Commitments Among Autonomous Agents in Information-Rich Environments[*]

Munindar P. Singh[**]

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA

singh@ncsu.edu

Abstract. Commitments are crucial to understanding and designing autonomous agents and multiagent systems. We propose a definition of commitments that applies especially well to agents in information-rich applications, such as electronic commerce and virtual enterprises. Our approach has a number of important features, including

- not gratuitously translating social concepts to psychological concepts
- distinguishing between satisfied and inapplicable commitments
- incorporating social policies to handle the creation, satisfaction, and cancelation of commitments
- relating commitments to organizational structure in a multiagent system
- showing how commitments are acquired by agents as a consequence of adopting a role.

1 Introduction

Commitments are central to DAI. In this paper, "commitment" refers to social, not psychological, commitment. Commitments have drawn much research attention because they are an important abstraction for characterizing, understanding, analyzing, and designing multiagent systems. Commitments help coordinate and structure multiagent systems to achieve coherence in their actions.

Multiagent systems are finding increasing application in heterogeneous and open information environments—such systems are called *cooperative information systems (CISs)* [Singh & Huhns, 1995]. CISs have increased expectations of robustness and guarantees of the atomicity, durability, and recoverability of actions. Our ongoing research program seeks to develop abstractions for building flexible CISs to the standards of robustness of traditional systems.

[*] This is an extended and revised version of a paper presented at the ICMAS-96 Workshop on Norms, Obligations, and Conventions. I would like to thanks Rosaria Conte, Christian Lemaitre, and the anonymous reviewers for their comments. I have benefited from discussions with several people over the years, most notably Nicholas Asher, Cristiano Castelfranchi, Les Gasser, Michael Georgeff, and Michael Huhns.

[**] This work is supported by the NCSU College of Engineering, the National Science Foundation under grants IRI-9529179 and IRI-9624425, and IBM corporation.

Technical Motivation Commitments arise not only in the study of agents, but also in distributed databases. However, databases (DB) implement a procedurally hard-wired and irrevocable form of commitment. Modern DB applications, which involve heterogeneity, flexibility, and human collaboration, do not fit the traditional mold. Some of these applications have been addressed using agent-based techniques, e.g., [Wittig, 1992; Singh & Huhns, 1994]; others with advanced database techniques, e.g., [Bukhres & Elmagarmid, 1996]; and still others by combining in organizational techniques, e.g., [Papazoglou *et al.*, 1992].

The DB and DAI strands of research into commitments have progressed without much cross-fertilization. The DB ideas have tended to be rigid, but in a manner that facilitates robustness. The DAI ideas have been more flexible. However, with respect to information systems, they do not guarantee correctness properties comparable to the DB approaches. We submit that a conceptually well-founded synthesis can yield abstractions for effectively programming CISs.

We view CISs as recursively composed loci of commitments. These commitments can be about actions, but in database settings they are typically about results that are released or "published" by different components. Whereas the traditional database approach is to release outputs only when they are definite, in the case of nonterminating computations, we cannot afford to wait till they end! In general, we must allow outputs to be released prematurely. This is also essential, for example, in cases where the given activities must cooperate, so they may exchange their partial results before they terminate.

The construction of effective CISs involves the careful synthesis of three kinds of concerns:

- data integrity: correctness of data despite concurrent access and failures;
- control and data flow: how triggering, i.e., control, information and data flows through the system; and
- organizational structure: how the various components relate to each other in achieving coherent behavior, e.g., whether a control signal is expected and would not be ignored depends on the organizational structure of the components.

Traditional nested transactions provide integrity, but restrict the other aspects. Extended Transaction Models (ETMs) also focus on integrity, but allow freer control and data flow at the cost of relaxing the integrity requirements. Database workflow approaches ignore the integrity aspects, but deliver the control and data flow required by specific applications. Workflows in groupware also provide application-specific control and data flow without regard to integrity.

In contrast with the above, our approach focuses on how the different components achieve coherence in their interactions. Control and data flow serve to achieve coherence, and integrity is a consequence of it. By organizational structure, we mean not only the roles that different agents play, but also the commitments that they may enter into based on their roles.

In our approach, each recursively composed CIS provides a context in which its constituent agents interact. In particular, the agents respect certain *commitment policies* and *cancelation policies*, which determine when they may adopt or drop commitments. In some cases, these policies might help achieve a correct data state; in others, they may only guarantee that the CIS as a whole is behaving properly.

