9. SharedPlans in Electronic Commerce

Merav Hadad[1] and Sarit Kraus[1,2]

[1] Department of Mathematics and Computer Science Bar-Ilan University, 52900 Ramat-Gan, Israel.
E-Mail: hadad@cs.biu.ac.il
[2] Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA.
E-Mail: sarit@cs.biu.ac.il

9.1 Introduction

Rational agents often need to work together [356]. There are jobs that can not be done by one agent, because no one individual has sufficient competence, resources or information to solve the entire problem alone. In other situations, agents that work in the same environment may benefit from cooperation. A joint action by a team does not consist merely of simultaneous and coordinated individual actions to act together. A team must be aware of and concerned with the status of the group effort as a whole. To rectify this problem, it was proposed that agents should have a well-grounded and explicit model of cooperative problem solving on which their behavior can be based. Several such models have been proposed [375, 344, 311, 259].

In recent years, many researchers as well as commercial companies have attempted to create intelligent, agent-based markets or retail outlets on the Web (see section 9.3.2). However, these systems have not had the anticipated impact on the methods commerce implemented on the Web. Tsvetovatyy and Gini [678] claim that a main cause of this shortfall is the lack of automated purchasing and agent cooperation algorithms. They developed a general framework for an automated agent that buy and sell on the Web, but have not developed cooperative capabilities for these agents.

In our work, we designed a system based on the SharedPlans model of Grosz and Kraus [259], and we have developed a collaborative multi-agent system for buying and selling items such as clothes[1]. In our Web-based buying and selling environment, SharedPlans may be formed between agents belonging to the same enterprise to work together to maximize their enterprise's benefits and also among agents that are self-motivated and interested in collaboration because it may maximize their individual benefits. For example, suppose a buyer of one enterprise would like to buy an item from a seller of another enterprise. Even though each agent tries to maximize its enterprise's benefits and they have certain conflicting interests they have the same joint goal of the transaction taking place. In particular, a seller wishes

[1] Other implementations of a collaborative multi agent systems that are based on the SharedPlan model in other domains were presented in [394, 512, 662].

to have a noncompetitive relationship with its buyers. Today's sellers desire highly cooperative, long-term relationships with their buyers to maximize customer satisfaction and to increase their long-term benefits [216, 265].

When an automated seller interacts with a human buyer using SharedPlans is beneficial to both sides. The seller can, for example, work it with the buyer to identify an item which is relevant to his/her needs by maintaining the intentional context of their interaction. For instance, the intentional context for buying a CD to listen to a specific song is different than that of buying a CD since the user admires the singer or likes the music style of the CD. In the last case, if the CD is not available, the automated seller may offer another CD of the same style. In the second case, it may offer a different CD of the same singer, and in the first case suggests a different CD which includes the same song (possibly by a different singer).

In order to increase their benefits in the Electronic Commerce environment the agents need to plan their activities. For example, planning may be needed when a buyer of some enterprise detects a missing good in its stock. Thus, the buyer must form an individual plan to obtain the missing good (e.g. she may decide whether to buy or to produce the missing good and which resources to use, given her choice). The forming plan may be only partial. For example, she may have decided to purchase the missing good, but has not yet chosen from which supplier to buy it. In addition the buyer must take into consideration several constraints, for instance, in order to perform the purchase she must have enough money.

Since the Electronic Commerce environment is a dynamic environment, the agents must take into account the dynamic nature of plans. In this environment plans are developed over time. For example, frequently a buyer is able to state only her general needs, i.e., she is not able to characterize the exact details of the item she would like to buy. The seller can assist the buyer in defining her needs, using his information about available items and their properties. By providing such information the seller and the buyer develop a plan for the purchase over time. That is, the agents in this environment begin with partial plans and extend them until they become complete plans. In addition, in this environment, an agent's beliefs may be faulty or the world may change. For instance, a buyer and a seller may agree upon the date of the payment for a selected item. As this date approaches, the buyer may realize that she is unable to pay for the item as originally planned. Thus, while the agent is planning or is acting on the basis of a partial plan, partial plans may have to be revised. To address these needs, the SharedPlan formalization [259] provides for both individual and collaborative plans to be partial. Thus, using the SharedPlan formalization enables us to develop agents which are able to act in the dynamic Electronic Commerce environment where they are uncertain concerning their own actions and have incomplete information about the other agents and the environment.

