Multiagent Systems for Workflow

Munindar P. Singh
Computer Science
North Carolina State University
Raleigh, NC 27695-7534, USA

`singh@ncsu.edu`

Michael N. Huhns
Electrical & Computer Engg.
University of South Carolina
Columbia, SC 29208, USA

`huhns@sc.edu`

Abstract

Workflows are ubiquitous in business computing. They arise not only within an enterprise, but increasingly across enterprises as well—in situations such as virtual enterprises and applications such as supply-chain management. Although the importance of workflows as a basis for understanding and automating business activities is widely recognized, current workflow practice leaves much to be desired. To a large extent, this problem arises because of the rigidity of current technology, which does not accord well with the complex, heterogeneous, dynamic environments in which workflows are applied. Agent technology promises to alleviate many of these problems and hence enable adaptive workflows in realistic settings. We consider interaction-oriented programming (IOP), an approach to software engineering based on multiagent systems that we have been developing. We focus on one aspect of IOP, which deals with social commitments and enables agents to flexibly enact a multienterprise workflow by entering into and behaving according to their commitments to each other. The agents can cancel or modify their base-level commitments only if they satisfy the metacommitments that then go into effect.

1 Introduction

The expansion of the computing and communications infrastructure has brought home the problem of the "islands of automation" that occur in traditional information systems in large enterprises. Such problems occur with redoubled strength in modern information systems, which in effect—if not by design—span across enterprise boundaries. Such systems arise directly in the case of virtual enterprises, but indirectly also in applications such as electronic commerce and supply-chain management. Such applications are garnering increasing attention among both practitioners and researchers. As a consequence, technologies such as workflow management have garnered much interest, but also hype.

To motivate this paper, we begin with an informal definition of workflows. A workflow is a distributed multitask activity, routinized or systematized in some way, that involves the coordinated execution of human and system tasks, usually in heterogeneous environments. This definition is in agreement with the folklore, e.g., see Georgakopoulos *et al.*'s survey [8], and a special issue on the subject [11].

Workflows are commonly understood to have certain key features, especially including the following. One, in consisting of a number of tasks, they are *composite*. Two, they are structurally and semantically *complex* in that they—and possibly some of their component activities—are long-running and failure-prone; they frequently update multiple data items across a number of resources; and, their components activities can have subtle consistency requirements. Three, workflows are often *cooperative*, meaning that they not only involve human interaction, but also involve back-and-forth interactions among their constituent activities. Four, workflows by their very nature arise in *heterogeneous* environments with unchangeable "legacy" components. Five, the components of workflows may be of *autonomous* ownership and not fully under the control of the workflow.

In light of this, it is helpful to think of workflows as the dynamic components of open information environments, the static components being the information repositories and ontologies.

1.1 Generations of Workflow Technology

Workflows have been with us from the dawn of time—ever since there have been "business" organizations or governments of some form or the other that have cared to systematize any of their activities. Therefore, in understanding the expected development of workflows, it is important to consider the major generations of workflow technology.

- *1st: Manual.* Bureaucracies have long been part of human society, and they typically involve the processing of information. Before there were computational aids for information processing, it was carried out manually. Although slow, this had the advantage of involving humans in every stage, thereby facilitating the handling of exception conditions and making modifications to the workflow as the needs of the organization evolved.

- *2nd: Closed.* The early days of business computing involved data processing applied in information management. Often this was based on straightforwardly automating existing manual processes. There was tight coupling between business objects and control information, thereby making system evolution labor-intensive and difficult.
- *3rd: Database-centric.* The development of database technology enabled open specification of business information. There was a certain amount of decoupling between data and process. However, control information remained hard-coded in procedures.
- *4th: Current tools.* Current workflow tools provide the separation of control from application. Processes are thus viewed at two levels of granularity: (a) as units of work that are composed together through a workflow tool, and (b) as implementations of those units of work via specific applications. However, the workflows themselves still prove complicated for both design and redesign. Also, they provide little support for handling exceptions.
- *5th: Agent-based.* This is the generation of workflow technology being promoted here. This generation is emerging through the use of agents and multiagent systems. We will describe some of its components below.

1.2 Themes in Workflow

The strong industry interest in workflow technology has led to a variety of descriptions of what workflows are. Often these are implicit in the definitions or the technical problems addressed by an approach. We identify the following most reasonable ones.

- *Form management and flow.* This applies to the most traditional organizations, which are essentially doing business in a manner closest to that of purely paper-based organizations—sometimes like the same organization used to, if it is old enough. However, these organizations use new technology to put forms on-line (imaging applications) and distribute and disseminate them on-line (transmittal applications). Much of the current workflow market is about form management. Sometimes, the term "document" may be used instead of "form," but more general documents are rarely intended in this category of approaches.
- *Groupware.* This is most appropriate for addressing the human collaboration aspect of an organization. Groupware tools can handle some components of organization modeling. Some of these tools are about documents, and take a broader view of documents than in the above. They help in the creation, dissemination, and maintenance of (versions of) documents even as users work on them concurrently.
- *Control logic specifications.* This view treats the workflow as primarily consisting of orchestrating the application activities that underlie it [4]. In this view, the workflow is specified by showing how the different activities are to be wired together. Frequently, these specifications

are given in a graphical activity modeling language, such as flow charts or activity diagrams [6]. The details of the individual activities are not specified. In some variants, the data flow may be explicitly shown as well.

