Embedding Workflow Engines

Dragos A. Manolescu Santanu Paul

1 Introduction

Software developers have observed that certain parts of their applications are more change-prone than others. For example, for product-centric applications such as those that determine premiums for insurance policies or rates for calling plans, the most dynamic elements are the business rules that govern how products and services are priced and promoted. Likewise, for process-centric applications that handle mortgage loans or manage collaborative, team-oriented projects, the most fluid aspect is the flow of information between humans, systems, and organizations.

An increasingly important goal of software design lies in isolating the highly dynamic elements of software applications, and empowering non-technical, business users with tools to manage them. Rule engines have emerged as a technology that permits business experts to control and maintain pricing-centric applications by manipulating business rules, without any knowledge of the underlying source code. Similarly, workflow engines empower business experts to manage and adapt process-centric applications by simply manipulating process flow diagrams, without touching the software that manages and executes the processes.

Process-oriented applications are partially ordered sequences of steps that execute over time. They are best implemented using a design principle called *flow independence* [1]. Each step in a flow independent application encodes specialized business logic, and is executed by a process actor (i.e., human, application, or an organization). The sequence of steps defines the process flow, and is subject to continual change and tuning by domain experts. Flow independent applications manifest a clear separation of concerns between business domain experts that program and maintain process flows at the macro-level, and software developers that program and maintain each sequence step at the micro-level. This separation of concerns yields significantly greater responsiveness to changing business requirements; the high-level application logic or the "macro-program" can be re-wired by business domain experts using graphical process management tools without depending on software developers. At the same time, each individual step or "micro-program" can be modeled as a context-free, reusable service, because they have little or no awareness of the process flows that invoke them. The astute reader will note the similarity between flow independent design and service-oriented architectures that are currently taking hold in the world of Web Services.

This article targets developers who are either considering embedding a workflow engine, or should be. It provides a narrative on how Openpages, a software vendor that develops and markets business applications for compliance and document management, selected a workflow engine for

its Openpages 4.x server. We cover issues such as identifying the workflow functionality necessary to implement the process flows, assessing the degree of architectural alignment with the workflow engine, evaluating integration efforts, and prioritizing a wide array of other factors. While we draw heavily from our experience evaluating a dozen commercial workflow engines, narrowing it down to three and finally selecting one, we focus on the process of selecting the right workflow engine for the problem at hand rather than provide a comparative survey. This is an important distinction because the selected candidate is a function of the business drivers and cannot be determined outside the business context.

2 Sidebar: A Brief History of Workflow

Since the late 1970s several generations of workflow systems have implemented a wide range of process-intensive applications, from office automation through scientific experiments. The first generation of such systems were hard to use beyond the particular applications for which they were built. The second generation used a master-slave paradigm. A department or enterprise-class workflow server orchestrated hundreds or thousands of simultaneous process flows, invoking actors such as humans and applications to perform the right steps in the right order. In other words, the workflow server was the active master entity that invoked passive slave entities—applications and humans—to perform their respective roles within a business process. Over the last few years however, the model has begun to morph in a different direction. Increasingly, workflow functionality is being perceived as a *component* that can be embedded within other business applications and frameworks [2]. The business application becomes the active entity that delegates the management of process-oriented business logic to an embedded workflow engine. This is the master-slave paradigm in reverse, with the application as the master entity that controls when and how the workflow engine is marshalled. This emerging model has clear benefits. Developers building complex applications can retain control over all aspects of functionality, and yet delegate to business domain experts the maintenance of those select elements of business logic that are fluid and involve subject matter expertise.

3 Embedded Workflow: Benefits and Risks

Making an informed decision about whether to embed a workflow engine or not requires performing a return on investment (ROI) analysis. This analysis involves evaluating the benefits of embedded workflow against its risks.

For an embedding application, a workflow engine provides several benefits:

Empowers Domain Experts Domain experts have direct access to the process representation and can manipulate and adjust it. This translates into saving time and eliminating potential misunderstandings between business analysts and programmers. For example, a Property and

Casualty expert can change the policy underwriting process without relying on application programmers.

Ensures Flow Independence Changes in the business process typically involve workflow changes only. Applications that use embedded workflow adapt better to changing business requirements. For example, a wireless carrier can entice customers to renew their yearly plans by offering the first month free without changing how the billing system rates individual calls.