Organization. Section 2 describes traditional ways of structuring computations. Section 3 discusses our approach to commitment, shows how it handles social policies and the structure of multiagent systems, and discusses its formal aspects and implementation. Section 4 reviews the pertinent literature from three main areas.

2 Problem: Structuring Computations in Open Information Systems

We introduce our running example, which involves a simplified form of electronic commerce and virtual enterprises.

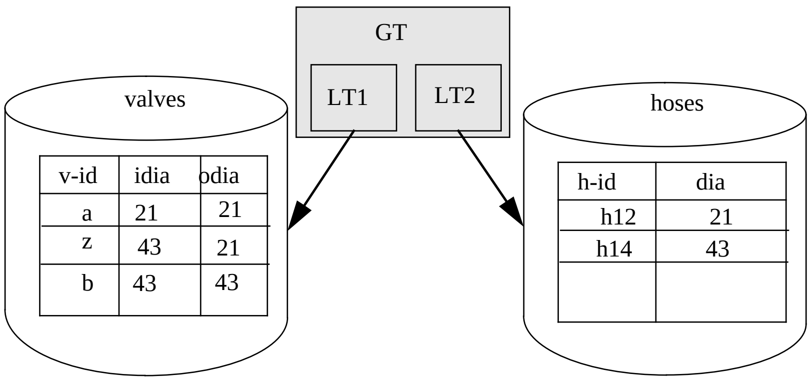

Fig. 1. Traditional (Closed) Transactions

Example 1. Suppose we need to purchase two interdependent parts—a valve and two hoses, with the requirement that their diameters match (otherwise, each is useless). Consider a composite activity that attempts to purchase a shipment of valves from Valvano & Co and matching hoses from Hoosier Inc., thus accessing the databases as shown in Figure 1 (please ignore GT, LT1, and LT2, for now). Let these subactivities be called *val* and *hos*, respectively. We imagine that Valvano and Hoosier form a virtual enterprise to provide a higher level of service to their common customers, but continue to run autonomous databases. The key requirement for the purchase is that either (a) both *val* and *hos* have an effect, or (b) neither does. ∎

Traditionally, it would be up to the application program to enforce this requirement. Although traditional database transactions have been used extensively in homogeneous settings, it now well-known that they are inappropriate for heterogeneous environments. We show why next. To avoid terminological confusion, we use the term "succeed" instead of the database term "commit" where only the success of an individual transaction is implied.

2.1 Traditional Database Transactions

Traditional transactions are computations that satisfy a number of useful properties, in particular the so-called ACID properties [Gray & Reuter, 1993].

- *atomicity:* all or none of a transaction happens
- *consistency:* a transaction preserves the consistency of the database
- *isolation:* intermediate results of a transaction are not visible externally
- *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.

Example 2. Continuing with Example 1, we can obtain database support for maintaining consistency as shown in Figure 1. GT is a global, closed-nested transaction corresponding to the purchase activity. It consists of local subtransactions, LT1 and LT2, corresponding to *val* and *hos*. GT preserves consistency (either both LT1 and LT2 succeed or neither does), and allows only correct purchases to be visible. ▮

Unfortunately, the above formulation proves highly undesirable. To ensure transaction atomicity, the system must ensure that both *val* and *hos* succeed, or neither does. To ensure transaction isolation, the system must ensure that no other transaction sees the intermediate results of *val* or *hos*. Further, if a transaction that runs on the same databases sees the final results of one subtransaction (e.g., *val*), then it also sees the final results of the other subtransaction (e.g., *hos*). The above requirements are stronger than our informal requirement that both or neither subtransaction should have an *effect*.

To realize the above transactional properties requires a mutual commit protocol, e.g., two-phase commit, to be executed. However, that might be impossible, since Valvano and Hoosier are independent enterprises and their databases may not even have visible precommit states, essential to execute a mutual commit protocol. Even if the precommit states are visible, it is seldom acceptable to lock resources while communicating with a remote site. Thus traditional transactions are unacceptable in heterogeneous environments.

