Anuj K. Jain, Manuel Aparicio IV,
and Munindar P. Singh

AGENTS FOR PROCESS COHERENCE IN Virtual Enterprises

At first glance, autonomy is a mixed blessing.

Open environments such as the Internet—and even corporate intranets—enable a large number of interested parties to use and enhance vast quantities of information. These environments support modern applications, such as manufacturing, virtual enterprises, and ubiquitous information access, which involve a number of information sources and component activities. However, without principled techniques to coordinate the various activities, any implementation would yield disjointed and error-prone behavior, while requiring excessive effort to build and maintain.

The agent metaphor, long in study in artificial intelligence, has recently become popular in mainstream computing, largely due to its suitability for open environments. Agents can be thought of as active objects with some special properties tailored to open environments. For our purposes, the key aspects of agents are their autonomy and abilities to perceive, reason, and act in their environments, as well as to socially interact and communicate with other agents [7]. When agents interact with one another they form a multiagent system. As part of a multiagent system, agents can capture and apply the semantic constraints among heterogeneous components in order to enact distributed workflows.

Autonomy is critical in open environments. Consider a manufacturing scenario requiring supply-chain coordination. It is natural to model independent companies in a supply chain as represented by autonomous agents. But, at first sight, autonomy is a mixed blessing—if the companies behaved arbitrarily, the supply chain would break. Consequently, our main technical challenge is how to manage autonomy, that is, how to maximize freedom without letting the system devolve into chaos. We propose that the main basis for managing autonomy lies in the notion of commitments. A flexible formulation of commitments can provide a natural means through which autonomous agents may voluntarily constrain their behavior. By flexible, we mean that it should be possible to cancel or otherwise modify the commitments. Consider a situation in which a purchaser is trying to obtain some parts from a vendor. We would like the vendor to commit to delivering the correct parts of the right quality to the purchaser. However, it is important that the supply chain be able to survive exceptions such as when the manufacturing plant goes down in an earthquake, or when the purchaser decides that it needs the parts to be of a lower error tolerance than initially ordered.

Information cannot be understood independently of the processes that create or consume it. Flexibility of behavior and the ability to recover from failures require an approach that is sensitive to how those processes interact. We show that when agents are associated with each independent process, our flexible notion of commitments can capture the desired interactions among those processes.

Spheres of Commitment

A multiagent system can be viewed as a *sphere of commitment*, which encapsulates the promises and obligations the agents may have toward each other. Spheres of commitment generalize the traditional ideas of information management so as to overcome their historical weaknesses. Information management involves three main concerns, which must be addressed by any approach for constructing information-based solutions:

- Data Integrity and Flow: Correctness of data and how it is conveyed from one party to another.
- Organizational Structure: How the various parties relate to each other.
- Autonomy: How the autonomy of the different parties is preserved.

Table 1 summarizes the major abstractions for programming such composite activities. Database transactions are the most rigorous, and require that only correct data ever be visible [6]. This requirement entails that outputs be released only when a transaction is completed. Thus, the producer of the data is restricted, but the consumer has full autonomy with respect to that data—it may use the data as it pleases. Further, the fact that outputs are released upon termination means that the transactions must terminate and may not exchange results with other transactions. Spheres of control release their results early, but may undo and redo the consuming computations if the results prove to be invalid—thus, the consumers have no autonomy [2]. Extended transactions release results liberally, but restrict the autonomy of their components by requiring compensating subtransactions to undo the effects of data that are invalidated [3]. Workflows ignore the integrity aspects, but capture the data flow required by specific applications [4]. They allow autonomy, but are not flexible. In general, workflow tools can capture routine processes, but the specifications become unmanageably complex—thus useless—when too many contingencies and exception conditions are specified. Further, workflow specifications are static; thus, defining workflows can be an onerous task. By contrast, commitments—such as those between manufacturers—are often dynamically formed and enacted.