Accelerates Time-to-market at a Lower Cost An embedded workflow engine provides features that otherwise would be too expensive in terms of time and money to implement on a per-application basis. For example, a Laboratory Information Management System built around a workflow engine can leverage the workflow history to comply with the Health Insurance Portability and Accountability Act.

At the same time, embedded workflow engines pose certain risks:

Metaphor Mismatch The workflow engine may be inappropriate for the embedding application. Certain workflow engines aim at enforcing high volume, repetitive processes that are rigid in nature (the "claims processing" metaphor). Others aim at enabling collaborative processes that allow for a significant degree of flow adjustment during execution (the "routing slip" metaphor). Selecting a rigid enforcement engine for a collaborative application (or vice versa) could lead to an implementation failure.

Lack of Control One of the obvious concerns of using an embedded workflow engine is the lack of sufficient control over its features. Unless the source code for the workflow component and the expertise for changing them are available, tailoring workflow features requires either programmable extension points such as documented interfaces, abstract classes or pluggable components (admittedly a tall order for most workflow engines we know), or a great deal of willingness from the workflow vendor to be adaptable.

Customization Costs Initially a workflow engine provides faster time-to-market for the embedding application. However, as new application requirements emerge, the cost of tailoring the workflow engine accordingly is hard to estimate. Additionally, with enhancement requests from several clients, the workflow engine vendor can become a bottleneck. The problem escalates if the architectural assumptions underlying the application and the workflow engine are not in alignment [3].

To summarize, embedding a workflow engine has its rewards as well as risks. On the positive side, the workflow engine can accelerate time-to-market and provide rich workflow functionality at a low cost. On the negative side, the workflow engine can limit the degrees of freedom for the application developer, and lead to costly implementation failures if the requirements are not well understood and analyzed in advance. However, unless the workflow needs of the embedding application are truly esoteric, the benefits of embedding a workflow engine usually outweigh the risks.

4 A Case Study

The case study involves the Openpages 4.x Server (henceforth called OP4), a J2EE framework that serves as a foundation for the rapid development and deployment of business applications ranging from marketing compliance, financial reporting, corporate governance, and brand management. The vision of OP4 is to provide a rich platform of services such as document management, multi-channel publishing, workflow management, and collaboration, so that these can be readily combined to solve a wide class of business problems that are product- and process-centric. To motivate the purpose of embedding a workflow engine within OP4, let's look at an applications built with it.

Consider a financial institution wishing to implement an application that will allow its marketing team to accelerate the delivery of key marketing messages to customers, investors, employees and business partners. Increasingly, most financial institutions are engaging in multi-channel communications across Web, email, print, syndication, and wireless channels. The complexity of the undertaking is further exacerbated by the fact that the financial industry is subject to strict compliance rules by regulatory agencies. For example, all marketing collateral produced by a brokerage firm is subject to advertising compliance rules managed by the National Association of Securities Dealers, an agency that acts in concert with the Securities and Exchange Commission. The cost of making material errors in marketing collateral is prohibitively high.

In a typical scenario, a marketing communications manager may be responsible for running a campaign to launch a new financial product, say a new annuity product. The campaign involves the development and delivery of three kinds of content:

- A new section to the public Web site on mutual funds
- A print marketing brochure that will be direct mailed to investors
- An email-based awareness campaign to notify select, high net worth investors

The productivity application places the marketing communications manager at the hub of a coordination workflow (Figure 1). Using the application the communications manager requests original content from copy writers and product managers, routes the content for editorial review by professional editors, requests legal compliance reviews from the legal department, and obtains approval from senior management. Each participant in the process has his or her own role-specific dashboard to create, review, and approve content. Once the content creation and collaborative review process completes, the communications manager forks off the process to three channel-specific groups. The Web group styles and publishes the content on Web sites using HTML templates; an advertising agency lays out the content using Quark Xpress and produces a print brochure; and an email communications group initiates an email campaign to deliver the information to select customers.