- *Distributed programs.* Sometimes, virtually any distributed program may be referred to as a workflow. This is not entirely unreasonable if the program is modeled with some workflow metamodel. However, sometimes the term is used loosely, in which case it has little meaning; if every distributed program can be a workflow, then the term "workflow" is unnecessary.

- *Transactions.* An interesting view of workflow treats them as transactions of some form: traditional or extended. This view builds on the database basis of most information environments. It is most suitable when the integrity of the stored data is given primacy [1, 3, 5, 20].

- *Coherent computations.* The view we promote in this paper is that workflows are *coherent*. By coherent, we mean that the components tasks of a workflow are selected and ordered to ensure the coherence of the entire workflow. In this view, the consistency of data items is important only to the extent that it helps ensure that the behavior is coherent; the behavior may often be coherent when data consistency is lost, e.g., by informing a user or sending out retractions for previous results.

Each of the previous views of workflow is suitable for some class of applications and environments either because it is designed for specific applications, or because they support certain properties (such as data integrity), which are of primary value only for some applications. The groupware view emphasizes the human aspects of any organization and is of special value in modeling important aspects of practical workflows, especially how people may participate in performing a shared task. Although somewhat restrictive in scope, the transactional view is supported by sound computational abstractions. The control logic view is able to accommodate the computational aspects of any other view, especially of any extended transaction model. However, it takes a lower-level stance than is often appropriate for modeling.

The technical challenge is to synthesize the considerations behind the above views into a more comprehensive and powerful view. This is the basis for our ongoing research program, whose results we describe below.

1.3 Agents

Agents are persistent active entities that can perceive, reason, and act in their environment, and communicate with other agents. Often, the agents are autonomous, intelligent, and sociable. What makes agents interesting for our purposes is that they can form multiagent systems. Agents are autonomous, but in order to form and participate in multiagent systems, they must be able to compromise on their autonomy somewhat—just so they can coordinate with others. The agents in a multiagent system would often be heterogeneous.

A variety of abstractions for agents have been proposed. These include those inspired from folk psychology, such as beliefs, knowledge, and intentions, and those inspired from organizations and societies, such as commitments and teams. Both kinds are appropriate in general. For our purposes, the latter are more relevant. Of course, even an entire multiagent system or team may be treated as if it were a monolithic agent. Viewed in this light, agents are structured—this accords well with hierarchical decomposition, which is a common theme in the analysis and design of complex systems.

1.4 Cooperative Information Systems

Cooperative Information Systems (CISs) are multiagent systems with organizational and database abstractions geared to open environments. Typically, a CIS includes an environment consisting of a variety of related information resources. A CIS also includes some means of attaching semantics to its resources and ways to view and update those resources in a manner that respects the semantics. Because CISs are based on multiagent systems, they are *open* in admitting new resources, *flexible* in allowing the resources to evolve, *intelligent* in ensuring valid states and coherent behaviors despite complex constraints, and *adaptive* in adjusting their behavior to accommodate unexpected changes in their environment.

We recast workflows in terms of CISs simply by defining workflows as the dynamic aspects of CISs. More precisely, a workflow is a well-defined specification of some coherent class of behaviors of a CIS. The coherence requirements are captured during modeling, can evolve, and provide a basis for the control logic and execution. Our claim is that by introducing the "right" high-level abstractions, multiagent systems can capture workflows better than conventional workflow technology. The rest of this paper is how we might achieve this with an appropriate combination of rigor and flexibility.

Organization. Section 2 reviews the traditional abstractions for describing CISs and analyzing, specifying, and realizing workflows. Section 3 shows how we may apply agent and multiagent techniques to address the above challenges. It also discusses a specific example in some detail. Section 4 concludes with a discussion of the important issues.

2 Abstractions

We consider the modeling and computational abstractions used to capture different components or aspects of an information system. The dynamic ones such as transaction models have immediate relevance for workflows, but even the static ones such as conceptual and organizational models relate to workflows, because ultimately any effective workflow must deal with those aspects as well.