Our approach focuses on how the different components achieve coherence in their interactions. The word "coherence" means a systematic or logical integration of diverse elements. Data integrity is a requirement that must be met while pursuing integration of heterogeneous systems. Otherwise, a multiagent-

Support Provided (columns) Technique (rows)	Integrity and Data Flow	Autonomy	Organizational Structure
Traditional Transactions	Fixed	Consumers	None
Spheres of Control	Relaxed	Producers	Rigid
Extended Transactions	Relaxed	Both, limited	Rigid
Workflows	None	Both	Relaxed
Spheres of Commitment	Relaxed	Both, constrained	Flexible

Table 1. Computational abstractions summarized

based information system will be nothing more than a specialized workflow system, which focuses only on control and data flow aspects and disregards integrity. In many practical situations, correctness requirements cannot be defined without reference to processes. The same data state may correspond to multiple histories of actions and interactions among the participants, and only some of those histories may be deemed acceptable. For example, a delayed order is undesirable in a supply chain, but not if the customer has granted an extension to obtain a higher quality outcome. We refer to the felicity of the (desired) interactions as *process coherence*. Our solution is to use a flexible formulation of commitments, which provides a high-level approach to achieving coherence in the processes executed by different agents. Control and data flow are managed in an organizationally felicitous manner to achieve coherence; the "right" kind of integrity is merely a consequence of coherence.

A *commitment* is a relationship between a debtor, a creditor, a context, and a proposition. The debtor owes it to the creditor to make the proposition true; the context serves as a witness and as the adjudicator of disputes. As you would expect, there is a close relationship between commitments and legal reasoning; this relationship is explored in [9]. A *sphere of commitment (SoCom)* is viewed conceptually as a scope within which a commitment applies; a SoCom is a multiagent system that the agents constitute, and which serves as the context for commitments among those agents. A SoCom is typically associated with a set of resources and authorities over them. A SoCom is modeled using a representative agent, which behaves as a group leader.

The agents can represent nonterminating computations. Their results, therefore, must be released prematurely, even if only to be invalidated later. As described here, recovery is effected by having the agents possibly adjust, but always satisfy, their commitments in the face of exception conditions. Typically, the agents communicate with other agents in order to create or adjust their commitments. The recipients autonomously process the communications. Consequently, by employing flexible commitments, we can achieve recovery without violating the autonomy of the consumers. Our approach applies recursively in that each SoCom can itself be treated as an agent and participate in larger SoComs. However, we will not emphasize the nesting of SoComs in this article.

Because flexibility is critical to our approach, we must allow the commitments to be manipulated in various ways, or even canceled. Consequently, we define the following major actions or operations on commitments:

- Create: instantiate, performed by debtor or context (by putting the debtor in a certain role).
- Discharge: satisfy, performed by the debtor (for example, through either physical actions or communication).
- Cancel: give up, performed by the debtor.
- Delegate: make another agent the debtor, performed by debtor or context.
- Assign: make another agent the creditor, performed by creditor or context.
- Release: eliminate entirely, performed by creditor or context.

The preceding set of operations is complete in the sense that it covers the possible manipulations to the different components of a commitment. By contrast, traditional commitments, once created, can only be discharged; notably, the cancel operation is not allowed. Obviously, these operations, especially *cancel*, cannot be wantonly performed, because that would undermine the very idea of a commitment. These operations are, therefore, typically governed by metacommitments. The set of applicable *metacommitments* effectively defines the structure of a multiagent system.

An *abstract SoCom* defines the structure of a SoCom using *roles*, instead of actual agents. Each role comes with a description of its requirements in terms of *resources, capabilities,* and *capacity*. A resource is instantiated with an information resource. A capability is a functionality that any agent playing the role may be expected to perform. The capacity corresponds to the spare capacity an agent must have in order to adopt the role. An abstract SoCom specifies the commitments associated with a role—these are commitments that agents playing that role are expected to satisfy. An abstract SoCom may also specify some lower-level constraints on the interactions of the agents, for example, whether an agent should initiate an interaction or wait for another agent to begin.