Implementing the marketing compliance application with a workflow engine provides:

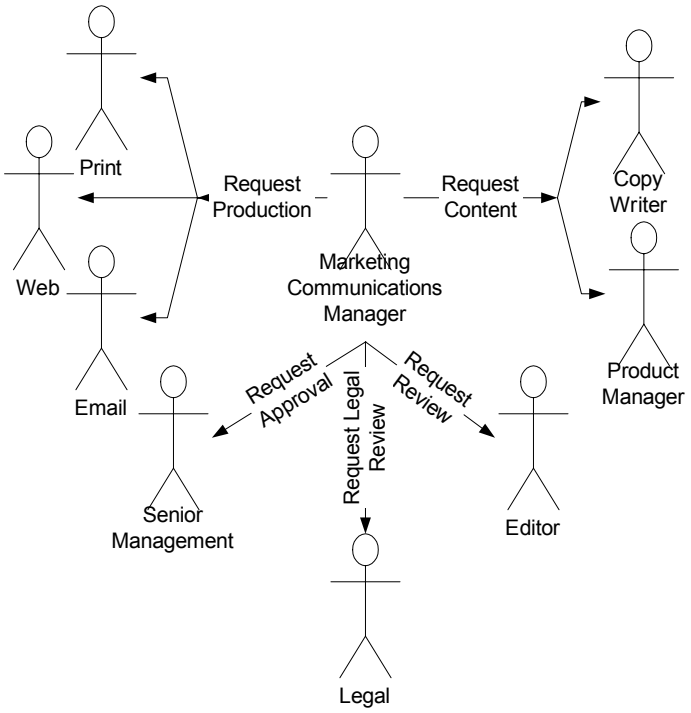

Figure 1: Marketing compliance application workflow

- The ability to change what a marketing campaign involves without changing the application code.
- Automatic routing of content to relevant participants in the right order, based on predetermined business workflows and compliance rules.
- Complete tracking and control of the project by the communications manager during the course of the project, including altering the flow of content midstream to accommodate emerging business needs.
- A rich audit trail that captures all events and decisions made over the course of the project.

5 Evaluating Workflow Engines

Following the buy versus build decision we assembled a cross-functional team to conduct the evaluation and make a recommendation. The team consisted of key members of the engineering and product management teams, and included people with past experience with workflow and collaboration technology.

The team decided that the evaluation framework would seek to address, for each product, three broad areas of concern:

Functional Requirements What can the workflow engine do?

Technology Constraints How has the engine been designed and built?

Business Requirements What are the implications of doing business with the vendor?

Next we look at how the above questions translate into specific criteria. Based on the OP4's business drivers we describe the priority we assigned to each criterion—*high* for must haves, *medium* for should haves, or *low* for nice to haves. Table 1 lists the evaluation criteria and their priorities in the context of OP4. Using the framework to select a workflow engine for other applications requires deriving these priorities from their business drivers.

5.1 Functional Requirements

In the context of workflow engine evaluation functional requirements revolve around the supported workflow features. Eliciting these requirements and assigning priorities is problem-dependent and involves significant input from domain experts. For OP4 we analyzed a set of representative workflows.

Area	Requirement	Priority
Workflow	Workflow constructs	High
	Routing slip metaphor	High
	Transactional aspect	Low
	Specification by business users	Medium
	Workflow template management	High
	Interpreted	High
	Access control	Medium
	Role-based assignment	Medium
	Composite workflows	Medium
	Integration with external services	Medium
	Audit trails	High
	Monitoring and administration	Medium
	Meta-data properties	Medium
	Work assignment policies	Medium
	Disconnected operation	Low
Technology	Technical infrastructure	High
	Open APIs	High
	XML compliance	Medium
	Embeddable presentation components	Medium
	Email integration	High
	Directory services integration	Medium
	Compact footprint	High
Business	Source code	Low
	Vendor support	Medium
	Pricing Model	High

Table 1: Summary of requirements and constraints

5.1.1 Workflow Constructs

Many workflow systems use an activity-based process model [4]. Workflow steps represent activity nodes. Control nodes such as conditionals, fork, join, and iteration determine the flow of control. Activities themselves may be performed by humans or applications; they can even be delegated to other self-contained workflows, thus providing a way to hierarchically decompose a complex workflow into a series of interconnected, nested sub-workflows.

Since OP4 delivers a wide range of collaborative content applications, the workflow capabilities of the framework must be sufficiently rich. We assigned a *high* priority to the availability of a rich set of control structures.

5.1.2 Routing Slip Metaphor

Different workflow systems have different design goals. One class of workflow systems revolve around the "routing slip" metaphor; these systems are proficient in routing electronic documents among a large set of workflow actors, much as paper documents are often routed between people using routing slips.

A typical OP4 scenario involves the collaborative creation of documents. These workflows follow the classic routing slip model. For example, a scenario may involve a content editor, an author, and a reviewer. The editor assigns an article to an author and a reviewer. The workflow system creates work assignments in the actors' work lists at the appropriate times. It also moves the work (i.e. the request, draft document, reviewed document, version history, annotations, etc.) between the various actors.