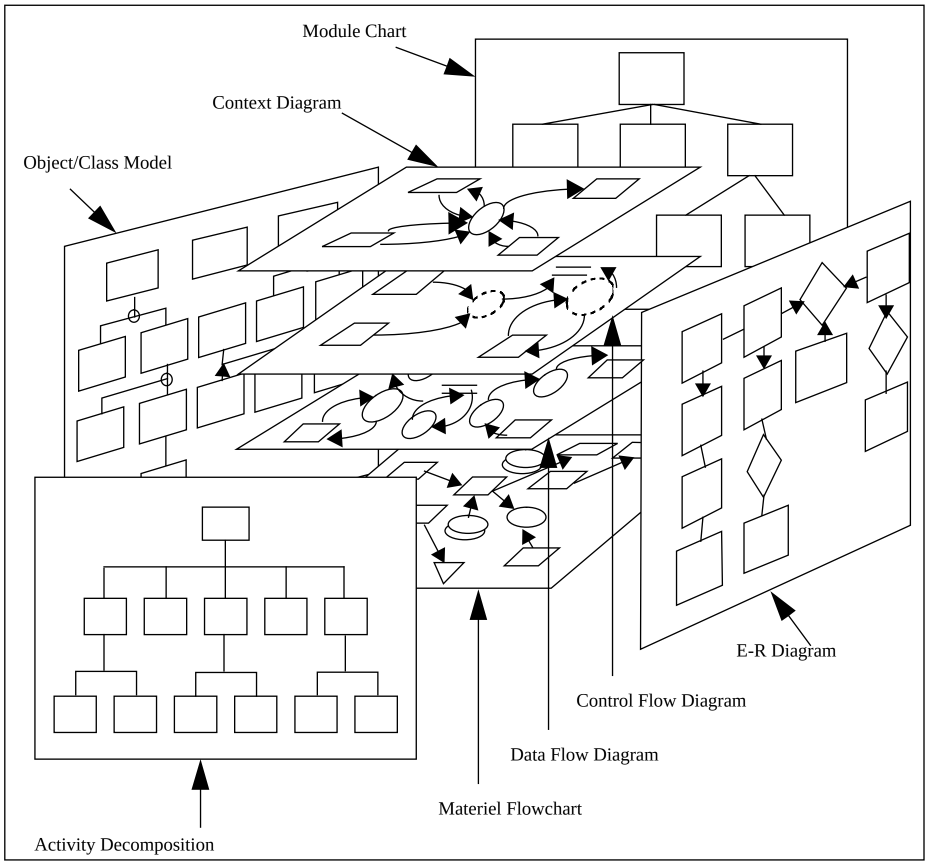

Figure 1: Metamodels Used to Describe a CIS

For CISs applied to enterprises or virtual enterprises, a variety of models are typically built. Figure 1 shows some of the common modeling approaches. Of the main ones, entity-relationship (E-R) diagrams describe a conceptual model of the information stored in (a subset of the databases in) the enterprise. Activity decomposition describes the relationship of inclusion among different activities, whereas the control, data, and materiel flows give additional information about it. E-R diagrams correspond to static information as in ontologies; the activity representations correspond loosely to the workflows. It is important to relate the two categories of representations, because the actions in the workflows depend on the concepts they manipulate, and the concepts are defined based on their patterns of usage.

2.1 Relating Models

In a number of settings, including enterprises, the organizational structure of a CIS is important. By the organizational structure, we mean the set of roles and responsibilities that make up a functioning system [7, 19]. There is an intimate relationship between the workflows executing in a CIS, and the organizational roles available in it. Figure 2 shows on the left a simple workflow corresponding to submitting a contract proposal from a company. The *write white paper* task itself may be decomposed into a subworkflow. The bottom left shows a possible subworkflow for travel. The tasks in the workflow impinge upon various databases, and other ongoing processes, such as *budget forecast*. They also relate to the organizational structure of the company, because key steps in the workflow must be performed by people with specific authorities.

Traditionally, the roles are mapped to tasks rigidly. However, in open and dynamic environments, more flexible role-bindings are needed. For example, if the *research director* is on leave, how may the workflow be rerouted? If one person fills multiple roles, how may the workflow be scheduled to optimize their time? Another, more important, issue is how the obligations of an organization be mapped to the obligations—and hence actions—of its members. And, how can be decisions of a member be over-ruled or undone when necessary; conversely, how may a participant obtain the necessary exceptions to some default policy in order to respond properly to an unexpected situation.

2.2 Transactions

Computations are of two main kinds: (a) *terminating*: these include traditional queries and transactions as well as application programs, and (b) *nonterminating or repeating:* these include information flows as well as administrative activities. Traditional database transactions are terminating computations that satisfy the so-called ACID properties [10], which describe next.

