Case Studies in Workflow: Three Approaches

“Lightweight workflow” is both an oxymoron, and a continual aspiration of the many stakeholders in the repository community.

As part of the Hydra Project, the University of Hull, University of Virginia and Stanford University are collaboratively developing a reusable application framework that will sit on top of Fedora, and provide the mechanism for rapidly deploying repository-powered applications that support different content types, use contexts and workflows.

Developing support for workflow (defined here as orchestrating multistep processes that may include human interaction) is integral to the project. And because the Hydra framework’s primary design is to support reuse and extensibility to new application needs, the methods for supporting workflow must themselves be easily adaptable.

While all three institutions are adopting the same Hydra application framework, the partners consciously chose to take three different paths in implementing and integrating workflow into the overall solution. This paper briefly details the three different workflow approaches the collaborators are taking, why they chose them, and the apparent pro’s and cons of each.

The Hull Approach: BPEL
The University of Hull first started its investigation of repositories in 2005 though the JISC-funded RepoMMan project. At the time both the University and the JISC were also interested in exploring the possibilities of using the Business Process Execution Language (BPEL) as a means of orchestrating Web Services. The University team was convinced that their embryo repository should be Web Services based and so it was natural that the two investigations should be brought together. Thus it was that RepoMMan started out with BPEL as ‘a given’.

The first few weeks of the RepoMMan project included a detailed survey of the available software and the decision was made to use the open source version of the Active Endpoints BPEL engine. BPEL is an awkward language to write by hand and the Active Endpoints offering had the additional advantage of a graphical interface to help create and orchestrate code.

The ‘RepoMMan tool’ available at the end of that project in 2007 allowed a user to identify a file on their local computer and upload it to a private area of the repository effectively with a single click. This click invoked a series of BPEL processes which transferred the file to the University SAN and built the corresponding digital object in the Fedora repository. In fact, the full sequence did allow the user to provide metadata, pre-populating the form where possible by passing the file to another Web Service to extract descriptive fields and referring to the user’s working context to populate others.

The University was fortunate to receive further JISC funding in 2007 for the two-year REMAP project which built on this initial work in two ways. Firstly, the RepoMMan tool
has been extended so that from a user’s private repository space an object can be passed to
the publishing queue of the repository proper. Here the object is checked by repository
staff before being made available to others. This BPEL driven publishing process takes the
content of the user’s object and from it creates a new object, owned by the repository, that
conforms to a standard content model. The model is different depending on the type of
digital content; the process may create archival datastreams, surrogate datastreams and/or
multiple metadata datastreams. Secondly, as part of this process a datastream is
embedded to facilitate ongoing Records Management and Digital Preservation (RMDP).
The contents of this datastream are effectively copied to a calendar server which tracks the
triggers it contains and fires off events over a period of time. In the short term, for
instance, an email is triggered to the original author to say that their material is published
and to provide its URL. Depending on the content of the digital object, it may trigger an
alert in five years time to warn that the content is likely becoming dated and should be
hidden (which might actually be a default action). A wide range of differing triggers and
alerts has been identified. All the functionality around the repository and its objects is
provided by BPEL-orchestrated Web Services based on version 5 of the Active Endpoints
(now Active VOS) engine.

Hull’s experience of BPEL has generally been very positive. Although, predictably, there
have been problems arising from working with emerging technologies the engine and its
graphical design tool has served us well. Sadly, the company has decided that it should no
longer develop an open source BPEL engine and version 5 will be the last such offering.
Further, the graphical interface we have been using to generate code is no longer available
free of charge. Consequently, although we are adequately served for the moment, we at
Hull are watching the emerging workflow approaches of our Hydra partners with interest.
Commercial BPEL engines are too expensive to contemplate!

The University of Virginia Approach: Orchestrating Web Services and User
Interactions
The University of Virginia has spent many years crafting special purpose scripts that
hardcode content-specific workflow processes for both preparing and then ingesting
content into a Fedora repository. When we began work on what later became the Hydra
project, we knew that we needed a more flexible approach to creating and managing these
workflows. And, since we envisioned supporting not only the processing of large batches
of institutional content (which is usually done by professionals who can handle less than
perfect processes) but also individual scholars preparing and ingesting their own content,
it was clear that we needed a workflow environment that directly orchestrated interactions
with users as well as orchestrating backend processes such as web service calls.

