# 2

# Scientific versus Business Workflows

Roger Barga and Dennis Gannon

The formal concept of a workflow has existed in the business world for a long time. An entire industry of tools and technology devoted to workflow management has been developed and marketed to meet the needs of commercial enterprises. The Workflow Management Coalition (WfMC) has existed for over ten years and has developed a large set of reference models, documents, and standards. Why has the scientific community not adopted these existing standards? While it is not uncommon for the scientific community to reinvent technology rather than purchase existing solutions, there are issues involved in the technical applications that are unique to science, and we will attempt to characterize some of these here. There are, however, many core concepts that have been developed in the business workflow community that directly relate to science, and we will outline them below.

In 1996, the WfMC defined workflow as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules." [478] While this definition predates the currently evolving models of workflow based on service oriented architectures, it does provide a window on the original workflow concepts, which are based on Business Process Management (BPM). The book, *"Production Workflows: Concepts and Techniques"* by Leymann and Roller [255] provides an excellent overview of the entire field. A business process is an instance of any well-defined task that is often repeated as part of a standard enterprise task. For example, it may be the steps required to complete a purchase order, or it may be related to internal business tasks such as internal audits or corporate database management. Those parts of a business process that relate to the computer automation of business processes are the domain of workflow management.

Leyman and Roller [255] characterize four basic types of workflows encountered in business, and most have direct counterparts in science and engineering. They define collaborative workflows as those that have high business value to the company and involve a single large project and possibly many individuals. For example, the production, promotion, documentation, and re-

lease of a major product fall into this category. The workflow is usually specific to the particular project, but it may follow a standard pattern used by the company. Within the engineering disciplines, this corresponds to the tracking of tasks and subsystem integration required to design and release a new microprocessor. Within the scientific community, it can refer to the management of data produced and distributed on behalf of a large scientific experiment such as those encountered in high-energy physics. Another example may be the end-to-end tracking of the steps required by a biotech enterprise to produce and release a new drug.

The second type of workflow they describe is ad hoc. These activities are less formal in both structure and required response; for example, a notification that a business practice or policy has changed that is broadcast to the entire workforce. Any required action is up to the individual receiving the notification. Within science, notification-driven workflows are common. A good example is an agent process that looks at the output of an instrument. Based on events detected by the instrument, different actions may be required and subworkflow instances may need to be created to deal with them. The third type of workflow is administrative, which refers to enterprise activities such as internal bookkeeping, database management, and maintenance scheduling, that must be done frequently but are not tied directly to the core business of the company. On the other hand, the fourth type of workflow, referred to as production workflow, is involved with those business processes that define core business activities. For example, the steps involved with loan processing are one of the central business processes of a bank. These are tasks that are repeated frequently, and many such workflows may be concurrently processed. Both the administrative and production forms of workflow have obvious counterparts in science and engineering. For example, the routine tasks of managing data coming from instrument streams or verifying that critical monitoring services are running are administrative in nature. Production workflows are those that are run as standard data analyses and simulations by users on a daily basis. For example, doing a severe storm prediction based on current weather conditions within a specific domain or conducting a standard data-mining experiment on a new, large data sample are all central to e-Science workflow practice.

There are however, areas where business workflows seem, at first glance, to be substantially different from their scientific counterparts. For example, a central concern about business workflows is the security and integrity of a sequence of actions. Paying customers demand that when they pay for a service, that service must be guaranteed complete and the results exactly as advertised. Customers demand service. They do not conduct experiments that may or may not succeed. This concept of the integrity of a sequence of actions in embodied in the concept of transaction and is central to understanding workflows in business.

An import class of transactions are those that are long running. Among these long running transactions are those that satisfy the ACID test. An

ACID transaction represents a logical unit of work that is composed of a set of operations. It is an activity that is completed in its entirety or not at all. ACID is an acronym where

- A stands for atomicity, which is an "all or nothing" execution guarantee
- C refers to the fact that the database is always in a *consistent* state
- I means the actions of each transaction are *isolated*, i.e. they are not seen and do not effect other operations that are not part of the transaction
- D is for durability. Once a transaction completes, its effect will survive even if the entire system crashes

The important point of an ACID transaction is that if some subtask fails, the entire transaction can be rolled back so that the entire state of the world is as it was prior to the start of the transaction. And the effect of the transaction is not visible until all subtasks have completed and the entire set of operations is committed. The application of this concept is clear. It is essential that any workflow that carries out the terms of a contract shall either complete the contract or the entire activity is aborted, and that fact is clear to all parties. For example, customers of a bank want to know when they have transferred funds from one account to another that the money was not lost along the way.

Unfortunately, not every workflow can be characterized as an ACID transaction. A long-running workflow is one that may involve many subworkflows each of which is an ACID transaction, but it may not be possible to completely rollback the entire workflow with a single rollback operation. Parts of the workflow may have already completed, and the state of the world may have been altered in various ways. In this case, a failure is something that requires a sequence of new workflows that involve compensating transactions. A typical example of a long-running workflow may involve multiple businesses engaged in a long-running collaboration to produce a product. One company may have been contracted to supply parts to another company producing the final product. The specific details of the interaction with the subcontractor may be governed by one subworkflow. But suppose the subcontractor is unable to deliver the goods. A compensating subworkflow may be to void the original contract and search for a secondary supplier and engage in a negotiation for a replacement service.