- *Atomicity:* all or none of a transaction happens
- *Consistency:* a transaction preserves the consistency of the database

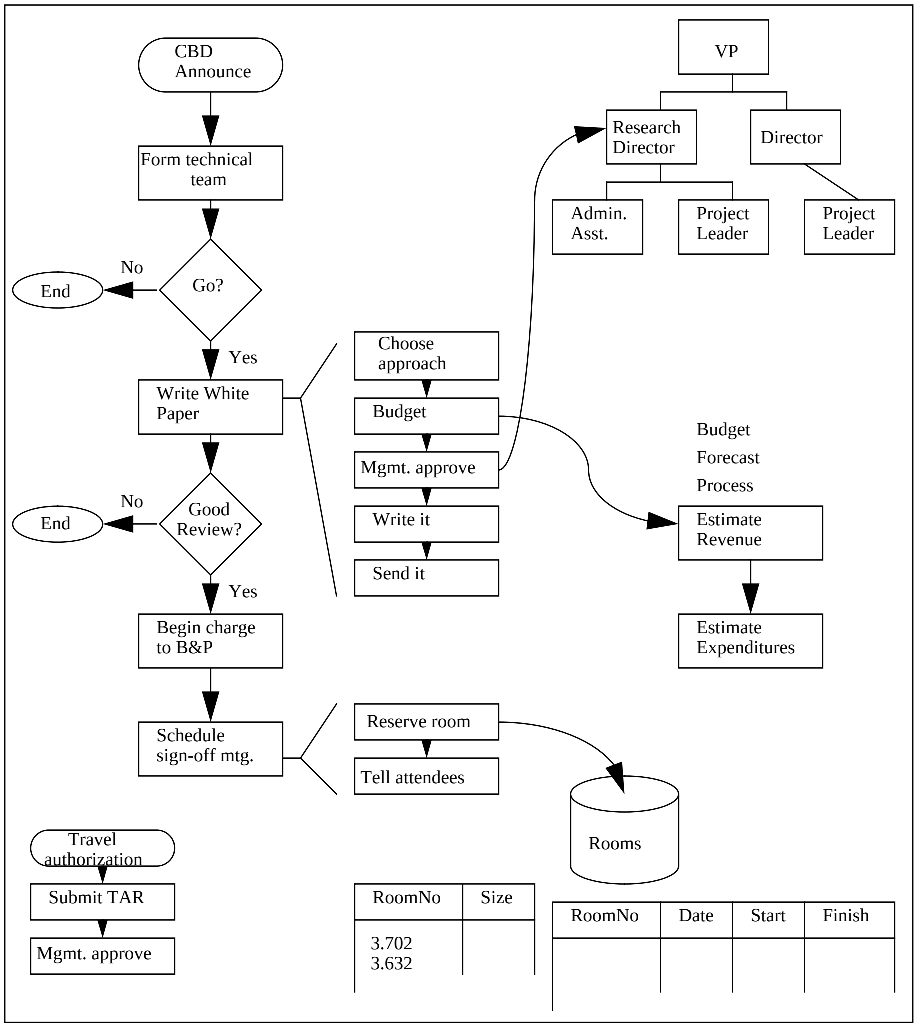

Figure 2: Relating Different Models

- *Isolation:* intermediate results of a transaction are not visible to any other transaction
- *Durability:* when a transaction concludes successfully, its effects are permanent.

If the individual transactions are programmed correctly, the system guarantees consistency for any arbitrary concurrent mix of transactions. Atomicity is essential to ensure that the integrity of distributed data is preserved. Consequently, the actions or subtransactions that constitute a transaction must either (a) all happen, thereby transforming the database from a consistent state to a new consistent state, or (b) each fail to happen, thereby leaving the database in its original (consistent) state. Ensuring the ACID properties requires locking all the data items accessed by a transaction until it completes (or achieving the same effect through an another, more optimistic, approach). Practically, this means that ACID transactions are limited to activities that are brief (at most seconds) and simple (few updates), and usually in homogeneous or centralized information environments.

The above difficulty has led to a number of extended transaction models (ETMs), which relax the ACID properties in various ways. ETMs embody some valuable intuitions about structuring activities, but are themselves not practical either. They usually assume that (a) compensating actions are defined for some of the subtransactions, and (b) it is acceptable to allow temporary inconsistencies. Without good conceptual models to back these relaxations up, they may easily be unrealistic or unsound. Further, ETMs are difficult to specify and schedule. Further, they retain a focus on data integrity, whereas the real challenge is to allow activities that are coherent, not necessarily consistency-preserving.

2.3 Speech Acts

Another class of abstractions is based on speech acts. The best known of these is formalized in the ActionWorkflow product [18], which builds on the theory of speech acts due to Winograd & Flores [27]. Like other theories of speech acts, Winograd & Flores' theory treats language as action. However, their theory focuses on the roles played by different speech acts in the progress of a conversation. A completed conversation with all nested subconversations thus constitutes a workflow. This has inspired the "loops" metamodel for workflows used in ActionWorkflow.

In this metamodel, each loop represents an exchange between two actors: a *customer* and a *performer*. The loop consists of four steps: (a) a request from the customer to the performer, (b) negotiation by the two to determine what the performer should do, (c) actual performance of the negotiated task by the performer, and (d) evaluation of the performance by the customer. The four steps close the loop. A step may potentially be nested with other loops and involving other customer-performer relationships.