2.2 Extended Database Transactions

Extended transaction models (ETMs) take some steps toward overcoming these limitations. However, they typically address only a part of the problem, chiefly by allowing results to be released prematurely. Failure recovery is typically achieved by compensating the subtransactions that erroneously recorded success (even though other related transactions did not)—the compensations are of course domain-specific. Consider the following example, which uses a simplified version of the DOM ETM [Buchmann *et al.*, 1992].

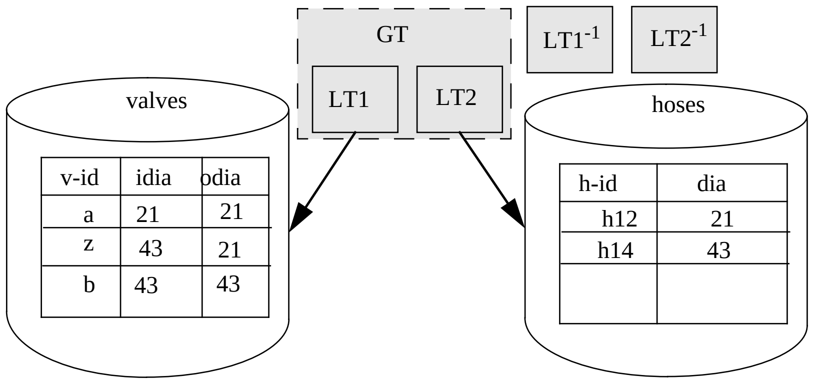

Fig. 2. Extended (Open) Transactions

Example 3. Continuing with Example 2, we now define a purchase activity as in Figure 2. Here, GT is an open-nested global transaction consisting of the *val* (LT1) and *hos* (LT2) subtransactions. GT executes LT1 and LT2 concurrently. The results of LT1 and LT2 are visible even before GT has completed. If both or neither succeed, consistency is preserved. If one succeeds and one fails, then either (a) the one that succeeded can be compensated through LT1^{-1} or LT2^{-1}, e.g., by canceling the its order, or (b) the one that failed can be retried. ▌

This assumes that (a) compensating actions are defined for some of the subtransactions, and (b) it is acceptable to allow temporary inconsistencies. Extended transaction models do not provide perspicuous means to specify and schedule activities, nor means to coordinate them. Scheduling techniques are hard-coded separately for each transaction model.

2.3 Agents

Agents can perform several functions in enterprise integration scenarios. They can capture the semantic constraints and apply them in order to execute or enact workflows in an integrity-preserving manner. In this way, agents can carry out the business processes in an enterprise. For example, although database consistency is assured even if both transactions fail, the agent might encode that some progress is essential from the purchaser's standpoint.

Example 4. In the scenario of Examples 1 and 2, a purchasing agent can be used. This agent initiates *val* and *hos* concurrently. If both succeed, the purchase succeeds. However, if one or both fail, the agent can (a) retry the failed transactions a certain number of times, (b) search for alternative sources and attempt the transactions there, or (c) negotiate with the user's agent and with database agents to enable progress. ▮

The agents can thus realize workflows that correspond to generalized forms of extended transaction models. More importantly, however, the agents can form a CIS and interact with each other in an effective manner. For example, agents can coordinate workflows so that the unavoidable interactions among those workflows do not violate data integrity or prevent progress. Further, the requirements for each workflow can be locally captured by the resource administrators most familiar with the resources that the workflow involves. The formal specifications are kept modular and small, which facilitates their acquisition and verification.

Example 5. Consider ongoing activities to repeatedly stock inventory, ship goods to customers, receive purchase orders, and forecast the market demand. These activities must be coordinated. (a) Stocking up replenishes the inventory for shipping. The stocking up and shipping agents must agree whether to build up large inventories or break up large purchase orders. (b) A purchase order must be received before it is shipped. (c) Market forecasting can either trigger stocking up, or disable it. ▮

But how can we ensure that the agents behave properly? Surely, we need better abstractions than having the application programmer supply hardcoded solutions.

2.4 The Problem

Thus the main problem is to structure activities in a manner that can respect the autonomy of the information resources. The database approaches are restrictive. The agent approaches are flexible, but there is need for tools and formal approaches for designing them. In particular, there is need for a notion of commitment that flexibly reflects the organizational structure of how agents interact.

3 Solution: Spheres of Commitment

We define commitments in a manner that satisfies the above requirements. We dub our approach *spheres of commitment (SoCom)*. SoComs involve not only the data integrity issues, but also reflect the organizational structure associated with CISs, which constrains the control and data flow as well. Each SoCom is autonomous, and has authority over some information resources, on the basis of which it can enter into commitments about those resources.