A *concrete SoCom* is obtained by naming an abstract SoCom, and binding agents to its roles. An agent may adopt more than one role in an abstract SoCom, and participate in more than one concrete SoCom concurrently. In order to adopt a role, an agent must certify that it has the requisite resources and capabilities, can spare the needed capacity, and that it is willing to acquire all the commitments associated with that role. Role adoption is the main way in which commitments are created. The commitments associated with a role are typically metacommitments. When the agents interact, their metacommitments yield the base-level commitments that the agents then attempt to discharge. For example, a request for valves by an agent may lead to the other agent committing to supply the requested number of valves. Each operation on commitments may potentially engage one or more metacommitments. For example, if the agreed-upon model of valve is discontinued, and the vendor cannot supply it, the vendor may commit to supplying a superior model for the same price.

The *SoCom manager* holds the definitions of the abstract SoComs, and supports their instantiation. Agents must register with the SoCom manager to inform it of their capabilities. They can browse the abstract SoCom definitions and also inform the manager of the roles (in specific abstract SoComs) that they are willing to play. While carrying out some application-specific task, such as creating a hose and valve configuration, the agents may request the creation of a concrete SoCom. The SoCom manager communicates with the other agents to determine their suitability for a specified role. If an agent offers or agrees to participate, the manager checks whether it meets the requirements. When the concrete SoCom is created, the agents in it can perform the capabilities of their chosen roles. Upon binding to a role, an agent adopts the commitments that go with that role. Thus, an agent should not be given roles whose commitments conflict. The agents then act according to the commitments of all the roles they are playing.

Implementing our approach requires representing and reasoning about commitments. Our approach was prototyped using IBM's Agent Builder Environment (ABE) [1], which provides an open architecture in which additional functionalities can be easily added. ABE includes a rule-based reasoning system; knowledge is specified via sets of declarative rules and facts. Our prototype enhances ABE with additional functionality, so that copies of ABE, with suitable knowledge, can function as different agents and communicate with each other (see sidebar). Our approach also encapsulates spheres of commitments as well-defined rules of behavior, and requires representing commitments explicitly. However, in almost all cases in which commitments are applied, there are already corresponding manual routines in place and the specific commitments to be represented are stored anyway, for example, in databases of pending orders or bills that enterprises routinely maintain.

Applications in Manufacturing

Manufacturing is a touchstone application area for any approach that deals with information management in open environments. This is because modern manufacturing is naturally distributed, involves a large number of autonomous commercial entities with a variety of heterogeneous information systems, makes use of human decision making, faces the realities of failure and exception in physical processes and contractual arrangements, and yet requires that the manufactured products meet design specifications and other quality requirements. Because they were not sensitive to these constraints, previous attempts at applying computing in manufacturing have had only had limited success.

With recent advances in the computing and communications infrastructure, there has been a resurgence of interest in manufacturing applications, especially in those dealing with the coordination of processes in different enterprises. Supply chains are the materiel flows that are arranged among different companies to accomplish a large manufacturing process. Virtual enterprises are composed from several companies, which enables them to make joint commitments to their common customers. Although the companies are involved in a tight relationship in order to make joint commitments, they still retain their autonomy. To address the application of advanced computing in manufacturing, the Advanced Technologies Program of the U.S. National Institute of Standards and Technology (NIST) is sponsoring is a multiyear, multimillion dollar project called the SMART project [11]. SMART is run by the National Industrial Information Infrastructures Protocols (NIIIP) consortium, which includes several computing and manufacturing companies. SMART's contributions include developing agent technology for manufacturing execution systems (MES)—that is, the "make-side" of supply chains [5]. Because the requirements for the control of proprietary processes and other MES information correspond well with our motivations, spheres of commitment apply naturally in the SMART project.