We find the idea of taking the perspectives of the customer and the performer both into account attractive—traditional workflow specifications usually take one or the other perspective. However, the metamodel has some limitations. It only considers two actors at a time, and does not explicitly consider the surrounding organizational structure. Because it does not include explicit representation of and reasoning about commitments, it does not easily accommodate modifications in the commitments, e.g., if an actor wishes to cancel a commitment or modify it in some way. After an interesting top-level structure, the approach rapidly reduces to a traditional activity network style specification. There are some other critiques of the speech acts approach, e.g., by Ljungberg & Holm [17].

Still, it has some important insights, which we incorporate in our approach.

3 Technical Approach

Through the many interesting features that they possess, agents provide autonomy and heterogeneity, constrain access to resources and guarantee specialized integrity requirements, model organizations and nonterminating tasks in them. Moreover, they can create "mini-societies" corresponding to different business processes, but retain responsibility for resolving conflicts among different processes. Consequently, agents are best applied to achieving flexibility and agility, improving efficiency of processes, and helping manage complexity.

3.1 Interaction-Oriented Programming

Merely using the terms "agent" or "multiagent" to describe a system would not ameliorate our problems. We also need specific solutions based on agents through which the challenges of workflow might be addressed. We define *Interaction-Oriented Programming (IOP)* as a collection of techniques centered around the notion of interaction. As indicated in Section 1.3, the ability to interact flexibly is the most important feature that agents can have. But this feature—which maps to a set of related abstractions techniques—is also key in workflow management as envisioned here.

Key issues include the autonomy and heterogeneity of agents, the flexibility and robustness of the multiagent system, and the assurance of properties of the resulting CIS. Accordingly, IOP involves high-level primitives for interactions, which synthesize insights from databases and distributed AI.

Our research program on IOP is developing primitives for the specification of systems of agents and constraints on their behavior. Distinct primitives are being studied for the three layers of IOP: (a) coordination [22], (b) commitment [24], and (c) collaboration [23]. Here we focus primarily on the commitment layer. This includes primitives such as societies, the roles agents may play in them, what capabilities and commitments they require, and what authorities they grant. Agents can autonomously instantiate abstract societies by adopting roles in them. The creation, operation, and dissolution of societies are achieved by agents acting autonomously, but satisfying their commitments. A commitment can be canceled, provided the agent then satisfies the metacommitments applying to its cancelation.

The representations for IOP must support several functionalities, which typically exist informally, and are either effected by humans in some unprincipled way, are hard-coded in applications, or are buried in operating procedures and manuals. Information typically exists in data stores, in the environment, or with interacting entities. The IOP contribution is that it (a) enhances and formalizes ideas from different disciplines, (b) separates them out in an explicit conceptual meta-model to use as a basis for programming and for programming methodologies, and (c) makes them programmable.

The notion of commitments may be familiar from databases. However, in databases, commitments correspond to a value being declared and are identified with the successful termination of a transaction. When a transaction terminates successfully, it commits, but it is not around any more to modify its commitments. Thus the commitments are rigid and irrevocable. If the data value committed by one transaction must be modified, a separate, logically independent transaction must be executed to commit the modified value. Traditional commitments presuppose that different computations are fully isolated and that locks can be held long enough that the atomicity of distributed computations can be assured. Although suitable for traditional data processing, for the above reasons, traditional commitments are highly undesirable for information-rich environments, where autonomous entities must carry out prolonged interactions with one another [21].

Commitments reflect an inherent tension between predictability and flexibility. Agents who can commit are easier to deal with. Also, the desired commitments serve as a sort of requirements on the construction of the agents who meet those commitments. However, commitments reduce the options available to an agent.

3.2 Commitments

We propose an alternative characterization of commitments that is better suited to agents and multiagent systems. In our formulation the commitments are directed to specific parties in a specific context. Thus an agent may not offer the same commitments to every other agent. The context is the multiagent system within which the given agents interact. An agent or multiagent system with jurisdiction over some resources and agents is called a *sphere of commitment (SoCom)*.

A commitment is a four-place relation. The *debtor* refers to the agent who makes the commitment, and the *creditor* to the agent who receives the commitment. Commitments are formed in a *context*, which is given by the enclosing SoCom (or, ultimately, by society at large). Based on the above intuitions, we motivate the following logical form for commitments.

> A commitment $\mathsf{C}(x, y, p, G)$ relates a debtor x, a creditor y, a context G, and a discharge condition p.

We define some useful operations on commitments, which capture how they are created, satisfied, canceled, delegated to or acquired from another party, or released. We can specify constraints on when any of these actions may or must be performed. This enables us to capture policies such as what an agent must do if he cancels a commitment to deliver some goods or if he retracts his claim about the validity of some data item. Some of the theoretical aspects of commitments are elaborated elsewhere [24].