3.1 Spheres of Control

To best appreciate our approach, it is instructive to see how *spheres of control (SoCs)* work. SoCs, which were proposed about two decades ago [Davies, 1978], capture some of the same intuitions as the extended transaction models. The database community is seeing a resurgence of interest in SoCs as the limitations of traditional transactions are being realized [Gray & Reuter, 1993, pp. 174–180]. Intuitively, SoCs attempt to contain the effects of an action as long as there might be a necessity to undo them. Ordinarily, a result is released only when it is established that it is correct (and will remain correct). However, if a result may later have to be undone, it can be released only if an SoC can be set up that encloses the activities that consume the result. When the result needs to be undone, the enclosing SoC can undo the activities that used that result.

Example 6. Continuing with Example 2, we can define an SoC that contains the *val* and *hos* subtransactions. The results of these subtransactions can be made visible only to those activities that are also within the given SoC. If the results of *val* and *hos* are inappropriately released, they can be undone, possibly by also undoing the activities that consumed those results. ▌

SoCs generalize transactions by essentially requiring the entire execution history to be maintained. SoCs require rolling back the execution to undo the effects of erroneously committed activities, followed by rolling forward the execution to redo the necessary computations. Unfortunately, despite their generality in some respects, in a fundamental sense SoCs remain almost as restrictive as traditional transactions. This is because SoCs are also data-centric, and attempt to preserve or restore data integrity. Specifically, we believe that the problem lies in two facts.

- SoCs are not active entities, and
- SoCs view commitments in the traditional DB sense, which is as depending solely on the computation that commits, not on the interplay between the computation that commits, and the computations that take advantage of that commitment.

3.2 Commitments

Despite its shortcomings, we find the SoC concept useful in motivating SoComs. SoComs provide a means for organizing agents and CISs. We begin by eliminating the distinction between agents and multiagent systems. We view agents as being either individuals or groups, which are recursively composed of agents. In this sense, a CIS is an agent and is potentially composed of agents. We augment our initial definition of agents to additionally require them to be loci of social commitments. Thus, each agent or CIS can be a SoCom.

Agents interact by forming commitments toward one another. We use the term *commiter* to refer to the agent that makes a commitment, and the term *commitee* (not "committee") to refer to the agent who receives the commitment. Commitments are formed in a context, which is given by the enclosing CIS (or, ultimately, by society at large). We refer to this as the *context group*. Concomitant with a commitment is a specification of how it may be satisfactorily discharged, and a specification of how it may be canceled. We define three main kinds of *social actions*, which are instantiated by the following operations on commitments.

- *create*
- (satisfactorily) *discharge*
- *cancel*

Based on the above intuitions, we motivate the following logical form for commitments.

Definition 1. A commitment is an expression of the form $\mathsf{C}(x, y, p, G, d)$, where x is the commiter, y the commitee, G the context group, p the discharge condition, and d the cancelation condition (formally a proposition).

It is convenient to define the operations of *notify* and *release* as follows. $\mathit{notify}(x, y, q)$ mean that x notifies y of q, and $\mathit{release}(y, c)$ means that y "releases" the commiter of commitment c, essentially agreeing to its success. In a sense, these are low-level operations, which can be used to implement the above social actions. They are however, quite natural and common to a number of domains.

Where necessary, we include the *release* requirements in the discharge condition. For example, it is possible to commit to "making the sky green," or "making the sky appear green to the commitee" (these are different commitments, with different chances of satisfiability). We now discuss some possible cancelation conditions, which relate to different situations. Let the given commitment be $c = \mathsf{C}(x, y, p, G, d)$. (Explicitly naming the commitment itself enables setting up mutual commitments.)

P1. $d = \mathsf{false}$: the commitment is irrevocable.
P2. $d = \mathit{notify}(x, y, q)$: the commiter is only obliged to notify the commitee, where q means that the commitment c is being canceled.
P3. $d = \mathsf{true}$: the commitment can be given up at will, and is effectively not a commitment at all.
P4. $d = \mathit{release}(y, c)$: the commitee must explicitly release the commiter.
P5. $d = \mathit{release}(G, c)$: the context group must explicitly release the commiter.