Since this chapter focuses on the planning formalism of SharedPlan, we summarize the SharedPlan model's definitions in section 9.2. In section 9.3 we present the electronic commerce domain which we use in the implementation, explain why using SharedPlan is beneficial in electronic commerce domain and compare our approach with others. In section 9.4 we present a general system for implementing the SharePlan model, and demonstrate its usage in electronic commerce scenarios. Finally, in section 9.5 we review the contributions of this chapter.

9.2 The SharedPlan Model

When agents form teams, new problems emerge regarding the representation and execution of joint actions. To rectify these problems it was proposed that the agents use a model of collaborative planning. The SharedPlan model [259] has been proposed to support the design and construction of collaborative computer systems, including systems that are able to collaborate with one another [311], systems that support groups of people working together, and collaborative systems for human-computer communication [394, 550].

The SharedPlan formalization [259, 258] provides mental-state specifications of both *SharedPlans* and *individual plans*. SharedPlans are constructed by groups of collaborating agents and include subsidiary SharedPlans [394] formed by subgroups as well as subsidiary individual plans formed by individual participants in the group activity.

The SharedPlan of a set of agents depends upon the individual plans of its members. For an agent G to have an individual full plan for an act α, it must satisfy four requirements: (1) G must know how to perform α; i.e., it must have the constituents of the act α, (2) G must believe that it can perform the subacts (i.e, α's constituents) (3) G must intend to perform the subacts, (4) G must have a subsidiary (individual) plan for each of the subacts which are not basic actions[2].

SharedPlans differ from individual plans in requiring that the set of agents have *mutual belief* of these requirements. In addition, because SharedPlans are multi-agent plans, the subsidiary plans of a SharedPlan may be either individual or shared, depending upon whether they are formed by a single agent within the group or by a subgroup. The full group of agents must mutually believe that the agent or the subgroup of each subact has a plan for the subact. However, only the performing agent(s) itself needs to hold specific beliefs about the details of that plan.

This section discusses the SharedPlan formalism as defined in [259]. Because the formal plan definitions are complex, in the following section we will

[2] We assume that a basic level action is an action which does not involve more than one agent, which must be performed in one sequence and which there is only one way to perform it.

provide an informal example that motivates the definitions presented in this section. We will refer to this example throughout the chapter.

9.2.1 A Collaborative Trade Scenario

The scenario that is considered includes two enterprises. Each enterprise has two intelligent agents, a buyer and a seller. The job of the buyer agent is to obtain the missing goods from the stock of its enterprise. The job of the seller agent is to sell the enterprise's goods to the other enterprise through the enterprise's buyer agent.

The example that is used exemplifies situations in which, the buyer agent of the first enterprise has an individual plan to maintain the stock. Particularly, we will consider the collaborative planning that arises when the individual agent, the buyer of one enterprise, detects a missing good in its stock. We will denote this agent by Buyer1[3]. In her individual planning to obtain the missing good, she discovers that a SharedPlan is required. Figure 9.1 illustrates the subactions which are needed for a "maintain the stock" action.[4] Suppose, Buyer1 finds out that the seller of the second enterprise can help her to complete her planning to maintain the stock. We will denote this agent Seller2. We assume initially that Buyer1 and Seller2 agents agreed to jointly perform the common action "transact". They have identified a recipe for doing the action "transact", i.e., they have figured out the relatively high level description of how to do the action, but they have not yet worked out the lower-level details. For instance, they may not have decided how to do some of the subactions in the recipe or who will do them. The subactions of the action "transact" is portrayed in Figure 9.1. Suppose they decided that:

1. Seller2 will do the subaction find-the-item, e.g, will check if he has such an item in the catalog of his enterprise.
2. Buyer1 will do the verification, e.g, Buyer1 will check that Seller2 finds the exact item which she intends to obtain.
3. Seller2 will check if he has the item in stock, and determine the date that he can supply the item.
4. Buyer1 and Seller2 will decide together about the method of payment (e.g., how Buyer1 will pay for the item) and the price of the item.
5. Buyer1 will execute the payment.
6. One of the agents will deliver the item. We assume initially that they did not decide who will do the delivery.