3.3 Commitments for Coherence

Commitments are computationally applied in the following manner. Initially, abstract SoComs are defined in terms of their *roles*. Each role is associated with the capabilities it requires, the commitments it engenders, and the authorities it creates. The capabilities are the tasks the agent *can* do, the commitments are what the agent *must* do, and the authorities are what the agent *may* do. The commitments, in particular, may be metacommitments. Indeed, they usually are metacommitments, e.g., that the agent will adopt a base commitment upon receiving a request.

At some point, during configuration or execution, an agent may decide to enter into a SoCom as a particular role or roles. To do so, he would have to cause the SoCom to be instantiated from the abstract specification. To adopt a role, the agent must have the necessary capabilities, and accept the associated commitments. In doing so, he also obtains the authorities to properly play the role. The agent must then behave according to the commitments. Agents can join a SoCom when configured by humans or during execution: this requires publishing the definition of the abstract SoCom.

We consider an example in two parts. The first deals with electronic commerce; the second combines in aspects of virtual enterprises [13]. The commitments are designed based on the corresponding roles in human society.

3.3.1 Electronic Commerce

We first define an abstract SoCom consisting of two roles: *buyer* and *seller*, which require capabilities and commitments about, e.g., the requests they will honor, and the validity of price quotes. To adopt these roles, agents must have the capabilities and acquire the commitments. Example 1 involves two individual agents who adopt the roles of *Buyer* and *Seller* to carry out a simple deal.

Example 1 Consider a situation involving two agents, *Customer* and *Vendor*, with authority over their respective databases. The SoCom manager has an abstract SoCom for buy-sell deals with the roles of *Buyer* and *Seller*. *Buyer*'s capabilities include asking for a price quote and placing an order. *Seller*'s capabilities include responding to price quotes and accepting orders based on checking the inventory locally. *Buyer*'s commitments include paying the quoted price for anything she orders. *Seller*'s commitments include (a) giving price quotes in response to requests and (b) fulfilling orders that he has accepted.

Customer asks the manager to instantiate a deal between her (*Customer*) as *Buyer* and *Vendor* as *Seller*. The manager asks *Vendor* if he would like to join as *Seller*. When *Vendor* agrees, and since both agents have the requisite capabilities, capacities, and resources, the deal is set up.

Customer now wishes to check the price of a valve with a diameter of 21mm. Upon the receipt of the query from *Customer*, *Vendor*—based on its role as *Seller*—offers an appropriate answer. ∎

3.3.2 Virtual Enterprises

Example 2 considers a more general situation where the role of *Seller* is adopted by an agent who happens to be a Valvano-cum-Hoosier virtual enterprise (VE)—i.e., a SoCom consisting of the hose and valve vendors. Example 3 considers the situation where the Valvano-cum-Hoosier VE detects a problem in the supply of valves for which an order has been placed. The VE automatically meets its commitments by revising the order and notifying the customer.

Now we consider the situation where one or more agents may form a cooperative SoCom or team. For simplicity, we assume that teams have a distinguished agent who handles their external interactions. We refer to this agent as the VE.

Example 2 We now consider two agents with authority over the Valvano and Hoosier databases, respectively. These agents have similar capabilities to the *Seller* of Example 1. They form a VE, called Valvano-cum-Hoosier VE, which can adopt the role of *Seller*. *Buyer* behaves as before and expects *Seller* to behave according to the buy-sell deal. However, *Seller* is implemented differently, with commitments among its members, which we do not elaborate here. The possible commitments of the Valvano-cum-Hoosier VE include the following.

- The VE will give price quotes to anyone who requests them.
- The VE will refund the purchase price if an order with matching valves and hoses cannot be fulfilled. There are still no refunds if an order for matching valves and hoses can be fulfilled.
- If the VE cannot fulfill an order, it will try to find an alternative order that will satisfy *Customer*'s requirements.

Recall that *val* or *hos* would not take refunds individually. Thus a customer might be saddled with valves for which matching hoses could not be found. However, when dealing with the VE, a customer can get a refund in those situations. ▌

In the above examples, the actions are performed by the constituents of the SoCom. Sometimes, however, it is useful to perform actions at a higher level SoCom. Such actions are necessary when the actions of the member agents must be atomically performed or undone. Example 3 is related to this situation.