Example 7. Consider the situation of Example 2 after *val* has successfully completed its internal processing, but not yet officially published its results. This can be modeled as $c_1 = \mathsf{C}(\mathit{val}, \mathit{hos}, \mathit{succeed}(\mathit{val}), G, \mathit{cannot\_succeed}(\mathit{hos}))$. Here

G corresponds to the global transaction. The above commitment means that if *val* can succeed, it will unless *hos* cannot succeed. Additional commitments are need to capture the entire specification, e.g., to ensure that *val* does not succeed unless *hos* succeeds. ▮

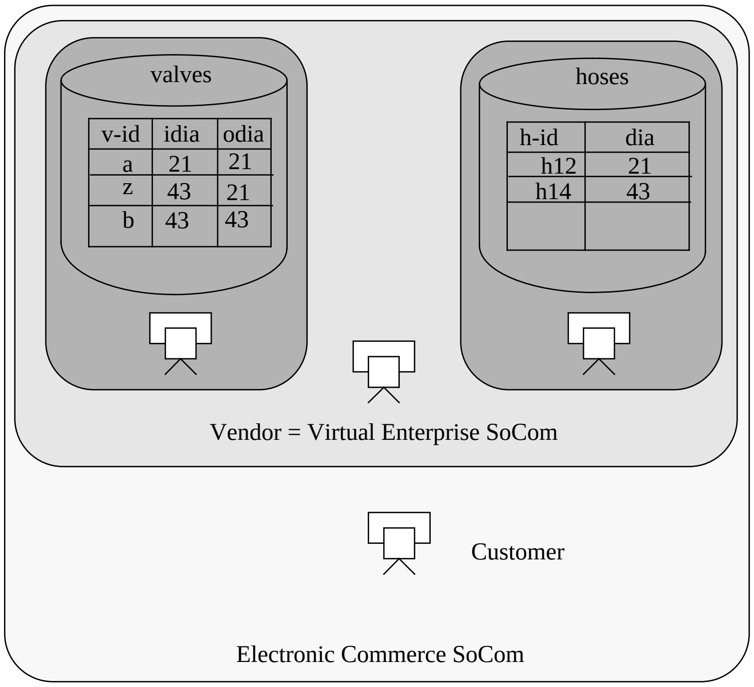

Fig. 3. Nested Spheres of Commitment

Example 8. Continuing with Example 2, we define two SoComs—shown in Figure 3—with authority over the Valvano and Hoosier databases, respectively. These SoComs execute the corresponding subtransactions. There is also a SoCom corresponding to the Valvano-cum-Hoosier virtual enterprise (VE). As in Example 4, a customer agent carries out the desired workflow. This agent might itself be a SoCom with authority over purchases in its enterprise. A possible set of commitments could be as follows.

- The Valvano and Hoosier SoComs inform other agents as to how many units of a valve or hose they have in stock.
- If stock is available, they will "lay-away" up to a certain number of units for a (potential) customer; if stock is not available, they will notify the customer.
- However, if the stock falls low, the SoComs can ask a customer to decide or pay a nonrefundable deposit.

– The customer commits to releasing a lay-away if he decides against the purchase.
– The customer can request to apply the deposit for another purchase, at the selling SoCom's discretion.
– The customer can request a refund from the VE SoCom. The entire deposit of purchase price is refunded if a matching item (hose or valve) was not available.

In this setup, *val* or *hos* in general cannot be undone—customers can't expect full refunds after the purchase. However, if *val* and *hos* are being performed as a package, i.e., in the Valvano-cum-Hoosier VE, the VE SoCom ensures that customers will get refunds if one of the subtransactions fails. Other customers who were told that stock was not available will be notified, and given an opportunity to retry their purchase. Lastly, negotiation and exceptions are allowed, although after commitments have been made, the decision might reside with one of the participants. ▌

3.3 Social Policies

Social policies are policies that govern the social actions—they characterize when the associated action occurs. It is helpful to define the *order* of a commitment as follows.

Definition 2. Consider a commitment $c = \mathsf{C}(x, y, p, G, d)$. c is 0-order iff p makes no reference to any commitments. c is $(i+1)$-order iff the highest order commitment referred to in p is i-order.