Based on Hull’s experience with BPEL, we began working with the open source BPEL
engine from Active Endpoints. We soon discovered, however, that this solution was too
heavyweight for our purposes and the decision to not pursue this solution was then made
easier by the discontinuation of the open source version of the engine.
After investigating a number of other possibilities, we decided to build our own lightweight workflow engine to support the orchestration of web services and user interactions. The current design is effectively a workflow scripting language tied into a long-term persistent data store and execution engine. The basic idea is that user workflows can be orchestrated by simple scripts whose output is automatically transformed into an interactive user interface. The scripts are then responsible for orchestrating arbitrary web services, further interaction with the end-users, and maintaining long-term state.

The Engine is composed of several components:

- lex/yacc-based compiler to parse the workflow script files and generate a compiled Abstract Syntax Tree (AST)
- core engine which manipulates the compiled AST files and maintains state in a flat-file database
- service wrapper which exposes engine functionality to the outside world as a web service
- application front-end which uses the exposed functionality to provide the end-user with an interactive user interface.

We have implemented an initial prototype of this design and are continuing further development.

**The Stanford Approach: WorkDo**

The Stanford Libraries’ approach to workflow in its digital library infrastructure has been a minimalist one. Leery of workflow efforts that take on a life of their own and that require their own separate resources to sustain, we have worked on a lightweight approach to supporting work that satisfies basic lifecycle requirements for the deposit, accessioning, archiving and delivery of digital resources. This method does not involve external rule or state engines, messaging, or separate process orchestration software, yet it supports all the cascade of tasks necessary to manage these digital objects. For this reason, rather than calling this “Workflow” (with a capital W), we call this approach “WorkDo”.

A Digital Object Registry (DOR) is at the heart of our digital library management architecture. DOR, a Fedora-based repository, supports managing a digital object from the moment it is identified as a library resource through the transfer of content to separate preservation/discovery/delivery environments.

The work required to "ready" a resource, to describe it and prepare it for preservation and delivery, is thought of as a set of conditions that must be addressed -- "get descriptive metadata", "validate files", "generate PDF", etc. Each conditions implies a task that must be performed, a unit of work to satisfy the conditions that might be performed by humans or programs. Web-based tools support tasks that require human input. When a task is an automated one, a relatively simple script should suffice. These are can perform basic tasks such as triggering the exchange of data, or invoking reusable service calls. Because of the autonomous nature of such scripts we affectionately refer to them as robots.
Work tasks are independent and generally may be performed in any order. When there are dependencies between steps, such as "don't catalog the item if it hasn't been copyright cleared", we find that a simple scheme of pre-requisite conditions is sufficient without more general facilities for choreographing tasks.

The key to the WorkDo approach is that it leverages data placed in the objects themselves as follows:

- Each object in DOR has locally defined resource-management metadata plus a special datastream to describe processing conditions and their state for that object.
- By placing such work related information in the object, it can be indexed (using SOLR) alongside other useful processing information like collection and selector identity to identify records ready for a particular process.
- Thus simple queries can be used to establish queues and such queues define the work ready for a particular robot or human interaction at any given time.
- The same queries provide ongoing management information about the flow of objects through the system. They can even be exposed as facets in an administrative discovery environment.
- Simple REST based interactions based on Fedora service calls are used to identify queues and update state.

This approach was an expedient way to support the needs of several projects underway in the Stanford digital library and is still evolving. We recognize that it does not have all the capabilities of a fully featured workflow system. It is associated with a specific set of objects and tasks so could not coordinate work across environments and related objects. It is a comfortable fit for a certain sized "lifecycle" unit of work and would not be suited for the orchestration of many small processes. It does not support complex and highly dynamic workflows. Yet so far it has proven quite capable of meeting our requirements. We find the integration of the workflow data with the object to be particularly effective in satisfying both the informational and processing needs of our digital resource management.

**Conclusion**

This paper is presented in the hopes of stimulating discussion around a need that faces every repository environment, but a need for which a clear and compelling solution has yet to present itself. We anticipate that this presentation will catalyze a Birds of a Feather discussion during the user group meeting.