---

[3] In the example, we will take the initiator of the joint action to be a female and the other participant to be a male, thus affording the use of the pronouns "she" and "he" in analyzing the example.

[4] Note that in the recipe for the single-agent action "maintain the stock" there is a multi-agent action "transact". Thus, performing "maintain the stock" requires cooperation with other agents.

Thus, Buyer1 and Seller2 must each form individual plans, Buyer1 forms plans for "verification", and "execution-payment"; Seller2 forms plans for "find-the-item", and "determine-date-supply". They do not need to know the complete details of each other's individual plans, but they need to prevent conflicts between these plans. For example, if it is decided that Seller2 will perform the subaction "execution-delivery" and it is decided that he will put the item in Buyer1's home at a specific time, Buyer1 cannot leave her home at that specific time in order to arrange the "execution-payment" subaction. Thus, as they develop their individual plans, choosing how to do actions and what resources to use, they must consider potential conflicts with each other and communicate if they detect a possible problem. In addition, Buyer1 and Seller2 together must form a shared, collaborative plan for the "determine-price-and-payment". The particular details of how they will do this must be mutually decided upon by an agreement of both of them. In forming their plans, Buyer1 and Seller2 may interleave planning and acting; hence, at any stage of their activity, their plans may be only partial. For example, Buyer1 decides to "borrow-money" for "execution-payment", but has not yet chosen a recipe for doing so. Alternatively, she may have chosen the recipe, but not yet decided how she will do some of the subtasks.

Two types of agents:
Seller: sells goods.
Buyer: obtains missing goods.
Single-agent-Action:
Maintain the stock:
- determine-the-missing-items.
- transact.

Multi-agent action:
Transact:
- find-the-item.
- verification.
- determine-date-supply.
- determine-price-and-payment.
- execution-payment.
- execution-delivery.

Fig. 9.1. Maintain-the-stock Action Example.

9.2.2 Definitions and Notation of the Model

Actions in the model are abstract complex entities that have associated with various properties such as action type, agent, time of performance, and other objects involved in performing the action. Following Pollack [531], the model uses the terms "recipe" and "plan" to distinguish between knowing how to do an action and having a plan to do the action. When agents have a SharedPlan to do a group action, they have certain individual and mutual beliefs about how the action and its constituent subactions are to be implemented. The term *recipe* [531, 395] is used to refer to a specification of a set of actions, which is denoted by β_i $(1 \leq i \leq n)$, the doing of which under appropriate *recipe-constraints*, denoted by ρ_j $(1 \leq j \leq m)$, constitutes performance of α.[5] The meta-language symbol R_α is used in the model to denote a particular recipe for α.

Recipes may include actions at different levels of abstraction and the parameters of an action may be partially specified in a recipe either in the library or in a partial plan. Thus, a recipe may include variables which are not instantiated (e.g., for the agent or time of an action) and constraints on these variables. However, for agents to have a complete plan, the parameters must be fully specified in a manner appropriate to the act-type.

The subsidiary actions β_i in the recipe for action α, may be either single-agent or multi-agent actions; all basic-level actions are single-agent actions. Likewise, a single agent "determine-price-and-payment" carried out only by Seller2 (e.g., Seller2 determines the price and payment by himself) is a different type of action from multiple agents "determine price and payment" together, (e.g., Seller2 and Buyer1 determine the price and payment together).

The intended actions that play a role in individual and collaborative plans are always planned and performed in some context. Various operators, functions, and predicates on actions as well as the plans that are formed for performing them need to refer to this context. The model represents two constituents of the context parameter. The first, Θ_α, includes a "constraints" component that encodes constraints on the performance of α. For example, Seller2's individual plan to "determine date supply" have the constraint of being done before a certain time or the constraint of not determining this date after a particular time. The second, IC_α, includes a representation of the intentional context in which an agent G is doing α. For example, if α is being done as part of doing some higher-level action $\mathcal{A}$, i.e., α is part of the recipe adopted in the plan to do $\mathcal{A}$, then IC_α encodes this fact.

