“Big Workflow” for Enterprise Applications

Jeff Sutherland and Steve Alpert
IDX Systems Corporation, 29 September 1999
mailto:jeff.sutherland@computer.org

Introduction

Integration of multiple disparate systems to support enterprise business processes across a supply chain or within a healthcare delivery system is a significant new opportunity presented by the internet infrastructure that is becoming widely accessible to all interested parties. Taking advantage of that opportunity requires implementation of an enterprise workflow capability that transcends previous workflow implementations or standards.

This paper provides an overview of the current evolution of workflow standards and implementation efforts designed to support enterprise workflow across multiple institutions, each running a wide variety of disparate systems. The goal is to use internet standard (or evolving internet standard) capabilities throughout.

Background

“Big Workflow” is a term that has been used at IDX Systems Corporation to describe workflow that crosses multiple applications and multiple vendors to support patient flow across an Healthcare Integrated Delivery Network (IDN). Similar systems have been implemented or are under construction in manufacturing and other vertical application domains.

As an example, HeathSystems Minnesota has carefully defined their business processes for managing patient flow across multiple institutions within their IDN. Workflow must be managed across 134 large applications provided by dozens of vendors. They would like the capability to superimpose HealthSystems business processes across all vendors and they would like to be able to modify their business processes globally without modifying vendor systems.

“Big Workflow” is federated, heterogeneous, distributed, enterprise workflow. An enterprise in this context consists of federated systems that have their own rules and regulations and support higher level interaction between systems (like the United States which is a federation of states). Systems are distributed (regionally, nationally, or globally) and may reside on different (heterogeneous) hardware/software platforms. Completing a **Process Instance** (see WfMC definitions in later section) may require completion of multiple work items by multiple users (human and automated) performing a myriad of roles and supported by large numbers of computing systems.

Workflow systems have become both more widely deployed and more adaptable in recent years. Traditionally, they were monolithic and required the installation of client software on the desktop. With the advent of Java, commercial software quickly moved towards applet-based clients and to more distributed systems.

More radically, workflow systems are increasingly taking advantage of Internet technologies to become more open. One example is the Workflow Management Coalition (http://www.aiim.org/wfmc/mainframe.htm) initiative to promote XML-based interoperability among workflow systems, which builds on earlier work on the draft Simple Workflow Access Protocol (SWAP) initiative. The Internet Engineering Task Force (IETF) Simple Workflow Access Protocol is a critical background resource for this work (http://www.ics.uci.edu/~ietfswap) as all efforts are directed at Internet implementations. A good overview can be found in SWAP: Leveraging the Web to Manage Workflow (http://www.ics.uci.edu/~ietfswap/swap-paper.pdf). At a higher-level of abstraction are efforts to develop adaptable agent-based workflow systems (http://www.aiai.ed.ac.uk/~paj/ijcai-wflow-wshop/). Workflow specification, standardization, and development efforts are intensifying as the coordination of work among organizations becomes an important component of electronic commerce.

Additional background may be found in Object Management Group (OMG) standards documents. The WfMC worked with the OMG BODTF Workflow Special Interest Group to develop jFLOW, now the OMG standard for workflow (ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf). The current state of jFLOW was presented at the OOPSLA'98 Business Object Workshop by the Chair of the WfMC. (http://jeffsutherland.com/oopsla98/mts.html). Another OMG BODTF document with concepts of interest is the latest version of the Business Object Component Architecture (http://192.245.64.8/dat/Download/standards info/boca0599.pdf).

Moving beyond the state-of-the-art: Rainman

At the OOPSLA'97 Business Object Workshop, Santanu Paul presented Essential Requirements for a Workflow Standard. He reviewed work on Rainman: A Workflow System for the Internet at IBM T.J. Watson Research Center and pointed out deficiencies in the WfMC architecture.