Since the routing slip metaphor is critical to the success of a workflow engine within OP4, we assigned it a *high* priority.

5.1.3 Transactions

A different class of workflow systems has traditionally focused on transactions. For example, most workflow systems in the financial and insurance industry are designed to optimize the processing of repetitive transactions such as mortgage applications, trade order flows, and insurance claims. In such scenarios, the workflow engine acts as an orchestrator of multi-step transactions that manipulate database records while simultaneously ensuring ACID properties (atomicity, consistency, independence, and durability) and high throughput volume.

Since OP4 is a framework for collaborative, human-centric applications, and not for transactional applications, we assigned a *low* priority to this aspect.

5.1.4 Workflow Specification by Business Users

Workflow systems provide different means of specifying workflow definitions. Graphical process editors are a popular choice. Graphical editors allow users to define workflows by dragging process elements on a canvas, connecting and configuring them without any programming. Packaged workflow applications typically include graphical process editors.

In contrast, other workflow systems rely on programmatic methods for workflow specification, such as scripting languages. This raises the skill set barrier in terms of who can define and manage workflow specifications in an organization. Usually, they require technical personnel to get actively involved.

OP4 must support self-service applications for business users. They should be able to design and specify workflows without the active participation of technical personnel. As a result we assigned a *medium* priority to this feature.

5.1.5 Workflow Template Management

Many workflow systems support template management. Templates are valuable for encoding "standard" (sub-)processes such as travel reimbursement or telecommunications provisioning. Workflow users can quickly define full-fledged workflows by combining these templates and customizing them to the problem at hand.

In the OP4 model workflow owners customize and instantiate workflow templates. For example, a process template may prescribe the steps required to publish a document. The amount of customization required to use this template in a workflow involves providing the work assignments and the time frame for completion.

We assigned a *high* priority to workflow template management.

5.1.6 Interpreted

First generation workflow systems made a clear distinction between build-time and run-time. Build-time involved defining workflow templates and translating their specifications into a compiled representation. Run-time involved executing instances of the compiled representation. However, this drastic separation between build- and run-times led to a well-deserved reputation that workflow technology was too rigid for practical use. Most workflow engines address this problem by treating a workflow as a set of rules to be interpreted at run time.

The interpreted approach to workflow execution has two major benefits, and both are of significant value to OP4. First, it allows for dynamic workflow instance modification. At run time business users deal with emerging exceptions or special situations by altering the routing rules. For example, a process may involve a legal compliance approval step. In one workflow instance the actors who produced the document may have worked closely with the legal staff during the creation of the document. Since the legal staff already knows the content, they may elect to skip the legal compliance step, especially if there is a looming deadline.

The second benefit has to do with the late binding of routing rules to a workflow instance. For example, in some collaborative projects it may not be feasible to completely define the workflow before execution begins. In other words, late binding allows the workflow to be complete its definition at run time.

OP4 regards workflow as a facilitating rather than an enforcing technology. Workflow actors are knowledgeable and drive the process. The workflow engine must help them perform their work and take over the mechanical parts of the process. It must not get in the way of their decision making processes. An interpreted approach is critical to this philosophy; we assigned it a *high* priority.

5.1.7 Access Control

Workflow systems typically permit control of several operational aspects of workflow execution. For example, most workflow systems let their users start, stop, and pause workflow instances. More

advanced functionality includes changing routing and work assignments, updating tasks, and even opening work items assigned to others.

OP4 workflows involve many actors. From these, only a subset of designated actors should have access to the workflow controls. For example, only the owner of a workflow instance should have permission to change its routing midstream. Similarly, not every actor should have permission to access every workflow template. The workflow engine should provide a means for defining permissions and enforcing them. We assigned a *medium* priority to this feature.

5.1.8 Role-based Assignment

A one-to-one mapping between activities and actors is not flexible. For instance, a workflow definition could specify that "John will review the article about Web services after Bill writes it." Should John become unavailable, the workflow can no longer execute.

Roles solve this problem by adding a level of indirection. The workflow definition refers to workflow actors by roles (or groups) rather than name. Roles specify the type of interactions actors can have with the workflow. In the previous example, a workflow definition that specifies "A reviewer will review the Web services article" is more flexible. A separate functional component such as a directory service usually maps roles to actors, thus providing the information that John is a reviewer.