Both ACID and long-running workflows have their counterparts in e-Science. The concept of the ACID workflow is essential for any activity that involves building and maintaining a database. and increasingly databases are becoming an essential tool for scientific data analysis. Databases store our collective knowledge in areas such as biological and chemical informatics. Any workflow that could potentially corrupt such a database is one that must be ACID in nature. Long-running workflows also play a role in scientific workflows. A scientist may divide up the overall task into smaller subtasks, each of which can be considered an individual step in the experiment. The results obtained from each such step are either analyzed and/or stored for dissemination to other sites or individuals, used as an input to the next step in an

experiment or exploration, or both. If the scientist later decides an experiment step was faulty, he or she can compensate the subtask, possibly deleting the result and notifying others. Such a together chaining smaller tasks to achieve a desired result from an experiment or exploration, using various data and analysis services, is easily captured as a long-running transaction.

The business workflow industry has had to deal with the increasing complexity of the business processes that have come about because of the distributed nature of enterprises. The corporate information technology landscape has become very heterogeneous. This is a result of many factors, including corporate mergers and piecemeal software and hardware upgrades to different divisions of the company. In addition, there is an increasing need to improve efficiency across the entire organization, and this implies different parts of the organization must work in close alignment. The corporate workflows have to become more corporation-wide.

To address these problems, the workflow industry has been aggressive in its pursuit of technology that improves the time to completion of a workflow design process, reliability of the result, and interoperability across a wide range of platforms. Object-oriented technology has been widely adopted within the industry, and distributed object systems such as the Common Object Request Broker Architecture (CORBA) were a major step forward. The concept of programming by scripting the composition of software components is central to many workflow tools. Leymann and Roller note that to be used as an effective workflow tool, scripts must obey a strict set of rules. For example, it must be possible to interrupt a script at any point and resume its execution later. This implies that the script's state must be saved in a persistent store. Likewise, scripts must be recoverable. If something goes wrong, we should be able to stop the script and roll back any ACID subworkflows and replay the script from a point prior to the failure. It is assumed that the script is orchestrating remotely deployed and executing components and that these components may run in parallel if there are no dependencies preventing it. An important property of any component system is that the implementation technology of the individual components is not exposed. The only thing the script and other components see are interfaces. Leyman and Roller observe that the exploitation of components requires data flow facilities; for example, the input parameters of a component are constructed from the output of several preceding components.

Businesses are also under competitive pressure to rapidly integrate existing applications and business processes to react to changing business conditions. Process integration has always been a challenge and is only complicated further by the fact that business processes today often span multiple internal and external systems. Historically, custom integration solutions have addressed point-to-point integration, in which integration comes at a great cost. The most recent response to the integration challenge is service-oriented architectures (SOAs) [135] and Web service technologies. The promise of SOA is that application components can be assembled with little effort into a network of

loosely coupled services to create a business process that spans organizations and computing platforms. SOA is supported by a range of emerging standards that make it possible to define, implement, and deliver a service in a uniform way so it can be reused in different contexts. The dominant set of standards are those known as WS-*. Included in this set of standards are the Web Service Description Language (WSDL) for service description), the Universal Description, Discovery and Integration (UDDI) protocol for service discovery, the Simple Object Access Protocol (SOAP) for service communication, and the Web Service Business Process Execution Language (WS-BPEL) for workflow.

The essence of SOA lies in independent services that are interconnected with messaging. Each service is a self-contained chunk of code and data that is private to that service, and can be described, published, discovered, orchestrated, and deployed across networks such as the Internet. Services communicate with each other exclusively through messages. No knowledge of the partner service is shared other than the message formats and the sequences of messages that are expected. The bottom-up view of the SOA is that different applications expose their functionalities through Web services. Thus, programmers can access different functionalities of different legacy and newly developed applications in a standard way through Web services.

However, Web services by themselves do not address the need to compose and coordinate a process. WS-BPEL, or BPEL for short, is the de facto standard for the combination and orchestration of Web services. Orchestration, and therefore BPEL, enables a user to specify how existing services should be chained together in various ways to design an executable workflow. The new workflow can then be presented as a new service, which is why BPEL is often described as a language for recursive composition.

BPEL offers a rich language for orchestrating both business and scientific workflows. A BPEL process specifies the exact order in which participating services should be invoked. This can be done sequentially or in parallel. A programmer can express conditional behavior; for example, a Web service invocation can depend on the value of a previous invocation. One can also construct loops, declare variables, copy and assign values, define fault handlers, and so on. By combining all these constructs, the programmer can define a complex scientific experiment in an algorithmic manner. BPEL also provides support for both ACID and long running transactions. Most BPEL implementations can cause the state of a process instance to persist, allowing a user to interrupt a running workflow and reactivate it later when necessary. Moreover, workflows specified in BPEL are fully executable and portable across BPEL-conformant environments, which is an important step toward workflow reuse and exchange.