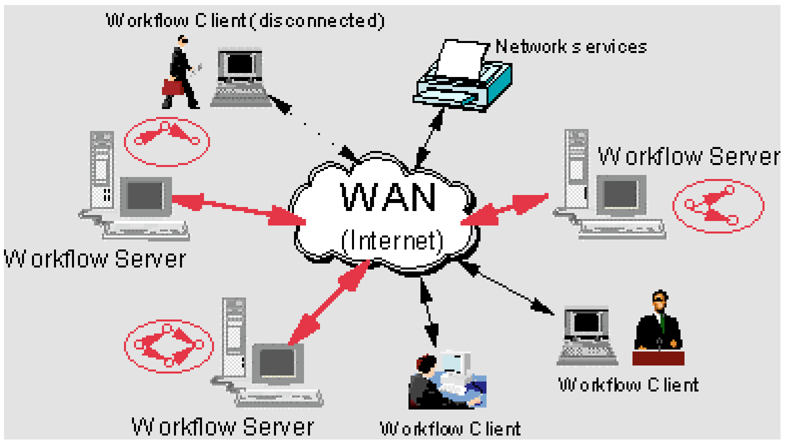


Figure 1: Distributed Workflows: Heterogeneous Servers, Participants, and Applications on a WAN from Essential Requirements for a Workflow Standard.

For users distributed on the Internet, the following requirements are essential:

Based on scenarios such as the one above, Santanu Paul et. al. established the following requirements for a distributed workflow infrastructure seem appropriate:

- **Heterogeneity of workflow components**: The basic execution model must be that of multiple, heterogeneous workflow backends (servers or systems) on the wide area network that are capable of executing workflows. Symmetrically, heterogeneous workflow participants or service providers on the network such as humans, applications, business objects, and invoked workflow systems that are utilized during workflow execution should be able to receive work from multiple workflow backends.

- **Flexible and scalable workflow participation**: Workflow participants should be able to participate using traditional devices (such as desktops) as well as nontraditional devices (such as thin clients, or personal digital assistants). Participants should not have to know *a priori* of specific backends that send them work. They should not have to maintain dedicated connections to the workflow backends from which they receive work, since this precludes thin clients from participating effectively as workflow clients.

- **Plug-and-play support for business objects and applications**: Third-party providers or developers of business objects and applications should be able to 'workflow-enable' their components and plug them into the distributed workflow infrastructure cheaply. They should not have to know any details about the workflow backends that are going to use these components in the course of workflow execution.

- **Disconnected operation**: Workflow participants may be mobile and infrequently connected to the network. Location-independent, disconnected workflow participation must be natural to the workflow execution model.

- **Recursive Decomposition of workflows, dynamically**: The execution model must support dynamic recursive decomposition of workflows. Work items assigned to participants may be *dynamically* refined and implemented as nested workflows, in a recursive manner. In other words, late-binding of work items should be allowed and it should not be necessary to statically define every step (and nested steps) of a workflow prior to its execution.

Essential Requirements for Interface to Other Workflow Systems

The Workflow Management Coalition (http://www.aiim.org/wfmc/mainframe.htm) initiative to promote XML-based interoperability among workflow systems is an evolving document, currently in alpha phase. It is a further development of the IETF SWAP document referenced previously which assumes that the HTTP protocol can be extended. At time of writing, Internet Explorer 5.0 does not support HTTP extensions, just one of the myriad of implementation details that must be resolved.

In order for “Big Workflow” to support interoperability between other web-based workflow system, several interfaces need to be supported. All definitions below are quoted from WFMC-TC-1023, Draft 1.0, 20 April 1999:

- **ProcessDefinition –** used to create process instances and find general information about process definitions. It contains the methods **PropFind**, **CreateProcessInstance**, and **ListInstances**.
- **ProcessInstance** - used to communicate with a particular instance of a process definition (or enactment of a service), acquiring information about the instance, controlling it and modifying it’s properties. Since a given instance may continue to execute for any amount of time, methods may be called on an instance while it is executing. These methods may obtain status information, supply new input values (although how these are handled is dependent on the specific implementation and the task being performed), or obtain early results (although the results of a process instance are not final until the instance has been completed). This interface contains the methods **PropFind**, **PropPatch**, **Terminate**, **Subscribe**, **Unsubscribe** and **GetHistory**.
- **Observer** - allow requesters of work or other resources to monitor the progress of a process instance and be notified upon it’s completion. Methods are provided to allow a resource implementing this interface to obtain information about a process instance for which that resource is a registered observer, as well as to notify registered observers of events if the implementing resource is the performer of the work. Once an instance’s registered observers have been notified of it’s completion or termination, the resource responsible for that instance is not required to maintain it any longer. Therefore, while most events are not required to be notified, the “completed” and “terminated” events must always be notified in order to indicate to observers that the results are final and this resource may shortly become inaccessible. This interface contains the methods **PropFind**, **PropPatch**, **Complete**, **Terminated** and **Notify**.
- **ActivityObserver** - is very similar to the Observer interface, with the primary difference being that it is relevant to a particular task, or “activity”, within a process instance. Implementing the interface at this level provides a way to find the particular process instance that contains a given activity, which further provides a way of tracking all the activities contained by that process instance. Finally, if any of these activities are other (sub) process instances, they can also be discovered via this interface. This traversal of instance/activity relationships can support distributed, nested process management to any level. Because of this potential nesting, it is important to note that the **ActivityObserver** resource must supply/expect information going to/coming from a process instance to be in that instance’s context. Therefore, data exchanged with that instance will be expressed with the set of fields relevant to that particular instance. This interface contains the methods **PropFind**, **PropPatch**, and **Complete**.

- **WorkList -** allows the potentially numerous distributed processes being performed throughout an interoperating enterprise to be filtered based on their relevance to a particular (human) user. While the WorkList and WorkItem interfaces parallel the ProcessDefiniton and ProcessInstance interfaces in some respects, they differ in that WorkList and WorkItem represent only the information about work to be performed, rather than representing the actual resources that perform it. Implementing the WorkList interface, when a system assigns an activity to a user it would create a workitem for that user on the user's worklist server (identifiable via a user directory) that corresponds with the activity being assigned. This interface contains the same methods as the **ProcessDefinition** interface (**PropFind**, **CreateProcessInstance**, and **ListInstances**).
- **WorkItem** - differs somewhat substantially from **ProcessInstance** in that its resource does not have any predefined action. It is merely a pointer to another resource that performs an activity, picking up the description of the task from the activity. This interface also provides a means of indicating the activity's status to the user. The user does not directly interact with the workitem, but rather with the activity via the workitem's pointer. Once the activity is completed, the workitem should be removed from the user's worklist. This interface contains the same methods as the **ActivityObserver** interface (**PropFind**, **PropPatch** and **Complete**).
- **EntryPoint -** used to create a process instance with a particular starting point within a process definition. If this interface is to be implemented, the process definition resource will have to provide a series of predefined entrypoints. Each entrypoint can potentially have it's own input requirements, access control, etc., which will have to be enforced by the implementation. This interface provides valuable functionality, but is optional, as process instances can be created directly through the **ProcessDefinition** interface if no special entrypoint is required. This interface contains the methods **PropFind** and **CreateProcessInstance**.

The following table indicates which methods are contained by each interface, and conversely which interfaces contain each method.

	Process Definition	Process Instance	Observer	Activity Observer	Work List	Work Item	Entrypoint
PropFind	X	X	X	X	X	X	X
PropPatch		X	X	X		X	
CreateProcess Instance	X				X		X
ListInstances	X				X		
Terminate		X					
Subscribe		X					
Unsubscribe		X					
GetHistory		X					
Complete			X	X		X	
Terminated			X				
Notify			X				

Agent Technology

The trajectory of enterprise systems is moving from:

- Monolithic systems to modular systems (APIs are there to be used)
- Modular systems to object-oriented systems (RMIs are there to be used)
- Object systems to component systems (loosely coupled)
- Component systems to internet workflow systems (XML messaging)
- Workflow systems to agent-based systems (goal-seeking, brokering, etc.)