Example 3 Continuing with Example 2, suppose an order for matching valves and hoses is successfully placed. It turns out later that the valve manufacturer discontinued the model that was ordered, but recommends a substitute. The substitute valve fits different diameter hoses than the original choice. The VE knows that the original order could be satisfied using the new valve and a different set of hoses. The VE can handle this replacement itself and, based on its prior commitment, not charge the customer any extra. The customer does not need to know of the internal exchanges among the members of the VE SoCom. Figure 3 illustrates the execution. ▌

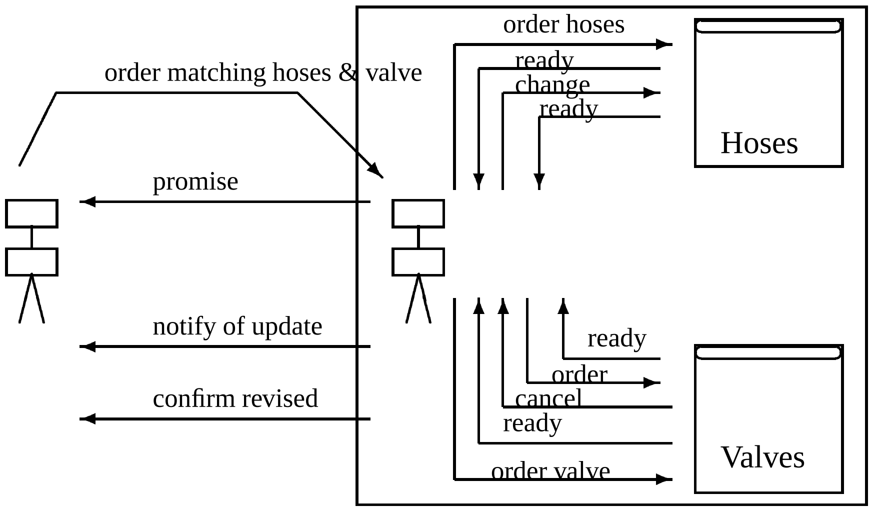

Figure 3: Commitment-Based Recovery

In the above example, the discontinuation of a valve after an order for it was accepted is a kind of failure that arises after the original interaction had ended. Traditional approaches would be inapplicable in such a situation.

4 Discussion

We started with an informal characterization of workflow. Although workflows are desirable in the open, inter-networked information environments of today, current workflow technology leaves much to be desired. The problems with current technology are not ones of mere detail, but are fundamental to the abstractions used for modeling and computation. We believe that the careful application of agents, in the form of multiagent systems, will yield rich dividends.

There are vast bodies of work on both multiagent systems and workflow specification and management. Even the specific topic of applying agents in workflow has been studied before. Among the earlier ones was some work in the Carnot project that we previously carried out [25]. In this work, we used a distributed expert system shell to implement a multiagent system, which was used to enact a telecommunications service order processing workflow. This approach used a combination of rules and nonmonotonic reasoning to handle exceptions. It got its inspiration in part from an ETM [2].

The advanced decision environment for decision tasks (ADEPT) project was also applied to a service order processing workflow [14]. The ADEPT project focused on negotiation among different agents to carry out a workflow. The agents use speech acts to make various negotiation moves. However, the underlying notion of commitments doesn't itself allow a contextual nesting, as in our approach.

The SMART project of the National Industrial Information Infrastructures Protocols (NIIIP) consortium deals with intelligent manufacturing [26]. A major focus of this project is on using agents for the manufacturing execution (or ”make-side”) of supply chains. Therefore, it also involves workflows over virtual enterprises [9]. SMART involves an alternative implementation for the commitment-based approach described here.

There are some interesting conceptual reasons why IOP is ideally suited to workflows. Multiagent systems and workflows have the some important unifying themes. Both require an emphasis on

- *openness* characterized by environments whose membership and behavior change dynamically
- *local control* in order to preserve the interests of workflow designers and owners
- *coherent* behavior instead exclusively of consistent data states, leading to *global coherence* in the face of local control.

This leads to a natural match between the two scientific areas, and makes several synergies available. Here we focused on agents applied in workflow management. However, workflow techniques for coordination (i.e., control logic specification and execution) also apply naturally in coordinating agents. Some of these connections are explored elsewhere [22].

Traditionally, the main stages in the workflow lifecycle are (a) analysis of a CIS, (b) design, (c) validation by simulation, (d) experimental deployment, and (e) production use. To these we add the stages (f) organize and coordinate and (g) refine *in situ*. Each stage in the lifecycle requires tools. However, successful tools must be based on correct models and accompanied by sound methodologies and patterns of usage.

The foregoing discussion brings forth a number of important shortcomings of current workflow technology; a slightly different, but useful, list is presented by Kamath & Ramamritham [15]. Besides accommodating heterogeneity and autonomy, there is need for improved methods for exception handling. Exceptions are difficult to predict during design. However, as they arise, humans or software agents (under human supervision) do handle them. This suggests that from the routine practice of exception handling, new workflow pathways are built, yielding a series of increasingly more complete production workflows. Indeed, the different between experimental and production use is primarily one of completeness in handling exceptions. Thus, design and enactment of workflows must be interleaved. Another research issue of great importance is specifying and controlling interactions among workflow instances and models.