Social policies are formally represented as conditional expressions. Policies for i-order commitments are $(i+1)$-order commitments. Even for policies, our fundamental correctness condition remains: *if a commitment is created, then it must satisfactorily discharged, unless it is canceled in time.* A variety of policies can be defined, with applicability in different settings.

Policies also have a computational significance, which is that they can lead to commitments being created, discharged, or canceled without reference to the context group. It is the locality of policies that makes them useful in practice. Consider a simple example.

Example 9. Continuing with Example 8, consider a customer who makes an Ecash deposit for some valves, but later decides not to get them. He might avoid losing his deposit by finding another purchaser for those valves. The selling SoCom would have to accept an alternative purchaser if the applicable social policies allow that, unless they were explicitly overridden by the contract. There is no need to invoke the context group, i.e., complain to the VE SoCom, or to file a lawsuit. ▌

In the above the actions are performed by the constituent CISs. Sometimes, however, it is useful to perform actions at a higher level CIS. Such actions might be necessary when the actions of the member agents need to be atomically performed or undone.

Example 10. Continuing with Example 8, 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 takes a different hose diameter than the original choice. Suppose the VE SoCom knows the relevant constraint, and is authorized to update the order. It would be better to undo and rerun both *val* and *hos* before notifying the customer, than to notify the customer about each subtransaction individually. This strategy assumes that the VE SoCom is responsible for performing actions to correct orders. ▌

3.4 Social versus Psychological Primitives

Some previous approaches, e.g., [Levesque *et al.*, 1990; Grosz & Sidner, 1990], attempt to reduce social constructs to psychological constructs. They do not have an explicit construct for commitments, but postulate mutual beliefs among the committing agents. However, Mutual beliefs require the agents to hold beliefs about each other to unbounded levels of nesting, which can be tricky [Singh, 1996a]. Also, mutual beliefs cannot be implemented except through additional simplifying assumptions, which is why the direct approach of using social constructs is more appealing. In fact, it is known that in settings with asynchronous, unreliable or unboundedly delayable communication, mutual beliefs can be obtained only if they are there from the start—i.e., the mutual beliefs are the invariants of the system [Chandy & Misra, 1986].

We conjecture that named groups and named commitments, which are reminiscent of contract numbers in business dealings, provide the necessary connections among the agents. This is a reasonable conjecture, because commitments and the groups they exist in can provide the requisite context that is *copresent* with all of the agents. Membership in a group can require mutual commitments, which can refer to each other (by using each other's names). Thus, the effect that traditional theories attempt to achieve by using mutual beliefs can be achieved without mutual beliefs, and without reducing social primitives to psychological primitives. We believe that with further technical development, this will prove to be an important point in favor of social commitments.

3.5 Implementation

The above view of commitments can thus lead to CISs that behave flexibly. In order to make the construction of such CISs equally flexible, we are developing a generic facility for commitment specification and management. This facility would allow the specification of CISs along with the social policies that apply within them. We provide a generic set of Java classes through which abstract CISs can be specified. These specifications include the different roles in a given CIS, and the capabilities and resources required to instantiate each role. These specifications also include the social policies—expressed in terms of roles—that apply within the abstract CIS. Essentially, these are the commitments that the role comes with. For example, the seller role presupposes that the seller will respond to requests for price quotes, and honor its quotes.

The abstract CISs are instantiated with concrete agents filling each role. The concrete agents may be individuals or groups. Recalling [Gasser, 1991], a concrete agent may fill in more than one role in an abstract CIS, and participate in more than one abstract CIS concurrently. The act of joining a CIS corresponds to creating commitments. The commitments associated with a role are schematic. Upon instantiation of the roles, these are instantiated into commitments by and toward concrete agents. Thus agents can thus autonomously enter into SoComs. Agents must make sure they have the capabilities and resources required to take on any additional role, and its concomitant commitments. Some of the inherited commitments might require overriding some prior commitments. For example, the Valvano agent must relax its refund policy when joining the above-mentioned VE.

Once the concrete CISs have been instantiated, any of the member agents can initiate an activity, which can trigger additional activities. The facility provides primitives through which agents can instantiate a CIS, create commitments within the context of a CIS, and satisfy or cancel commitments. The facility takes care of the bookkeeping required for these operations, and to ensure that the correctness condition is met. The underlying means of execution is based on a temporal logic approach, which extends the results of [Singh, 1996b], to provide primitives for coordinating heterogeneous activities.