To a significant extent, each step is a prerequisite to the next step. Let's take a simple example, prescription renewal. The average pharmacist spends 70% of their time calling doctors to try to get approval for prescription renewals. Patients are frustrated and often lack medication.

A patient should be able to pull up their medications in a Web browser and click to request a prescription renewal. This should generate a workflow that allow the online medical record to approve the medication 80% of the time. The remaining 20% of the time, the request should be put on the physician's worklist and the patient should be able to track the request on the internet like a Federal Express package. If the physician does not renew or reject in a specified interval, the workflow should escalate this task. When approval is received, the computer should submit the request, print the labels, and put the prescription on the appropriate pharmacist's worklist. In no case should the pharmacist ever have to call the doctor for approval. This would eliminate 70% of the total pharmaceutical workload in the United States!

Better yet, the patients agent should be monitoring prescription renewal requests and prompting the patient to renew or not renew before it became an emergency.

The Agent Working Group, part of the OMG Electronic Commerce Task Force has developed a green paper on Agent Technology, OMG Document ec/99-08-06. Emerging business object systems need more intelligence and are essentially complex, adaptive systems. See:

Sutherland, J. Business Object Component Architectures: A Target Application Area for Complex Adaptive Systems Research. In Patel , D., Sutherland, J., Miller, J., (Eds.) *Business Object Design and Implementation II: OOPSLA'96, OOPSLA'97, and OOPSLA'98 Workshop Proceedings.* Springer, 1998.

Workflow systems need to interoperate with agent infrastructures. The Agent Technology Green Paper references:

Foundation for Intelligent Physical Agents. *FIPA98 Agent Management Specification.* Geneva, Switzerland, Oct 1998.

FIPA98 lays out an architecture for an agent infrastructure which includes an **AgentWrapper**. An area of future development is creating of an AgentWrapper to "Big Workflow" that enables seamless interoperability with an agent environment.

This work is of particular interest for e-commerce systems (which are already incorporating agent technologies). Healthcare will move to an e-commerce environment and these technologies will need to be supported.

Definitions

The Workflow Management Coalition (WfMC) Glossary and Terminology (http://www.aiim.org/wfmc/standards/docs/glossy3.pdf) is an important source of definitions that we will use wherever possible (see Figure 1 below).

