1998

Workflow Agents

Michael N. Huhns
University of South Carolina - Columbia, huhns@sc.edu

Munindar P. Singh

Follow this and additional works at: https://scholarcommons.sc.edu/csce_facpub

Part of the Computer Engineering Commons

Publication Info
http://ieeexplore.ieee.org/servlet/opac?punumber=4236
© 1998 by the Institute of Electrical and Electronics Engineers (IEEE)
When you use a computer at work, chances are you're engaged in a workflow. Even when you're doing non-computer work—for example, ordering parts for your project or factory by telephone, or calling an airline or a government office—you are most likely participating unwittingly in a workflow.

A workflow is a composite activity consisting of tasks involving a number of humans, databases, and specialized applications. The component tasks are related and share various control, data, and temporal dependencies. A classic example of a workflow is loan processing: When you apply for a loan, you fill out a form, a clerk reviews it for completeness, an auditor verifies the information, and a supervisor invokes an external credit agency or uses a credit risk assessment tool. Each person in the loan process receives information concerning your application, modifies or adds to it, and forwards the results.

Another example, illustrated in Figure 1, is when you order a service from a telecommunications provider. You initiate the order by interacting with a sales representative from the provider, who fills out a form on your behalf. The sales representative checks with a provisioning database to determine whether the necessary hardware is in place. If it is, you receive an estimate of when the service will be ready for your use. A local service installer is dispatched to install your service while the telecommunications provider checks your credit history.

If all goes well, the installer successfully installs the service, the auditors find your credit history acceptable, the billing department is notified to begin charging you, and the workflow concludes successfully.

**Murphy's Law**

However, things don't always go that smoothly. For example, in checking whether you already have an account, the telecommunications provider might discover that you have an unpaid and overdue balance—or that someone else previously at the same address has an unpaid balance. Such discoveries would raise a red flag.

Perhaps the service installer for your area calls in sick, requiring a revision in the installation schedule. Or the installer might discover that the available hardware is unusable and must be replaced. Each of these situations can lead to modified behavior, as illustrated in Figure 2. Such modifications might lead to an additional change in schedule or possibly even cause you to cancel the order altogether because you don't want to wait indefinitely.

These occurrences are instances of exceptions that can arise during workflow execution. The number of possible exceptions is very large; their scope and the great variety of possible contexts make it practically impossible to specify all exceptions statically and in advance. Unfortunately, the only sure thing about exceptions is that they are far from exceptional. As a consequence, most natural workflows are inherently incomplete.

Exceptions differ from a simple alternative flow of control; indeed, the two are conceptually distinct. Attempting to include all exceptions is not only futile but would also clutter the workflow so much as to render it incomprehensible. For the same reasons that programming languages such as Java treat exceptions separately, we prefer to think of exceptions as parasitic on the main workflow. Of course, if some exceptions occur often enough to become almost routine, they will be incorporated as
explicit alternatives within the workflow, as illustrated in Figure 3.

State of the Art
Workflow technology is important to network computing because workflows exist naturally wherever distributed resources are interrelated. Currently, most workflows arise in intranets, although multienterprise Internet workflows are emerging in applications such as electronic commerce.

There are many workflow tools—at least 100, and by some counts as many as 250. Each tool provides some type of process-modeling mechanism coupled with an execution framework. In general, the metamodels underlying most workflow tools are based on a variant of activity networks, which show different activities as nodes and use links to represent various temporal and exception dependencies among the nodes. Figures 1–3 reflect this general idea.

System analysts design workflows on the basis of their understanding of the given organization and the abstractions the chosen workflow tool supports. Once designed, the workflow can be executed automatically by the tool. This can result in improved efficiency. For example, when workflows involve human workers, the workers can be automatically informed of the tasks they should be performing.

Challenges Facing Workflow Technology
Workflow technology is not universally acclaimed, and many CIOs are not convinced of its capabilities and benefits. One problem is that current workflow technology is often too rigid. Because workflows are constructed prior to use and are enforced by some central authority, this rigidity is inevitable. However, the lack of freedom accorded to human participants causes workflow management systems to appear unfriendly. As a result, they are often ignored or circumvented, and eventually discarded.

This rigidity also causes productivity losses by making it harder to accommodate the flexible, ad hoc reasoning that is the strong suit of human intelligence. This need for flexibility is most apparent when an exception occurs and rigid workflow management tools behave incorrectly. In our earlier example, if the credit bureau is unresponsive, a poorly designed workflow might just hang, whereas a flexible one would let a human make a decision based on available information.

Another challenge is that system requirements are rarely static. A workflow's design context might not remain applicable in every detail over the workflow's lifetime. Dynamic requirements can necessitate arbitrary extensions not recorded in the workflow model itself. Suppose our telecommunications provider makes a special offer at the start of an academic year whereby it waives credit-history checks of full-time students. Would this change require the workflow to be redesigned and reinstalled?

Agents for Workflow
As natural loci of autonomy and decision, agents promise to address these challenges. They perceive, reason about, and affect their environment. They can be designed to be adaptive and communicative.

Agents in an information environment can play a number of distinct roles. The roles of greatest interest to a workflow setting are user agents, resource agents, and brokers. When a workflow is constituted in terms of distinct roles that agents can instantiate, the agents can be set up to respect the constraints of their users and resources. Being aware of their local situation enables agents to adapt to a workflow. User agents negotiate with one another and with resource agents to ensure that global constraints are not violated and that global efficiencies can be achieved.

Agents can include functionality to identify different kinds of exception conditions and react appropriately, possibly by negotiating a special sequence of actions. More importantly, agents can learn from repeated instances of the same kinds of exceptions. With this learning ability, agents can process the updated set of constraints when system requirements change.

Countering ACIDity. Workflow agents can implement a form of relaxed transaction processing. Relaxed or extended transactions are activities consisting of several tasks, or operations, that do not satisfy one or more of the ACID properties.

- Atomicity means that either all changes a task causes to a system state happen or none do.
- Consistency means that a task takes the system from one consistent state to another.
- Isolation means that a task's intermediate results are not visible to another task.
- Durability means that any changes committed by a task persist.

ACID transactions are the staple of traditional databases because they
are impractical, because they require reasonable behaviors. Moreover, an order was received only if it was processed—or worse, that the order was received until after it had been processed. The model describes how workflow engines ought to be connected to applications and databases. Agents can contribute to achieving interoperability among the different resources while satisfying their local constraints.

Another, more profound, kind of interoperability occurs among different workflows. A workflow represents a meaningful unit of processing that affects a number of people and information resources. Clearly, multiple units must interact with each other, because some people participate in more than one, and the units inevitably share resources. Workflow designers must understand, model, and manage these interactions properly. If they don’t, all manner of chaos may ensue—and indeed often does. For example, one workflow of our workflow technology. The model describes how workflow engines ought to be connected to applications and databases. Agents can contribute to achieving interoperability among the different resources while satisfying their local constraints.

REFERENCE