Today, scientists face many of the same challenges found in enterprise computing, namely integrating distributed and heterogeneous resources. Scientists no longer use just a single machine, or even a single cluster of machines, or a single source of data. Research collaborations are becoming more and more

geographically dispersed and often exploit heterogeneous tools, compare data from different sources, and use machines distributed across several institutions throughout the world. And as the number of scientific resources available on the Internet increases, scientists will increasingly rely on Web technology to perform *in silico* experiments. However, the task of running and coordinating a scientific application across several administrative domains remains extremely complex.

One reason BPEL is an attractive candidate for orchestrating scientific workflows is its strong support for Web services. With scientific resources now available as Web and Grid services, scientists can transition from copying and pasting data through a sequence of Web pages offering those resources to the creation and use of a workflow for experiment design, data analysis, and discovery. Many types of *in silico* genomics analyses, such as promoter identification, start with an initial set of data, perhaps acquired in a more mechanical way such as through fast sequencing equipment or from a microarray chip. This is followed by an ordered sequence of database queries, data transformations, and complex functional, statistical, and other analyses. Such work may require computing power ranging from a desktop computer to a remote supercomputer but is relatively loosely coupled and in many instances asynchronous. By defining a workflow to automatically invoke and analyze more routine parts of the process, multiple data sets can be processed in parallel without requiring a significant amount of additional effort from the scientist and can considerably increase productivity. With the proper tools, scientists with limited programming skills can use BPEL to construct a workflow that carries out an experiment or that retrieves data from remote data services.

There are other advantages to be gained from adapting BPEL for scientific workflows. Since BPEL workflows are designed to act as a Web service, a workflow can be published as a Web service and easily combined with other Web services. Capturing an *in silico* experiment or data transformation as a reusable workflow that can be defined, published, and easily reused is essential in sharing scientific best practice.

Using BPEL to orchestrate an experiment also enables fault tolerance. Because scientists are allowed to select and employ services from a UDDI registry into the workflow, they also have the ability to use an alternative service with similar functionality from the registry in case the original service fails. This ensures that no experiment terminates unexpectedly because of the failure of one particular service in the flow.

Furthermore, a BPEL workflow is specified in terms of service invocations. This allows all aspects of the workflow, such as service execution, message flow, data and process management, fault handling, etc., to be specified as a single integrated process rather than handled separately. The result is a workflow in which each step is explicit, no longer buried in Java or C code. Since the workflow is described in a unified manner, it is much easier to comprehend, providing the opportunity to verify or modify an experiment.

There is a clear case for the role of workflow technology in e-Science; however, there are technical issues unique to science. Business workflows are typically less dynamic and evolving in nature. Scientific workflows tend to change more frequently and may involve very voluminous data translations. In addition, while business workflows tend to be constructed by professional software and business flow engineers, scientific workflows are often constructed by scientists themselves. While they are experts in their domains, they are not necessarily experts in information technology, the software, or the networking in which the tools and workflows operate. Therefore, the two cases may require considerably different interfaces and end-user robustness both during the construction stage of the workflows and during their execution.

In composing a workflow, scientists often incorporate portions of existing workflows, making changes where necessary. Business workflow systems do not currently provide support for storing workflows in a repository and then later searching this repository during workflow composition.

The degree of flexibility that scientists have in their work is usually much higher than in the business domain, where business processes are usually predefined and executed in a routine fashion. Scientific research is exploratory in nature. Scientists carry out experiments, often in a trial-and-error manner wherein they modify the steps of the task to be performed as the experiment proceeds. A scientist may decide to filter a data set coming from a measuring device. Even if such filtering was not originally planned, that is a perfectly acceptable option. The ability to run, pause, revise, and resume a workflow is not exposed in most business workflow systems.

Finally, the control flow found in business workflows may not be expressive enough for highly concurrent workflows and data pipelines found in leading-edge simulation studies. Current BPEL implementations, and indeed most business workflow languages, require the programmer to enumerate all concurrent flows. Scientific workflows may require a new control flow operator to succinctly capture concurrent execution and data flow.

Over the last 20 years, there has been a great deal of interest in both research and industry in systematically defining, reasoning about, and enacting processes and workflows. With so many driving forces at work, it is clear that workflow systems are here to stay and will have a major role to play in the future IT strategies of business and scientific organizations, both large and small. The current focus is on the use of Web services and a move toward a new paradigm of service oriented architecture in which many loosely-coupled Web services are composed and coordinated to carry out a process, and orchestrated using an execution language such as BPEL.

It is genuinely hard to build a robust and scalable orchestration engine and associated authoring tools, and few groups have succeeded in doing so. The emergence of BPEL as the de facto industry standard for Web service orchestration is significant because it means that a number of commercial-grade BPEL engines will be readily available.

The strength of BPEL for orchestrating scientific workflows is its strong support for seamless access to remote resources through Web services. As scientific applications and curated data collections are published as Web services, as will increasingly be the case with the emergence of service-based Grid infrastructures, commercial BPEL engines will be an attractive execution environment for scientific workflows.