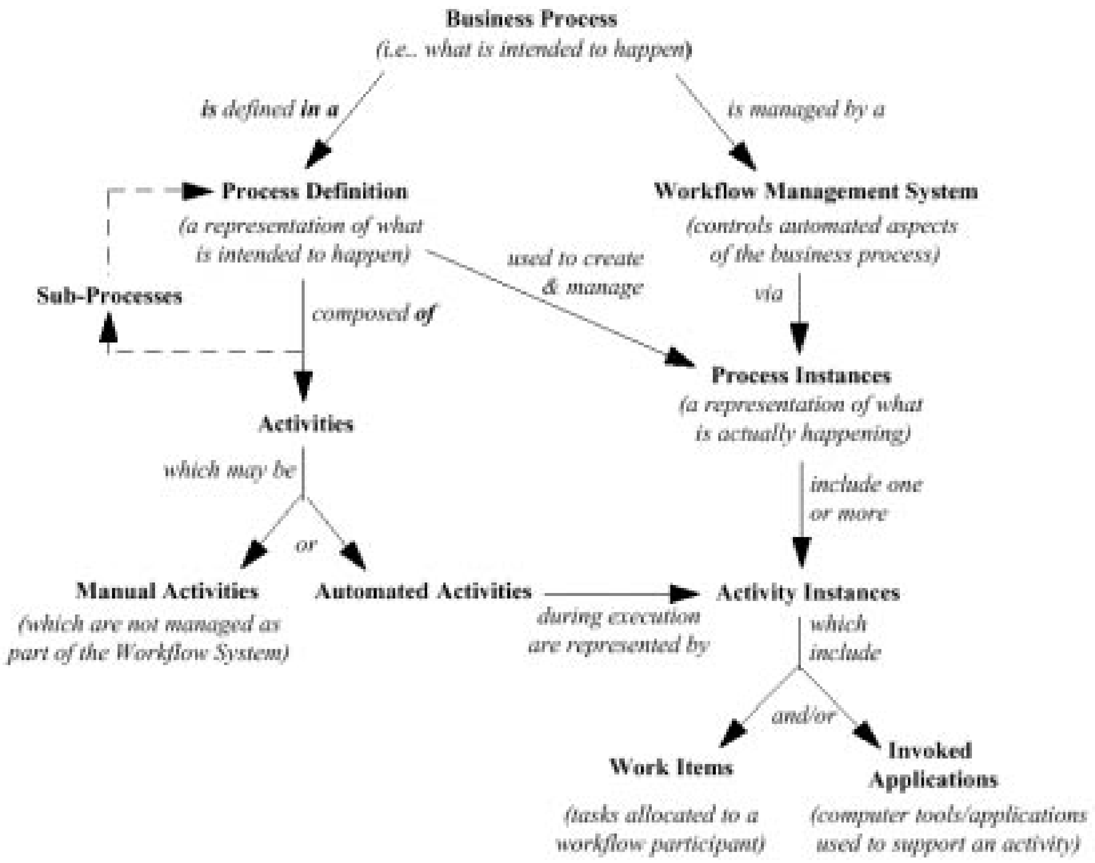

Business Process
(i.e.. what is intended to happen)
is defined in a
is managed by a
Process Definition
(a representation of what is intended to happen)
Workflow Management System
(controls automated aspects of the business process)
used to create & manage
via
Sub-Processes
composed of
Process Instances
(a representation of what is actually happening)
Activities
which may be
include one or more
or
Manual Activities
(which are not managed as part of the Workflow System)
Automated Activities
during execution are represented by
Activity Instances
which include
and/or
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

Figure 1: WfMC Workflow Glossary –Relationships between basic terminology

Process Instances are a graph of nodes of activity, typically a tree of nodes where units of work flow in a logical sequence. Each node or **Activity Instance** may be delegated to any user (human or automated) on the enterprise network by a workflow engine (or a **Workflow Management System**).

The Enterprise can consist of one corporate entity or may extend its reach to suppliers and partners. In a healthcare setting, patients will ultimately be viewed as part of the enterprise, and will be able to initiate enterprise workflows on their own behalf.

The ability to have multiple workflow engines residing on any server in the enterprise assign and track units of work (**Work Items**) to be completed on any other server in the enterprise will allow the CIO of an enterprise to specify an enterprise workflow for a patient (or other package of materials that need work) and layer the global workflow on heterogeneous application systems supported by multiple vendors.

Workflow Concepts

The workflow management system will consist of **Performers**, processes that perform work items on a **WorkList**.

A **Process Manager** is contacted to initiate workflow with a requested **Process Template**. The **Process Manager** looks up the appropriate **Process Template** and creates a **Process Instance**. Then:

1. A validity check is done to insure sufficient context is available to complete the request.
2. A **Policy Manager** is consulted to get an assignment of the required resource (**Performer**).
3. An **LDAP** is consulted to find the **WorkList** for that **Performer**.
4. A **WorkflowEngine** is asked to determine the “first activity” of the **Process Instance**.
5. The instance together with the required context are inserted on the **WorkList** for the **Performer**.

A **WorkflowEngine** is a lightweight process that:

1. Is a **Performer** with a performer interface.
2. Can traverse the nodes in a **Process Instance** graph. It puts **WorkItems** on **Performer WorkLists**.
3. Works off a persistent **WorkList** that is specialized in the sense that it keeps track of all the workflow requests and their status on a specific server.
4. Typically runs a department in an enterprise.
5. Can run on any server in the enterprise.

