.	–	–	+	–	–	–
9. History-based Alloc.	–	–	+/–	–	–	–
10. Organizational Alloc.	+/–	+	+/–	–	–	–
11. Automatic Execution	+	–	+	+	+	+
Push Patterns						
12. Distr. by Offer-Single Rsr.	–	–	+	–	–	+
13. Distr. by Offer-Multiple Rsr.	+	+	+	–	+	+
14. Distr. by Alloc.-Single Rsr.	+	+	+	+	–	–
15. Random Allocation	–	–	+/–	–	–	–
16. Round Robin Alloc.	–	–	+/–	–	–	–
17. Shortest Queue	–	–	+/–	–	–	–
18. Early Distribution	–	–	–	–	–	–
19. Distribution on Enablement	+	+	+	+	+	+
20. Late Distribution	–	–	–	–	–	–
Pull Patterns						
21. Rsr.-Init. Allocation	–	–	–	–	–	–
22. Rrs.-Init. Exec.-Alloc. WI	+	+	+	+	–	–
23. Rsr.-Init. Exec.-Offered WI	+	+	+	–	+	+

Pull Patterns, continuation	A	B	C	1	2	3
24. Sys.-Determ. WL Mng.	+	–	–	–	–	–
25. Rrs.-Determ. WL Mng.	+	+	+	–	–	–
26. Selection Autonomy	+	+	+	+	+	+
Detour Patterns						
27. Delegation	+	+	+	–	–	–
28. Escalation	+	+	+	–	+	–
29. Deallocation	–	–	+	–	+	+
30. Stateful Reallocation	+/–	+	+	–	+	–
31. Stateless Reallocation	–	–	–	–	–	–
32. Suspension/Resumption	+/–	+/–	+	+	–	–
33. Skip	–	+	+	–	–	–
34. Redo	–	–	–	–	+/–	–
35. Pre-Do	–	–	–	–	–	–
Auto-start Patterns						
36. Comm. on Creation	–	–	–	–	–	–
37. Comm. on Allocation	–	+	–	–	–	+
38. Piled Execution	–	–	–	–	–	–
39. Chained Execution	–	–	–	–	–	–
Visibility Patterns						
40. Config. Unalloc. WI Vis.	–	–	–	–	+/–	–
41. Config. Alloc. WI Vis.	–	–	–	–	+/–	–
Multiple Resource Patterns						
42. Simultaneous Execution	+	+	+	–	–	–
43. Additional Resources	–	–	+	–	–	–

For the resource perspective, 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 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.

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 marginally wider range of features, whilst 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 clearly lags 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, although if not, choosing jBPM is less advisable. Similarly, whilst OpenWFE has a powerful language for workflow specification in terms of its support for the workflow patterns, we postulate that it will be difficult to understand by non-programmers. Finally, Endydra Shark's minimalistic support for the workflow patterns may require complicated work arounds for capturing nontrivial business scenarios.

Acknowledgement. We would like to thank John Mettraux for prompt and helpful responses through the OpenWFE help forum and Saša Bojanic for constructive and valuable feedback on Enhydra Shark.

References

1. Harmon, P.: Exploring BPMS with Free or Open Source Products. BPTrends 5(14) (July 2007)
2. Workflow Patterns Initiative. Workflow Patterns - homepage. www.workflowpatterns.com, (last accessed September 27, 2007)
3. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture and Implementation. Thomson Computer Press, London, UK (1996)
4. Java-source.net. Open Source Workflow Engines in Java. java-source.net/open-source/workflow-engines, (last accessed September 27, 2007)
5. Kiepuszewski, B.: Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows. PhD thesis, Queensland University of Technology, Brisbane, Australia (2003) http://www.workflowpatterns.com/documentation/documents/phd_bartek.pdf
6. Manageability. Open Source Workflow Engines Written in Java. www.manageability.org/blog/stuff/workflow_in_java, (last accessed September 27, 2007)
7. Wohed, P., Andersson, B., ter Hofstede, A.H.M., Russell, N.C., van der Aalst, W.M.P.: 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)