1998

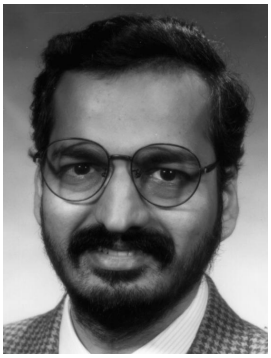
Workflow Agents

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

Munindar P. Singh

Publication Info

Published in *IEEE Internet Computing*, Volume 2, Issue 4, 1998, pages 94-96.



WORKFLOW AGENTS

Michael N. Huhns • *University of South Carolina • huhns@sc.edu*
Munindar P. Singh • *North Carolina State University • singh@ncsu.edu*

When you use a computer at work, chances are you're engaged in a workflow. Even when you're doing noncomputer 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

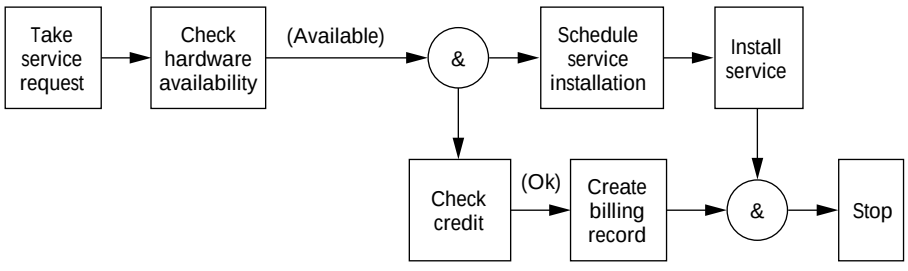


Figure 1. A workflow for processing a telecommunications service order.

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.[1] 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.[2] 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.[3] 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.[4] Relaxed or extended transactions are activities consisting of several tasks, or operations, that do not satisfy one or more of the ACID properties.[5]

- *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

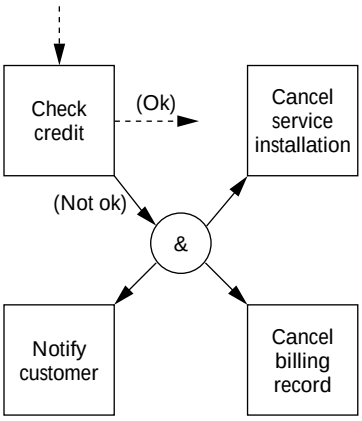


Figure 2. Exceptions—unexpected occurrences that interrupt and possibly alter a workflow—can arise during workflow execution.

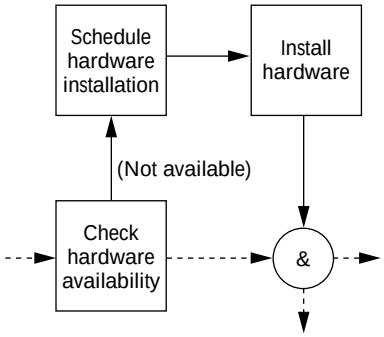


Figure 3. An exception that occurs often enough to be considered routine can be incorporated into the workflow as an alternative flow of control.

URLs for this column

ITESM Monterrey's CORREA project (in Spanish) • www-cia.mty.itesm.mx/~rbrena/CORREA/CORREA.html
NIIIP consortium • www.niiip.org/
Reference model for workflow management • www.aiim.org/wfmc/DOCS/refmodel/rmv1-16.html
Simple Workflow Access Protocol • www.people.netscape.com/kswenson/SWAP/
University of Georgia Large-Scale Distributed Information Systems Laboratory • www.lsdis.cs.uga.edu/demos/workflowindex.html
Workflow Management Coalition • www.capv.com/dss/resources/glossary/2282_13a.htm

guarantee that only consistent data is stored and that only consistent snapshots of a changing database are viewed. However, implementing ACID transactions makes stringent demands that cannot be met in an open environment, such as the Internet. For example, if the workflow in our figures were modeled as an ACID transaction, we would have to ensure that the user couldn't be told the order was received until after it had been processed—or worse, that an order was received only if it was completed. Of course, these are not reasonable behaviors. Moreover, they are impractical, because they require delaying one task until another task, which might not occur until much later, catches up.

So without transactions how can we ensure consistency? Resource agents working in conjunction with user agents can contribute to a solution. By keeping track of their interactions and how the stored data is being accessed and updated, these agents can help maintain overall system consistency. They do not do this in the lockstep manner of an ACID transaction, but they can ensure consistency at intervals sufficient for the particular workflow. By describing at a high level how different components of a workflow ought to be treated, relaxed transactions serve as the basis for designing the behavior of such agents. However, additional functionalities, such as negotiation, are necessary.

Interoperation. A workflow represents the interoperation of several applications and databases. This interoperation can be achieved by implementing an appropriate workflow from scratch. However, recent standards activities, chiefly led by the **Workflow Management Coalition**, attempt to define a **reference model for workflow management**. The model describes how workflow engines ought to be connected to applications and databases. Agents can contribute to achieving interoperation among the different resources while satisfying their local constraints.

Another, more profound, kind of interoperation 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 communications provider might be upgrading communications wiring with a view to discarding the old wiring, while another workflow might treat the old wiring as freely available and assign new telephone circuits to it.

We can view a workflow itself as a resource and then associate workflow agents, acting as resource agents, with it. Workflow agents can coordinate the workflows they manage and thereby provide for larger, possibly enterprise-wide, workflows. This requires an ability to communicate and negotiate. Such coordination benefits from standards that enable workflows modeled and managed by tools from different vendors to be related. One example is the recently announced **Simple Workflow Access Protocol** (SWAP).

In the future, much as they enable databases to interoperate today, agents will enable Internet-wide workflows to be coordinated and executed.

Systems of the Bimonth

A number of interesting projects involve agents in workflow, but many of them have proprietary details. However, you should check out the following projects on the Web.

- The **NIIIP consortium** is applying agents in workflows in the manufacturing domain.
- Agents for bureaucratic assistance, developed at the Center for Computing Research, National Polytechnic Institute, Mexico City, do not enact a whole workflow, but they capture a component common to several workflows. They essentially help people in a distributed organization fill out forms to satisfy various internal or external needs.
- **ITESM Monterrey's CORREA project** focuses on agents for collaboration.
- The **University of Georgia Large-Scale Distributed Information Systems Laboratory** is studying workflow technology. ■

REFERENCES

1. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, http://lsdis.cs.uga.edu/activities/NSF-workflow.
2. M.N. Huhns and M.P. Singh, "Ontologies for Agents," *IEEE Internet Computing*, Vol. 1, No. 6, Nov./Dec. 1997, pp. 81–83.
3. M.N. Huhns and M.P. Singh, "All Agents Are Not Created Equal," *IEEE Internet Computing*, Vol. 2, No. 3, May/June 1998, pp.94–96.
4. M.P. Singh and M.N. Huhns, "Automating Workflows for Service Order Processing: Integrating AI and Database Technologies," *IEEE Expert*, Vol. 9, No. 5, Oct. 1994, pp. 19–23.
5. J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, San Mateo, Calif., 1993.