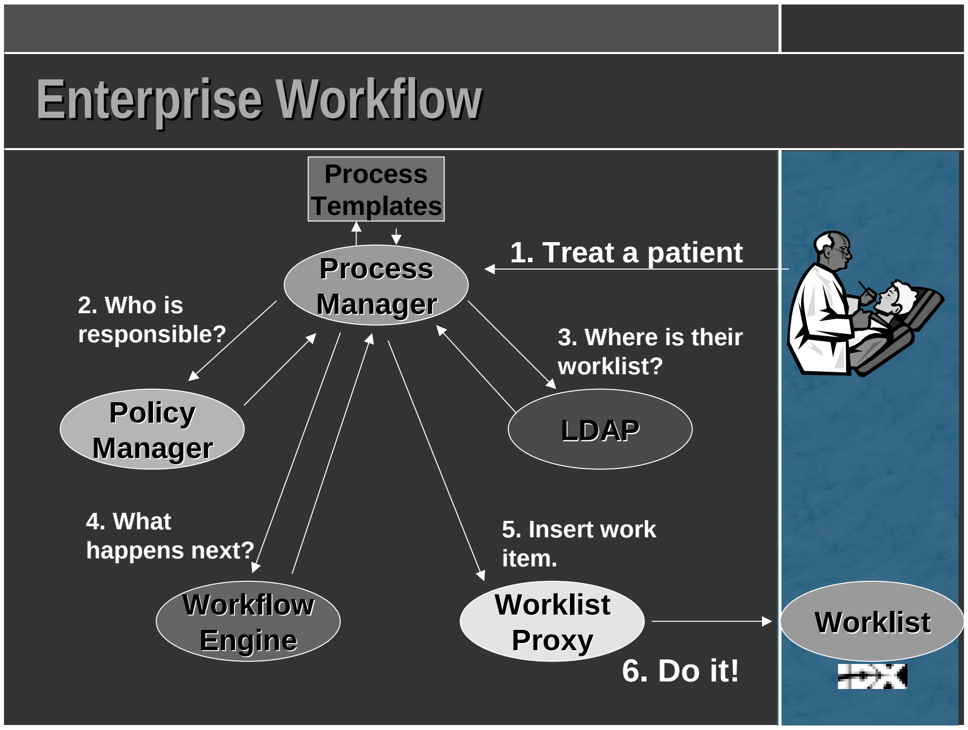
The **PolicyEngine** is responsible for determining who should perform workflow requests. When a **WorkflowEngine WorkList** item does not have a performer assigned, the **WorkflowEngine** asks the **PolicyManager** to assign a **Performer** to appropriate tasks.

An **Observer** observes a set of workflows in the enterprise and is able to report on their current state. An **Observer** is a specialized type of **Performer** and there may be several types of **Observers**:

- A WorkList observes itself.
- An archival Observer.
- A statistical Observer generating summary statistics.
- An Observer waiting for a subsidiary workflow to get done.

A **Performer** is a human or a machine that has a **WorkList** that may be filled with **WorkItems** by one or more **WorkflowEngines**. A **Performer** may delegate pieces of work to another **Performer**, i.e. act like a **WorkflowEngine**.

The entire system can be hierarchically defined as a set of controllers with **Performer** interfaces that operates off of persistent queues, or **WorkLists**. All queues are persistent so that if any queue fails, the system can work around a disabled **Performer** or restart the **Performer** without losing any **WorkItems**.



Summary

Experience has shown that internet standard technologies make it easy to implement “Big Workflow” if:

1. The implementers have a broad background in previous workflow implementations successes and failures.
2. Standards are well understood and implemented by application software, particularly XML messaging.

3. A small, specialized group with deep computer science background does the early prototyping and specification.

Just as workflow can be easily layered over a loosely coupled set of application components, agent technologies can be layered over workflow capabilities reducing the adaptation time of a complex adaptive system from months or years (with programming software interventions) to minutes or seconds in real time. The economic implications are enormous, particularly in healthcare.