Figure 1 shows a simplified supply chain from the SMART project. Here *Hot Air Bros.* is an assembly plant, which procures valves (from *Valvano & Co.*) and hoses (from *Hoosier Inc.*), and assembles them

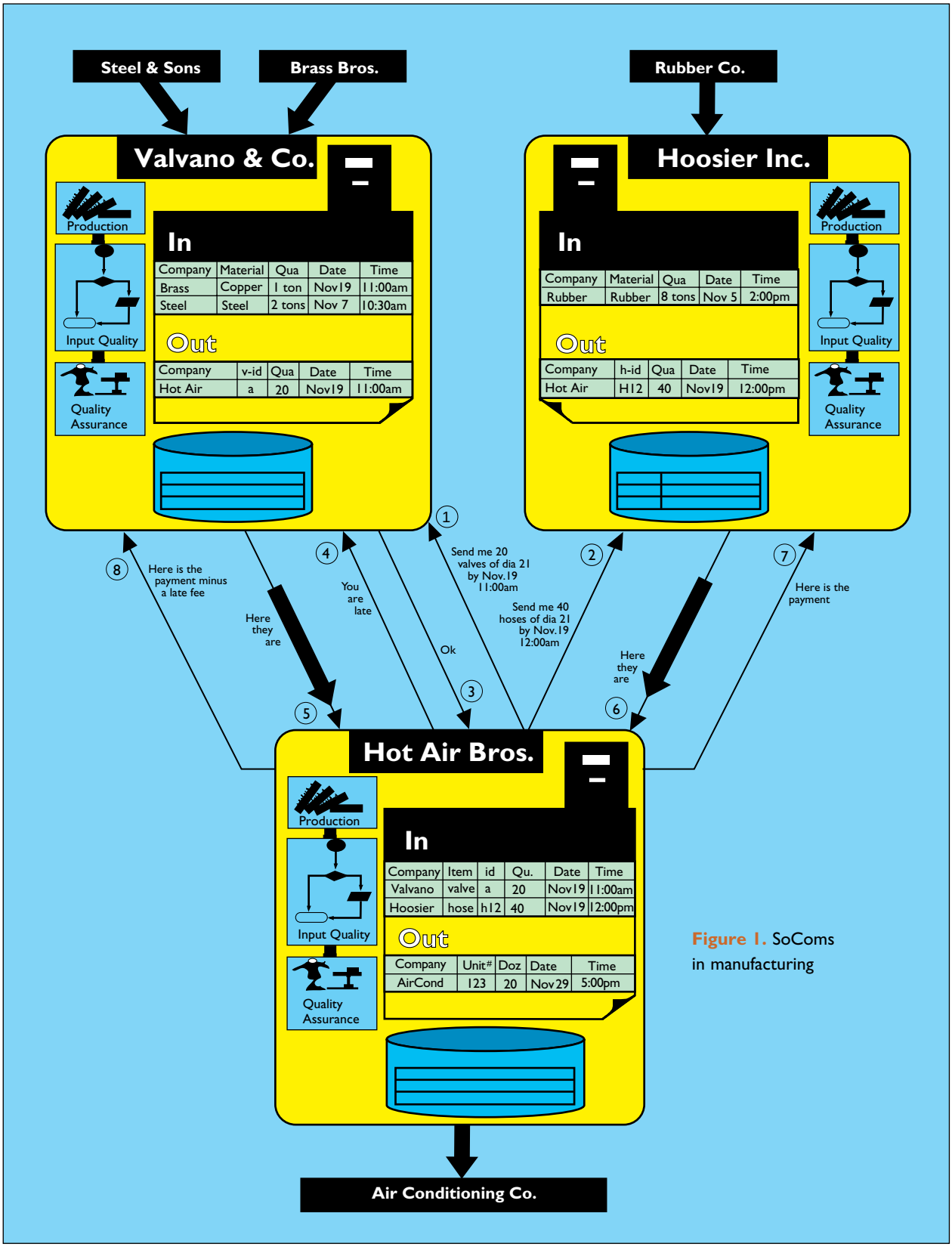

Figure 1. SoComs in manufacturing

into parts for air conditioners. In order for the entire system to operate efficiently, *Hot Air* must assemble units according to a schedule, which dictates the arrival of various parts from various vendors—these are the commitments owed *Hot Air*—and dictate the delivery of parts to others—these are the commit-