In the context of OP4 we assigned a *medium* priority to this feature.

5.1.9 Composite Workflows

Some workflow systems support hierarchical decomposition. Hierarchical decomposition breaks a workflow into smaller sub-workflows. In other words, an entire workflow can represent a step in another workflow. Sometimes this decomposition allows several workflows to share common sub-workflows. In other situations it allows sub-workflows to execute within different organizational boundaries.

In the context of OP4 certain organizational processes are common across many different situations; for example, the legal compliance workflow can be shared among several line of business workflows. When combined with dynamic workflow modification, the ability to introduce a new activity (e.g., a legal review step) into a workflow at run-time and then bind the step to a predefined legal compliance workflow can deliver a lot of power and flexibility into the hands of a business user. For these reasons, we assigned a *medium* priority to this feature.

5.1.10 Integration with External Services

Workflow engines deployed in an enterprise setting must have the ability to access external systems that provide workflow-relevant data and services. For example, OP4 could employ a document translation service. A workflow instance may create a document in English, but it might be desirable to translate the document into Russian before its publication. In an ideal world, the final step of the workflow would be an automatic invocation of the external translation service, with the

original document as an input parameter. Once the translation is completed, the Russian version of the document would be returned to the workflow.

In addition to invoking external services, the workflow system should also be able to generate and respond to events to and from the external world. For example, a certain OP4 application may involve triggering a workflow when a new file is uploaded into a specific folder. It may also involve firing an event when a document has been published to a Web site so that interested subscribers can take action.

To accommodate the above scenarios, a workflow engine for OP4 should provide an event-based communication and coordination mechanism. We assigned a *medium* priority to this feature.

5.1.11 Audit Trails

A major benefit of workflow systems is that they record the state of running processes as they unfold. The audit trail information can serve several purposes. For example, with the increased scrutiny of business practices in the financial industry, audit trails are becoming a key compliance requirement mandated by the Securities and Exchange Commission and the National Association of Securities Dealers. Similarly, in the healthcare industry, the Health Insurance Portability and Accountability Act mandates audit trails for medical systems using any electronic means of storing patient data and performing claims submission. At a more mundane level, historic information provides an excellent way to perform process analysis in order to streamline and optimize them.

Given its focus on the financial industry, OP4 requires strong audit trail capabilities. We assigned a *high* priority to this feature.

5.1.12 Monitoring and Administration

Workflow systems manage the runtime data corresponding to each running workflow. A workflow monitor application enables administrators and power users to examine this information at run time. Possible actions may include performing diagnostics and gathering summary reports of workflows in progress, as well as controlling their execution via start, stop, suspend and resume commands. These permissions must be role based and protected by access control rules.

Given the project-oriented and collaborative nature of OP4 applications, the ability to visualize and control the exact state of workflows in the system has a *medium* priority.

5.1.13 Work Assignment Policies

The interaction between a workflow system and workflow actors can be governed by many different policies. The specific work assignment policy that makes sense in a given scenario depends on the business context. The most common policy has the workflow engine pushing work items to specific individuals via dedicated worklists. Other policies may involve a centralized work item queue that is accessible to a pool of actors who pick work items based on expertise. This model is popular in exception handling workflows in financial trade flows; the expertise to know what is wrong with a specific trade requires domain experience that is beyond the scope of most workflow engines.

Alternately, role based assignment policies may involve work items being assigned to a group of resources. The first actor to accept becomes the owner; the work item is then retracted from the other worklists.

The functional requirements for OP4 framework called for a dedicated worklist metaphor with role-based work assignment. Complex work assignment policies are not critical to the success of the framework. We assigned a *medium* priority to this feature.

5.2 Technology Constraints

The technology constraints focus on the platform (software and hardware) and the integration points. Several constraints from Table 1 are general; we have omitted them from this section.

5.2.1 Open Application Programming Interfaces

Open and documented programmatic access to all aspects of the workflow engine functionality lie at the focal point of our evaluation. The programmatic access includes the creation and management of workflow templates; the ability to start, stop, suspend, and resume workflow instances; the ability to attach documents to workflows, the assignment and management of tasks to and through task lists; adapting and modifying workflows on the fly; and managing audit trails and reporting capabilities.

We determined that securing access to all of the above functions using Java classes and methods, as well as javadocs, programming guides, and cookbooks has a *high* priority.