4 Comparisons with the Literature

DAI Approaches. Gasser describes some of the sociological issues underlying multiagent systems [Gasser, 1991]. His notion of the multiple simultaneous roles played by social agents inspired part of our discussion above. Castelfranchi studies concepts similar to those here [Castelfranchi, 1993]. Our context groups generalize his notion of a *witness*. Castelfranchi distinguishes a notion of collective commitment, which is subsumed by our concept of commitment (through the orthogonal representation of the structure of multiagent systems). Tuomela develops an interesting theory of joint action and intention that bears similarities to collective commitments [Tuomela, 1991]. [Sichman *et al.*, 1994] develop a theory and interpreter for agents who can perform social reasoning. Their agents represent knowledge about one another to determine their relative autonomy or dependence for various goals. Dependence leads to joint plans for achieving the intended goals. This theory does not talk about commitments *per se*, so it is complementary to our approach. We also believe that our approach with its emphasis on structure and context can be married with that of [Sichman *et al.*, 1994] to lead to more sophisticated forms of social reasoning.

The approach of [Levesque *et al.*, 1990] requires the agents to have a mutual belief about their goals. Further, it hardwires a specific approach to canceling commitments (for joint intentions)—the participating agents must achieve a mutual belief that the given commitment has been canceled. The approach of

[Jennings, 1993] is closer in spirit to the present approach. Jennings postulates *conventions* as ways in which to reason about commitments. Thus, he can generalize on [Levesque *et al.*, 1990]. However, for teams, he requires a "minimum" convention, which recalls the approach of [Levesque *et al.*, 1990]. Jennings also requires a mental state as concomitant with a joint commitment. While we share many of the intuitions and motivations of [Jennings, 1993] (including applications involving heterogeneous information systems), we attain greater generality through the explicit use of the structure of multiagent systems. The agents always enter into commitments in the context of their multiagent system, and sometimes to that system. This has the pleasant effect that social concepts are not made dependent on psychological concepts. The multiagent system serves as the default repository for the cancelation and commitment policies, although these can, in several useful cases, be assigned to the member agents. We believe the relationship of our approach to open-nested transaction models and workflows will lead to superior multiagent systems for information applications.

Distributed assumption-based [Mason & Johnson, 1989] or justification-based [Huhns & Bridgeland, 1991] truth maintenance systems (DTMSs) are also germane. These systems help a group of agents revise their beliefs as a consequence of messages received. On the one hand, DTMSs can be given a knowledge-level characterization in terms of commitments; on the other hand, they can be used to implement some of the reasoning required in maintaining commitments.

DB and Groupware Approaches. A number of extended transaction models have been proposed, e.g., [Bukhres & Elmagarmid, 1996]. The extended transaction models allow partial results to be released, and then attempt to restore consistency through actions to compensate for the effects of erroneously completed actions. Some workflow scheduling approaches exist that provide functionality to capture control flow among tasks. The database approaches don't provide much support for the organizational aspects. For example, they ignore social commitments altogether.

Some of the groupware approaches, which study organizational structure, do not consider quite as rich a form of commitments as here. For example, information control nets are primarily geared toward control and data flow aspects [Nutt, 1993].

The notion of commitments finds applicability in some groupware tools. For example, [Medina-Mora & Cartron, 1996] shows how the flow of work in an organization is expressed through commitments in the ActionWorkflow tool. This tool comes with a fixed set of specifications from which the developer can choose. Although the participants can decide whether a given task was successfully performed, there is no notion of failure recovery, of commitments being canceled, or of commitment and cancelation policies. Still, we believe, this is an interesting system that shows how much can be achieved through the careful use of commitments.

5 Conclusions and Future Work

We sought to present the unifying principles behind commitment for single-agent and multiagent systems. Our approach marries insights from DB and DAI, to yield a framework for flexible, yet robust, cooperative information systems. Our approach makes the following contributions. It

- does not require translating commitments to psychological concepts, such as beliefs
- distinguishes between satisfied and inapplicable commitments
- incorporates policies to handle the creation, satisfaction, and cancelation of commitments
- relates commitments to organizational structure in a multiagent system
- shows how commitments are acquired by agents as a consequence of membership in a group.