Acknowledgments

This paper is based on a talk given at the *IJCAI Workshop on Business Applications of Artificial Intelligence* held in Nagoya, Japan in August 1997. An seminar on this topic was hosted by the IBM Intelligent Agents Group in RTP, North Carolina in May 1997. Parts of Section 2 were presented at the *ECAI Workshop on Learning in Distributed Artificial Intelligence Systems* held in Budapest in August 1996, and parts of Section 3 in the *2nd International Workshop on Cooperative Information Agents* held in Paris in July 1998. However, this is the first journal version of this material. We are indebted to a number of people for comments, especially Manny Aparicio, Carl Hewitt, and Aditya Ghose.

Munindar Singh is supported by the US National Science Foundation under grant IIS-9624425 (Career Award), IBM, and the NCSU College of Engineering. Michael Huhns is supported by the US Defense Advanced Research Projects Agency.

References

[1] Yuri Breitbart, Andrew Deacon, Hans-Jörg Schek, Amit P. Sheth, and Gerhard Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. *SIGMOD Record*, 22(3), September 1993.

[2] Alejandro Buchmann, M. Tamer Özsu, Mark Hornick, Dimitrios Georgakopoulos, and Frank A. Manola. A transaction model for active distributed object systems. In *[5]*, chapter 5, pages 123–158. 1992.

[3] Omran A. Bukhres and Ahmed K. Elmagarmid, editors. *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice-Hall, 1996.

[4] Panos K. Chrysanthis and Krithi Ramamritham. Synthesis of extended transaction models using ACTA. *ACM Transactions on Database Systems*, 19(3):450–491, September 1994.

[5] Ahmed K. Elmagarmid, editor. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, San Mateo, 1992.

[6] Martin Fowler. *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley, Reading, MA, 1997.

[7] Les Gasser. Social conceptions of knowledge and action: DAI foundations and open systems semantics. In *[12]*, pages 389–404. 1998. (Reprinted from *Artificial Intelligence, 1991*).

[8] Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–152, April 1995.

[9] Charles R. Gilman, Manuel Aparicio, J. Barry, Timothy Durniak, Herman Lam, and Rajiv Ramnath. Integration of design and manufacturing in a virtual enterprise using enterprise rules, intelligent agents, STEP, and work flow. In *SPIE Proceedings on Architectures, Networks, and Intelligent Systems for Manufacturing Integration*, pages 160–171, 1997.

[10] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, San Mateo, 1993.

[11] Meichun Hsu, editor. *Special Issue on Workflow and Extended Transaction Systems*, volume 16(2) of *Bulletin of the IEEE Technical Committee on Data Engineering*. June 1993. Contains 13 articles.

[12] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francisco, 1998.

[13] Anuj K. Jain and Munindar P. Singh. Using spheres of commitment to support virtual enterprises. In *Proceedings of the 4th ISPE International Conference on Concurrent Engineering: Research and Applications (CE)*, pages 469–476. International Society for Productivity Enhancements (ISPE), August 1997.

[14] N. R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O'Brien, and M. E. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(2&3):105–130, 1996.

[15] Mohan Kamath and Krithi Ramamritham. Bridging the gap between transaction management and workflow management. In *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions*, May 1996. `http:// optimus. cs.uga.edu:5080/ activities/NSF-workflow/ kamath.html`.

[16] Morten Kyng and Lars Mathiassen, editors. *Computers and Design in Context*. MIT Press, Cambridge, MA, 1997.

[17] Jan Ljungberg and Peter Holm. Speech acts on trial. In *[16]*, chapter 12, pages 317–347. 1997.

[18] Raúl Medina-Mora and Kelly W. Cartron. ActionWorkflow^R in use: Clark County department of business license. In *Proceedings of the 12th International Conference on Data Engineering (ICDE)*, pages 288–294, February 1996.

[19] Mike P. Papazoglou, Steven C. Laufmann, and Timothy K. Sellis. An organizational framework for cooperating intelligent information systems. *International Journal of Intelligent and Cooperative Information Systems*, 1(1):169–202, 1992.

[20] Marek Rusinkiewicz, Amit Sheth, and George Karabatis. Using polytransactions to manage interdependent data. In *[5]*, chapter 14, pages 555–581. 1992.

[21] Munindar P. Singh. Commitments among autonomous agents in information-rich environments. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*, pages 141–155. Springer-Verlag, May 1997.

[22] Munindar P. Singh. A customizable coordination service for autonomous agents. In *Intelligent Agents IV: Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL-97)*, pages 93–106. Springer-Verlag, 1998.

[23] Munindar P. Singh. The intentions of teams: Team structure, endodeixis, and exodeixis. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI)*, pages 303–307. John Wiley, August 1998.

[24] Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 1998. In press.

[25] Munindar P. Singh and Michael N. Huhns. Automating workflows for service provisioning: Integrating AI and database technologies. *IEEE Expert*, 9(5):19–23, October 1994.

[26] SMART project description. http://smart.npo.org/, 1997. National Industrial Information Infrastrucure Protocols (NIIIP) Consortium.

[27] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley, Reading, MA, 1987.