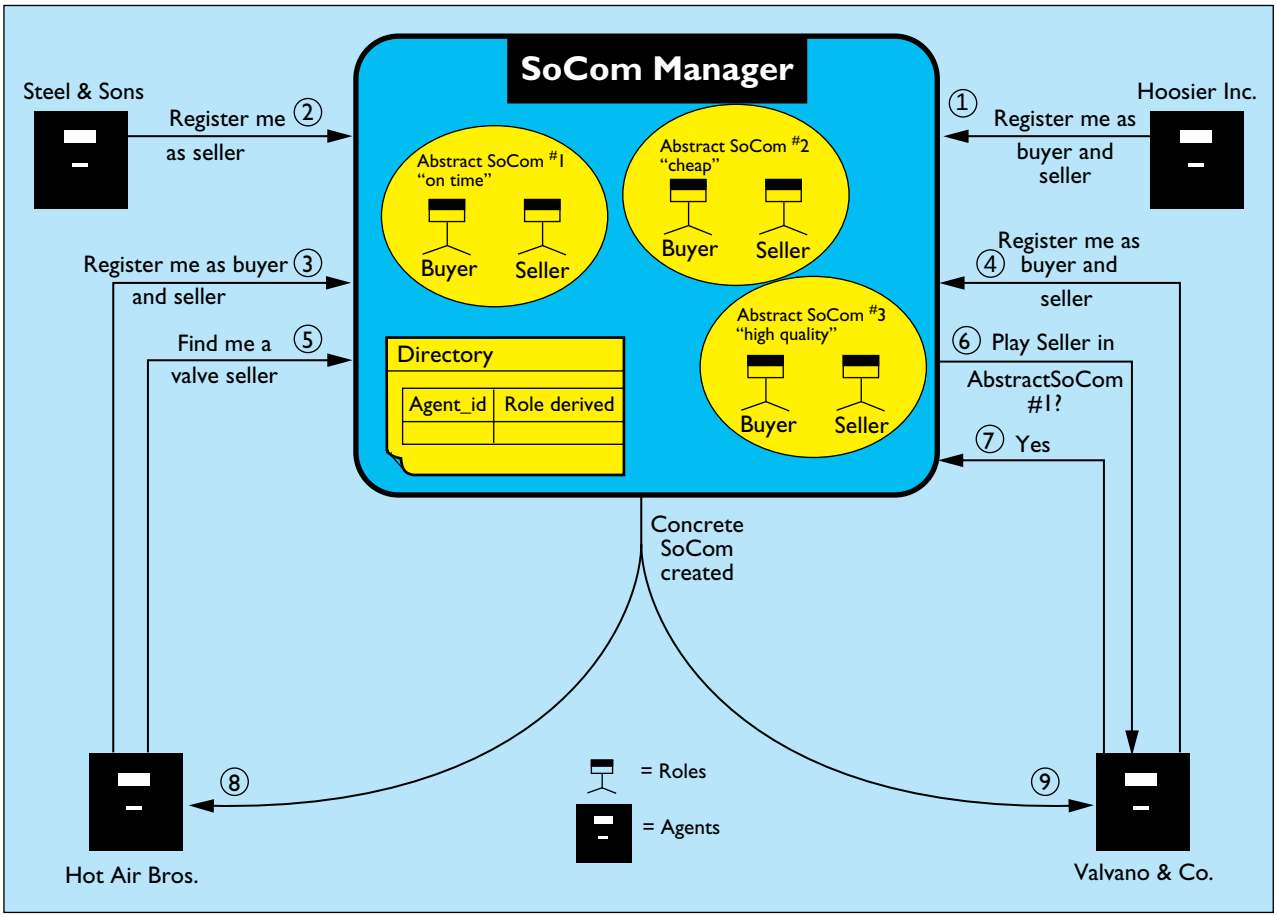

Figure 2. Instantiation of a concrete SoCom

ments that *Hot Air* owes to its customers. For the same reasons, the other participants are also creditors and debtors of commitments. Each participant may test the quality of the inputs, carry out a production process, and assure the quality of its products. Further, each participant monitors the events in its environment and communicates as necessary with its vendors and customers about the commitments.

In the example given, each participant also has some internal product information under its sole control. For instance, *Valvano* knows features of its valve models, specifically, their input and output diameters; *Hoosier* knows what diameter hoses it manufactures; and, *Hot Air* knows what configurations of hoses and valves it produces. For simplicity, we assume that the different agents speak the same language with regard to the products of their trade. This would be accomplished by using an *ontology*, that is, a representation of knowledge about the chosen domain. We postulate that *Hot Air* tracks how far the schedule has progressed in real time and in terms of tasks completed. For each delivery received, *Hot Air* determines whether it was on time, and evaluates whether it was of acceptable quality. Whenever a delivery fails to materialize on time and the delay causes some problems, *Hot Air* notifies the errant vendor and requests their attention. For each exception, a human member of the *Hot Air* organization may be notified—here, for simplicity, the human is not shown. In general, however, it is important to keep the human in the loop, both for robustness and for the acceptance of the technology in the manufacturing industry, which is conservative in adoption of new technologies. If a human is notified of a delay, then she is also notified of any corrective actions taken.

In this example, *Hot Air* orders hoses and valves. The hoses arrive, but the valves do not and *Hot Air* sends a reminder to *Valvano*. *Valvano* delivers the valves. Because the valves were late, *Hot Air* assesses a late fee on the payment. However, *Hot Air* pays *Hoosier* in full.

The example given presupposes the existence of the appropriate SoComs between *Hot Air* and *Valvano* and *Hot Air* and *Hoosier*. We now discuss how these SoComs themselves came into existence. It all begins with the SoCom manager, which records the abstract SoComs that have been published with it. Figure 2 illustrates the interactions between *Hot Air* and *Val-*