A practical challenge is determining classes of commitments and policies that are more relaxed than the traditional approaches, yet can be efficiently implemented. Two other technical challenges are introducing temporal aspects into the language, and relating the development of commitments to decision theoretic analyses of rational behavior.

References

[Buchmann *et al.*, 1992] Buchmann, Alejandro; Özsu, M. Tamer; Hornick, Mark; Georgakopoulos, Dimitrios; and Manola, Frank A.; 1992. A transaction model for active distributed object systems. In *[Elmagarmid, 1992]*. Chapter 5, 123–158.

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

[Castelfranchi, 1993] Castelfranchi, Cristiano; 1993. Commitments: From individual intentions to groups and organizations. In *Proceedings of the AAAI-93 Workshop on AI and Theories of Groups and Organizations: Conceptual and Empirical Research.*

[Chandy & Misra, 1986] Davies, K. M. and Jayadev Misra; 1986. How Processes Learn. *Distributed Computing* 1:40–52.

[Davies, 1978] Davies, Charles T. Jr.; 1978. Data processing spheres of control. *IBM Systems Journal* 17(2):179–198.

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

[Gasser, 1991] Gasser, Les; 1991. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence* 47:107–138.

[Gray & Reuter, 1993] Gray, Jim and Reuter, Andreas; 1993. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann.

[Grosz & Sidner, 1990] Grosz, Barbara and Sidner, Candace; 1990. Plans for discourse. In Cohen, P.; Morgan, J.; and Pollack, M., editors, *SDF Benchmark Series: Intentions in Communication.* MIT Press, Cambridge, MA.

[Huhns & Bridgeland, 1991] Huhns, Michael N. and Bridgeland, David M.; 1991. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1437–1445.

[Jennings, 1993] Jennings, N. R.; 1993. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review* 2(3):223–250.

[Levesque *et al.*, 1990] Levesque, H. J.; Cohen, P. R.; and Nunes, J. T.; 1990. On acting together. In *Proceedings of the National Conference on Artificial Intelligence.*

[Mason & Johnson, 1989] Mason, Cindy L. and Johnson, Rowland R.; 1989. DATMS: A Framework for Distributed Assumption-Based Reasoning. In Gasser, L. and Huhns, M. N., editors, *Distributed Artificial Intelligence, Volume II.* Pitman/Morgan Kaufmann, London. 293–318.

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

[Nutt, 1993] Nutt, Gary J.; 1993. Using workflow in contemporary IS applications. Technical Report CU-CS-663-93, University of Colorado.

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

[Sichman *et al.*, 1994] Sichman, Jaime Simão; Conte, Rosaria; Demazeau, Yves; and Castelfranchi, Cristiano; 1994. A social reasoning mechanism based on dependence networks. In *Proceedings of the 11th European Conference on Artificial Intelligence.*

[Singh & Huhns, 1994] Singh, Munindar P. and Huhns, Michael N.; 1994. Automating workflows for service provisioning: Integrating AI and database technologies. *IEEE Expert* 9(5). Special issue on *The Best of CAIA '94* with selected papers from Proceedings of the 10th IEEE Conference on Artificial Intelligence for Applications, March 1994.

[Singh & Huhns, 1995] Singh, Munindar P. and Huhns, Michael N.; 1995. Cooperative information systems. Tutorials notes from conferences including the International Joint Conference on Artificial Intelligence (IJCAI), Montreal, 1995; the IEEE International Conference on Data Engineering (ICDE), New Orleans, 1996; and the European Conference on Artificial Intelligence (ECAI), Budapest, 1996.

[Singh, 1996a] Singh, Munindar P.; 1996a. A conceptual analysis of commitments in multiagent systems. Technical Report TR-96-09, Department of Computer Science, North Carolina State University, Raleigh, NC. `http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/commit.ps`.

[Singh, 1996b] Singh, Munindar P.; 1996b. Synthesizing distributed constrained events from transactional workflow specifications. In *Proceedings of the 12th International Conference on Data Engineering (ICDE).*

[Tuomela, 1991] Tuomela, Raimo; 1991. We will do it: An analysis of group-intentions. *Philosophy and Phenomenological Research* LI(2):249–277.

[Wittig, 1992] Wittig, Thies, editor. *ARCHON: An Architecture for Multi-agent Systems.* Ellis Horwood Limited, West Sussex, UK.

This article was processed using the LaTeX macro package with LLNCS style