Notation	Meaning	Comments
α	action	Also: β_i,β_r, β_k, δ, γ
IC_α	intentional context of α	
R_α	recipe for α	Also, R_β
G	agent	
GR	group	
Θ_α	constraints of α	Also, ρ_j

Table 9.1. Summary of notation used for special variables and constants.

[5] The indices i and j are distinct; for simplicity of exposition, we omit the range specifications in the remainder of this document.

Grosz and Kraus' plan definitions employ four different intention operators: **Int.To** and **Int.Th** represent intentions that have been adopted by an agent; **Pot.Int.To** and **Pot.Int.Th** are variations of the first two that are used to represent *potential* intentions.

Int.To and Pot.Int.To are used to represent an agent's *intentions to* do some action; Int.Th and Pot.Int.Th are used to represent an agent's *intention that* some proposition hold. The commonality between intending-to and intending-that is that both commit an agent not to adopt conflicting intentions [715] and constrain replanning in case of failure [84]. The significant distinction between them is not in the types of objects each relates, but in their connection to planning and in their different presumptions about an agent's ability to act within the boundaries of the intention.

An Int.To commits an agent to means-ends reasoning [84] and, at some point, to acting. In contrast, an Int.Th does not directly engender such behavior. Int.Th's form the basis for meshing subplans, helping one's collaborator, and coordinating status updates [85, 600, 375], all of which play an important role in SharedPlans; any of these functions may lead to the adoption of an Int.To and thus indirectly to planning and performing actions. In addition, an agent can only adopt an intention-to toward an action for which it is the agent.

The model distinguishes between five different types of plans: **FIP** for *full individual plans*; **PIP** for *partial individual plans*; **FSP** for *full SharedPlans*; **PSP** for *partial SharedPlans*; and **SP** for SharedPlans of indefinite completeness. When PIP or FIP exist for an agent, that agent has the collection of intentions and beliefs specified in the meta-predicate definition in [259].

The SharedPlan meta-predicate representing that a group of agents GR has a collaborative plan to jointly perform some action α. The meta-predicate FSP is used to represent the situation in which a group of agents has completely determined the recipe by which they are going to do some group activity, and members of the group have adopted intentions-to toward all of the basic-level actions in the recipe as well as intentions-that toward the actions of the group and its other members.

Partial SharedPlans, like their counterpart partial individual plans, differ from complete ones in four ways: (1) the agents may have only a partial recipe for doing the action; (2) they may have only partial individual plans or partial SharedPlans for doing some of the subsidiary actions in the recipe; (3) they may have only partial individual plans or partial SharedPlans for doing some of the contracting actions; and, (4) there may be some subactions about which the group has not deliberated and for which there is as yet no agent (individual or subgroup) selected to perform the subaction.

9.2.3 Complex Actions for Planning and Cultivating Process

In the SharedPlan formalization, means-ends reasoning is represented by the complex planning action **Elaborate_Individual** and the group plan-

Type	Notation	Meaning
Modal Operators	Int.To	intend-to
	Int.Th	intend-that
	Pot.Int.To	potential intention-to
	Pot.Int.Th	potential intention-that
	Bel	belief
	MB	Mutual belief
	Do	performance of action
Meta-Predicates (Plans)	FIP	full individual plan
	PIP	partial individual plan
	SP	SharedPlan
	FSP	full SharedPlan
	PSP	partial SharedPlan
Act-types for Planning Actions	Select_Rec	agent selects/extends recipe
	Select_Rec_GR	agt. group selects/ext. recipe
	Select_Agent	agt. group selects member for action
	Select_Subgroup	agt. group selects subgroup for action
	Elaborate_Individual	agent extends partial plan
	Cultivate	cultivate an intention-that
	Elaborate_Group	agt. group ext. partial SharedPlan
	Select_Rec_GR_Member	planning action for Select_Rec_GR
	Select_Agent_Member	planning action for Select_Agent
	Select_Subgroup_Member	planning action for Select_Subgroup
	Elaborate_Group_Member	planning action for Elaborate_Group
	Reconcile	reconcile new intention with old ones
Processes	CULTIVATE	process for *Cultivate*
	ELABORATE_GROUP_MEMBER	process for Elaborate_Group_Member
	SELECT_AGENT_MEMBER	process for Select_Agent_Member
	SELECT_SUBGROUP_MEMBER	process for Select_Subgroup_Member
	SELECT_REC_GR_MEMBER	process for Select_Rec_GR_Member
	RECONCILE	process for Reconcile