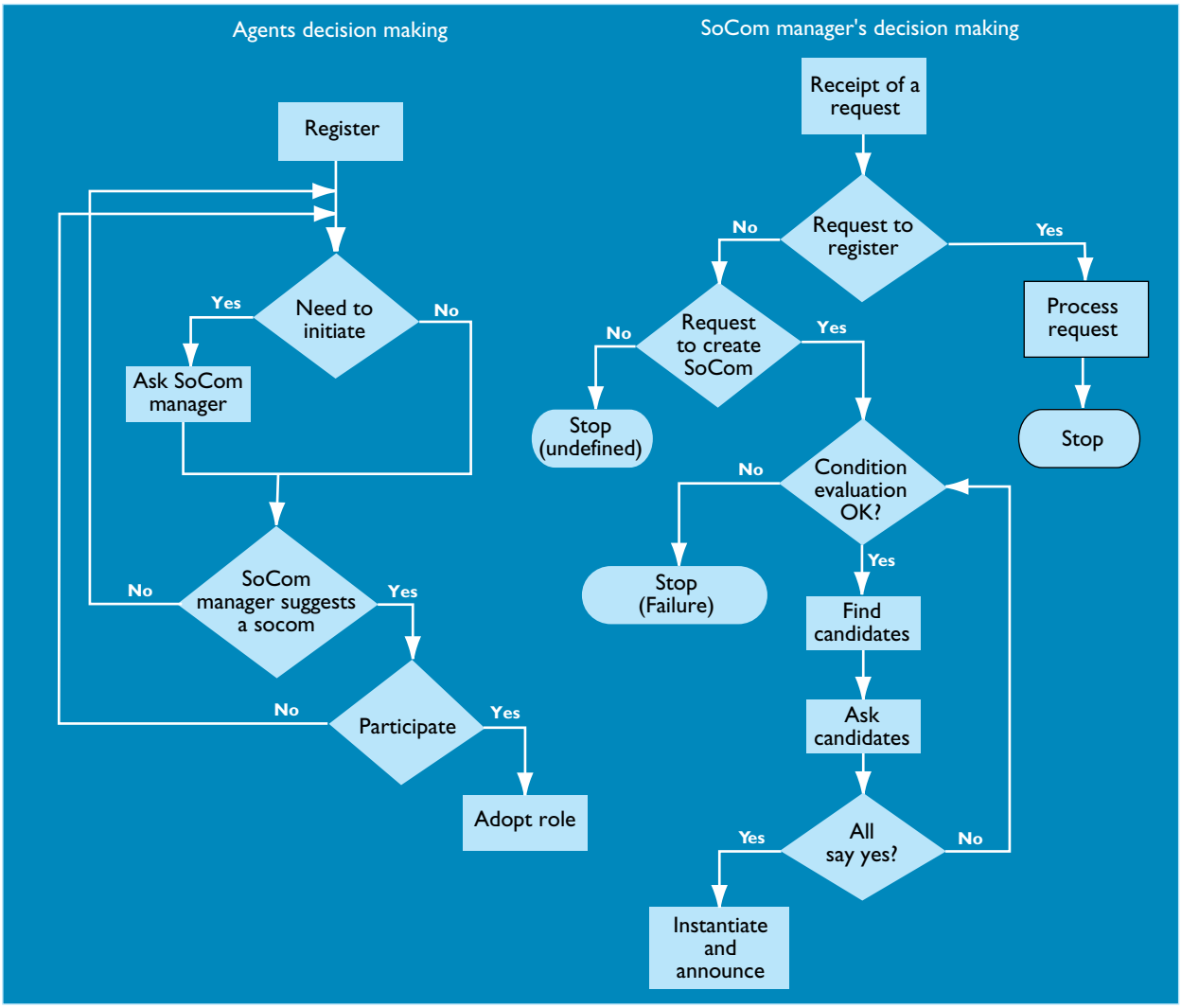

Figure 3. Interactions between agents and the SoCom manager

Commitments and Communication

Because our agents are autonomous, we must ensure that the interactions among them do not violate their independence. The most obvious autonomy-preserving interactions are communications. Two prominent agent communication languages (ACLs) are the Knowledge Query and Manipulation Language (KQML) [8] and the ACL of the Foundation for Intelligent and Physical Agents (FIPA). The FIPA ACL is rapidly spreading to replace KQML as the ACL of choice, so we followed its syntax in our prototype. Figure 4 shows an example message, with which the assembler requests the vendor, in response to a recent advertisement, to send it 20 valves of model "a" at the unit price of $40. The air-conditioning "ontology" is where the terms of the message content are defined.

```
(request
        :sender         Hot Air
        :receiver       Valvano
        :content        (send valves model-a 20 $40)
        :ontology       Air-Conditioning
        :in-reply-to    Advertisement-37
)
```

Figure 4. Sample agent communication detailing a message from Figure 1

The messages among agents either explicitly refer to a commitment—to create or manipulate it in some way—or presuppose the existence of metacommitments under which additional commitments are created. For example, if *Valvano* had a metacommitment

vano and the SoCom manager, leading up to the formation of a concrete SoCom. Figure 2 shows three abstract SoComs, all dealing with simple buying and selling, but providing differing sets of commitments: the delivery will be on time, the delivery will be of a specified quality (as opposed to merely best effort), or a guarantee of some configuration design constraints. Agents register with the SoCom manager and can browse the abstract SoComs. The SoCom manager records the capabilities of the different agents and their willingness to play specific roles. When requested to form a concrete SoCom, the manager requests likely agents to join. An agent can join a SoCom either by requesting to do so (for example, *Hot Air*), or because of another agent's request relayed to it (for example, *Valvano*) via the SoCom. After the SoCom is instantiated, the agents communicate directly. In general, the agents may communicate directly even to form the SoCom. Figure 3 gives a high-level description of the steps performed by an agent and by the SoCom manager.