5.2.2 XML Compliance

XML is central to the way OP4 manages documents, content, and their versions. Likewise, the ability to represent workflow templates as XML documents facilitates version control of templates, as well as interoperability. In addition, retrieving the state of a workflow instance as XML allows for superior reporting capabilities in the long run. Finally, a workflow engine that can also respond to XML messages would be of considerable value as well.

While XML compliance is a desirable and elegant feature, it is not critical to the success of OP4. We assigned it a *medium* priority.

5.2.3 Embeddable Presentation Components

OP4 applications are always Web-based without client-side components. Therefore, any presentation components that accompanies the workflow engine such as graphical workflow designers, worklist controls, workflow monitoring tools, and so forth should be browser-based. Embeddable HTML and Java applets would be valuable because OP4 could reuse them.

To accelerate OP4 development we assigned a *medium* priority to the availability of web-enabled presentation components for the workflow engine.

5.3 Business Requirements

The business requirements focus on vendor- and product-related issues.

5.3.1 Source Code

Ideally application developers have access to the source code of all third party software components they use. However, source code represents intellectual property to the software vendors. As a rule, they are loathe to part with it—exceptions to this rule do exist, however. In addition, to the best of our knowledge, no competitive open source workflow systems are available yet.

Nevertheless, the absence of source code should not be a road block. It can be equally valuable to extend a workflow engine designed for growth, providing clean, well-documented object oriented APIs. In such situations we recommend that the workflow vendor escrows the source code with a credible third party escrow agency. This protects application developers in case the workflow vendor declares bankruptcy or goes out of business. Such an event then triggers the release of the source code to the application developer.

In the light of these considerations we assigned a *low* priority to the availability of source code.

5.3.2 Vendor Support

Assuming that workflow engine source code is not available, the application developers must have very clear contractual agreements with the workflow vendor regarding support responsibilities. To keep support costs low we recommend that the developers take responsibility for first level support, i.e., supporting the workflow needs of the users of the embedding application. However, the application developers must necessarily rely on the vendor for second level support, i.e., dealing with bug fixes and other issues that cannot be resolved without source code access.

A second level support agreement must also explicitly spell out the resolution process and levels of responsiveness expected from the workflow vendor when bugs or other workflow engine problems are reported. For example, an agreement may specify that the workflow vendor must acknowledge all reported problems within forty eight hours. Furthermore, it may also specify that based on the severity of the problem, the vendor must provide a resolution within one, two, or four weeks. The agreement should also provide for free upgrades to certain future releases of the workflow engine, and provide for sufficient number of training classes for the application developer's engineering staff. Typically, these are all issues that are best resolved through negotiation.

Based on the importance of the availability of second level support, we have assigned it a *medium* priority.

5.3.3 Pricing Model

Last but not least, the pricing model is an important (and regrettably sometimes the only) consideration in choosing software components. Workflow vendors are waking up to the opportunity of licensing their products as embeddable components rather than enterprise software. However, in

our experience few workflow vendors have adjusted their pricing models to reflect the needs of application developers.

Workflow vendors interested in capturing this emerging market must perceive application vendors as *channels* rather than end customers. As such, pricing models that emphasize upfront software licensing fees will not win favor with application developers. Instead, workflow vendors must understand the business model of the application vendor and tailor their pricing models accordingly. Application developers earn revenue when their products are sold. They are much more likely to share with the workflow vendor a percentage of the revenue than pay large upfront fees.

Astute workflow vendors are developing OEM pricing models for application developers that have two components: a modest upfront fee, and a royalty-based arrangement where the workflow vendor can participate in successful sales of the application, as a percentage of total revenue.

Given the above considerations we assigned a *high* priority to a royalty-based pricing model with a low upfront fee.

5.4 Evaluation Summary

We distilled the answers from the vendors who answered the (informal) request for proposals into a comparison table. The example from Table 2 shows the results for three products; a $+$ denotes that the corresponding workflow engine is better, a $-$ denotes that it is worse, and a 0 means that there is no significant difference. Since we focus on the evaluation process as opposed to specific outcomes, the table does not disclose the identities of the evaluated products.

Upon assembling vendor responses we computed a cumulative score for each product by marrying the score on each criterion with the priority assigned to that criterion. Effectively, this was a quantitative calculation that combined the information in Tables 1 and 2. At the end of this phase Fujitsu Software's i-Flow scored the highest.