Table 9.2. Summary of notations.

ning activity is represented by the complex action **Elaborate_Group**. In order to develop mechanisms for expanding partial plans to more complete ones, Grosz and Kraus provide the complex planning actions **Select_Rec** and **Select_Rec_GR** which refer respectively to the act-types for the complex planning actions that agents perform individually or collectively to identify ways to perform (domain) actions. A basic claim of the SharedPlans formalization is that collaborative activity is rooted in the individual mental actions and domain actions of individual agents. This constraint holds of complex group planning actions as in domain actions. Thus, Elaborate_Group and Select_Rec_Group comprise individual planning actions of the group members. We use the complex actions **Elaborate_Group_Member** and **Select_Rec_GR_Member** to represent these individual actions, respectively. In addition, the SharedPlan formalization includes complex planning actions, **Select_Agent_Member** and **Select_Subgrp_Member**, for determining which agent(s) will do the constituent actions of the chosen recipe.

In multi-agent activities, participants not only do means-ends reasoning about their own actions, they also reason about how to coordinate with and support the actions of others in the group. These activities require plan-based reasoning that ensues from the participants' attitudes of intentions-that toward the actions of others and of the group as a whole. To handle this aspect of the dynamics of SharedPlan the complex planning action, **Cultivate** was introduced [260]. In particular, SharedPlans are motivated by intentions-that and thus Cultivate plays a central role in the architecture.

To assist the reader, Table 9.1 provides a summary of the notation used in the formalizations in this chapter. Table 9.2 lists the various operators and predicates used in the formal definition and provides an informal description of their meaning.

9.3 The Benefits of Using SharedPlans in Electronic Commerce

As we mentioned, the environment which we consider includes several enterprises, each with several kinds of goods which it sells to users or to other enterprises. Each enterprise has intelligent seller and buyer agents. The job of the seller agents is to sell the enterprise's goods to users or to other enterprises through their buyer agents. The job of a buyer agent is to obtain from other enterprises the goods that are missing from the stock of its enterprise. This environment requires complex SharedPlans of groups of more than two agents including both people and automated agents. However, for simplicity our current implementation considers an environment, in which each enterprise includes only one seller and one buyer (see Figure 9.2).

Agents that belong to the same enterprise can form SharedPlans to work together toward the shared goal of maximizing the benefits for their joint enterprise. SharedPlans also provide a beneficial framework for communication processes between different enterprises. Even though each agent has its own goals and in respect to certain issues (e.g. the price of the item) their interests may conflict, they have the same shared goal that the purchase will be carried out. The seller of one enterprise would like to sell an item to the buyer of another or to a user, and the buyer or the user would like to buy the item. Forming a SharedPlan may increase the benefits of both agents.

9.3.1 Means for Collaboration

As has been discussed above, SharedPlans in our environment may be formed between agents belonging to the same enterprise who try to maximize their enterprise's benefits and also among agents that are self-motivated and interested in collaboration because it may maximize their individual benefits. To provide the basis for more collaborative interactions, our system will include

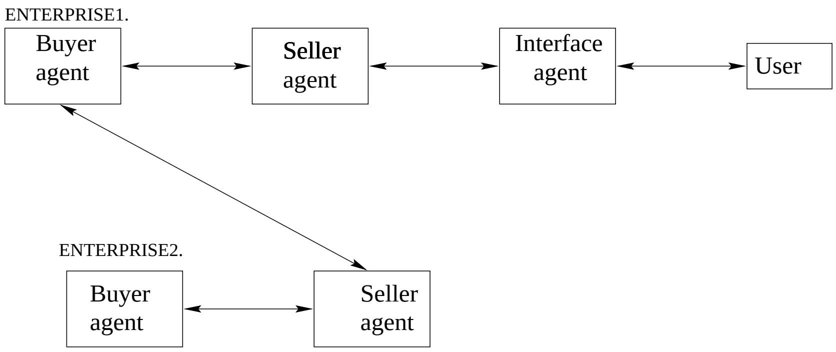

Fig. 9.2. The Electronic Commerce scenario

capabilities for helpful behavior and the avoidance of conflicting intentions. The importance of such capabilities in this environment and the way the SharedPlans formalization supports it are illustrated below.