Conclusion

Traditional programming techniques are designed for closed environments, in which the programmer has (at least in principle) complete knowledge of the meaning of the information and full control over the disposition of the participating activities. By contrast, in open environments, a programmer has partial knowledge of and virtually no control over the behavior of the components created by other designers and being executed by autonomous users. Although preserving the autonomy of participating components is crucial, unrestrained autonomy would be risky, because it may easily lead to undesirable consequences. Nowhere are these concerns more urgent than in manufacturing. As manufacturing becomes increasingly reliant on the dynamic formation and management of extended and overlapping virtual enterprises, agent-based, flexible approaches will play an increasing role.

Our approach seeks not data consistency directly, but a coherent state in the ongoing interactions of the participating components. This shift in focus from consistency to coherence not only facilitates automation, but is also more intuitive and closer to some aspects of human social behavior. People cannot make irrevocable promises when they do not fully control their environments, but they can warn each other of potential problems. For example, if an order is not going to come through, a good service would at least notify the others concerned. We believe the development of spheres of commitment is the lasting contribution of the present work. C

to fulfill orders, the request shown in Figure 4 would ordinarily lead it to commit to supplying the correct number of valves. Other messages may be used to cancel a commitment or to modify it some way. For example, if the valve factory is destroyed, *Valvano* would send a message canceling the commitment and simultaneously creating a commitment to supply alternative valves.

One of the benefits of a commitment-based approach is that it yields a natural account of how the participating agents may comply with the requirements of the roles they play. This idea can be used to give a public basis for the meanings of communications, and to verify the compliance of agents [10]. Verifying compliance is critical, because in an open environment, you cannot implicitly trust everyone.

References

1. Agent Builder Environment. http://www.networking.ibm.com/iag/iagsoft.htm.
2. Davies, C.T. Data processing spheres of control. *IBM Systems Journal 17*, 2 (1978), 179–198.
3. Elmagarmid, A.K., Ed. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann, San Mateo, 1992.
4. Georgakopoulos, D., Hornick, M., and Sheth, A. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases 3*, 2 (Apr. 1995), 119–152.
5. Gilman C.R., Aparicio M., Barry J., Durniak T., Lam H., and Ramnath R. 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*, (1997), pp. 160–171.
6. Gray J. and Reuter A. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, San Mateo, 1993.
7. Huhns, M.N. and Singh, M.P., Eds. *Readings in Agents*. Morgan Kaufmann, San Francisco, 1998.
8. Labrou Y., Finin T. Semantics and conversations for an agent communication language. In M.N. Huhns and M.P. Singh, Eds. *Readings in Agents*, Morgan Kaufmann, San Francisco, 1998, pp. 235–242.
9. Singh M.P. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 1999. To appear.
10. Singh M.P. Agent communication languages: Rethinking the principles. *IEEE Computer 31*, 12 (Dec. 1998), 40–47.
11. SMART. http://smart.npo.org/

Anuj K. Jain (anuj.jain@ericsson.com) is a software engineer with the Ericsson New Concepts Group in Research Triangle Park, NC.
Manuel Aparico IV (aparicio@us.ibm.com) is with the IBM Intelligent Agents and Knowledge Management Group in Research Triangle Park, NC.
Munindar P. Singh (singh@ncsu.edu) is an assistant professor with the Department of Computer Science at North Carolina State University, Raleigh.

Research leading to the preparation of this article supported by the NCSU College of Engineering, the National Science Foundation under grant IIS-9529179, the NIST Advanced Technology Program, IBM, and Ericsson.