In the next phase we performed a prototyping exercise to test i-Flow in context. We used a simplified version of the workflow shown in Figure 1. The results of the prototyping phase provided enough information to make a final decision.

More than a year has elapsed since we recommended i-Flow. In hindsight, choosing it was a sound decision. Certainly, there have been some bumps along the way. Overall the evaluation described here worked well and provided us with early insight into the areas of potential risk.

6 Summary and Conclusions

Software developers are turning to workflow technology to implement their process-oriented applications. Consequently, new as well as familiar workflow vendors are beginning to introduce embeddable workflow engines into the market.

Embeddable workflow engines address different requirements relative to full-fledged workflow management systems, and serve a different audience. In addition to workflow capabilities such as process models, control constructs, work assignment policies, ad-hoc workflow and so forth,

embeddable workflow engines must address additional needs such as integration with other application components, the ability to customize through programming techniques, and architectural alignment with the embedding application.

This article presented a practical evaluation framework for embeddable workflow engines. We began with an evolutionary sketch, emphasizing the paradigm shift from stand-alone controlling applications to components of modern enterprise architectures. We then discussed the benefits and risks of embedded workflow, and introduced a case study providing the context for the evaluation framework. We discussed the evaluation along the functional, technological and business dimensions, and concluded by showing a subset of the framework's outputs applied to our case study.

Workflow engine evaluation covers a broad range of issues, such as framing the process flows, identifying the workflow features necessary to implement them, assessing the degree of architectural alignment, evaluating integration efforts, and prioritizing other technical factors. Our framework represents a blueprint for practitioners seeking to embed a workflow engine suitable for their application. This blueprint was instrumental in our making a sound decision about a dozen workflow engines. Practitioners evaluating workflow engines could reap the benefits of a proven evaluation framework by adapting and extending it with new features and constraints.

About the Authors

Dragos Manolescu is a software architect with ThoughtWorks, an application development and consulting company. His research interests include lightweight workflow systems, software architecture and object technology. His research publications are available on the web from `http://micro-worklfow.com/`. Dragos has earned a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign for his research on workflow architectures. Contact him at `dmanolescu@thoughtworks.com`.

Santanu Paul Santanu Paul is a Vice President at Virtusa Corporation, a global software services firm, and General Manager of its technology center in India. He has served as CTO of product companies such as Openpages and Viveca, and as a research staff member at IBM's T. J. Watson Research Center. Santanu has a Ph.D. in Computer Science from the University of Michigan, and a Bachelor of Technology in Computer Science from the Indian Institute of Technology. His areas of interest include software engineering, architecture, workflow management, and content management. He has published over twenty papers and articles and has multiple US patents, awarded or pending. Contact him at `spaul@virtusa.com`.

References

[1] Frank Leymann and Dieter Roller. *Production Workflow—Concepts and Techniques*. Prentice-Hall, Upper Saddle River, New Jersey, 2000.

[2] Dragos A. Manolescu, *Micro-Workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development.* Ph.D. thesis, University of Illinois, Urbana-Champaign, October 2000.

[3] David Garlan, Robert Allen, and John Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In *Proc. 17th International Conference of Sofwtare Engineering*, Seattle, WA, 1995.

[4] Andrzej Cichocki, Abdelsalam Helal, Marek Rusinkiewicz, and Darrell Woelk. *Workflow and Process Automation—Concepts and Technology*. Kluwer Academic Publishers, 1998.

	Requirement	Product		
		A	**B**	**C**
Workflow Requirements	Workflow constructs	+	0	+
	Routing slip metaphor	0	0	0
	Transactional aspect	0	+	+
	Specification by business users	0	0	0
	Workflow template management	0	0	0
	Interpreted	−	0	0
	Access control	+	0	+
	Role-based assignment	+	−	+
	Composite workflows	0	0	0
	Integration with external services	+	−	+
	Audit trails	−	+	+
	Monitoring and administration	0	0	0
	Meta-data properties	0	0	0
	Work assignment policies	+	+	0
	Disconnected operation	0	+	−
Technology Constraints	Technical infrastructure	0	0	0
	Open APIs	0	0	0
	XML compliance	+	0	+
	Embeddable presentation components	0	0	0
	Email integration	0	0	0
	Directory services integration	0	−	0
	Compact footprint	+	0	+
Business Requirements	Source code	0	0	0
	Second level vendor support	0	+	0
	Price	0	+	0

Table 2: Comparison table for 3 workflow engines