The definitions of partial and full SharedPlans require each of the participating agents to intend-that the group action α be performed. Furthermore, the group must mutually believe that all members have such intentions-that the action be performed. In addition to explicitly representing each group member's commitment to the group's performance of α, this clause in the definition captures three key characteristics of collaboration: (1) agents avoiding the adoption of intentions that conflict with their doing α; (2) agents forming intentions to help each other in the performance of α; and (3) agents adopting intentions to communicate about their plan for doing α and its execution. The importance of these key characteristics in the Electronic Commerce domain, are illustrated by the following examples: In the first example, the buyer and seller agents engaged in collaborative activity must reconcile intentions and avoid conflicts among them. In particular, buyer and seller agents that work for the same enterprise will avoid taking resources from one another. For example, if a seller agent needs the company's largest truck to deliver some goods to a primary customer, then the buyer agent—wanting the seller agent to succeed—will not intend-to use this truck at the same time. The second characteristic may be illustrated by the following example. Frequently, a buyer will be able to state only his general needs. He will not be able to characterize the exact details of the item he would like to buy. The seller can assist the buyer in defining his needs, using her information about available items and their properties. By providing such information the seller can help making the purchase take place which is a shared goal of both buyer and seller. The next example exemplifies the importance of the third characteristic. As part of their recipe negotiation, a buyer and a seller will agree upon the supply date of a selected good. As this date approaches, the seller may realize she will be unable to deliver the product as originally planned. To address this problem, the seller communicates with the buyer, and to keep the collaboration active works with the seller to identify a new date or means of delivery.

The definitions of SharedPlans also stipulate that each participant have intentions-that toward the ability of other agents to carry out their parts of the group action, i.e., doing the constituent actions β_i required for doing α (according to the recipe they have adopted); intentions-that relate to ability are required both for individual-agent subactions and for multi-agent subactions. These intentions-that ensure that the subsidiary plans (individual and group) for doing the subsidiary actions are compatible. For example, the plan for the "transact" action which is discussed in section 9.2 includes Seller2's intention that [Int.Th] Buyer1 'be able to' do "verification", and "execution-payment", and Buyer1's intention that [Int.Th] Seller2 'be able to' "find-the-item" and "determine-date-supply". The Buyer1 and Seller2's intention that [Int.Th] provide the basis for more collaborative interactions between Buyer1 and Seller2 as described below.

Several axioms for the operator Int.Th are needed to support the roles of intending-that in ensuring that agents avoid conflict, assist each other, and provide status information when necessary. The formalization [260] have specified a set of conflict avoidance axioms that constrain an agents' adoption of intentions (both intentions-to and intentions-that) so that they do not simultaneously hold conflicting intentions. In addition, the formalization has developed a set of axioms to encode situations in which intending-that would lead agents to consider engaging in helpful behavior. We do not describe these axioms here for space limitations, but instead we discuss them informally.

One of these axioms states that if an agent has an intention-that toward some proposition that it believes is not currently true and the agent believes it is able to do some act γ that will bring about the proposition's holding, then the agent will consider doing γ. In particular it will adopt a potential intention to do γ, leading to deliberation about adopting an intention to do it, and, barring conflicts, lead to this becoming a full-fledged intention. This situation may be illustrated with the maintain-the-stock example. Suppose Buyer1 and Seller2 are doing the "transact" action together and Seller2 believes that Buyer1 will *not be able* to execute the payment and furthermore believes that he can take some action to remove the roadblock to his being able to do so (e.g., delaying the payment date of the good), then Seller2 will adopt a potential intention to do that action.

A second axiom provides for more indirect helpful behavior. This covers the case in which Buyer1 has an intention-to "transact" several goods until a specific date, but there are no such goods in the stock of Seller2's enterprise. Therefore, the buyer of Seller2's enterprise, Buyer2, will adopt potential intention-to "transact" in order to buy goods until this date. The axiom states that if an agent has an intention-that toward some proposition that it believes does not currently hold and the agent believes it is able to do some act δ that will enable another agent (or group of agents) to do an action γ that will bring about the proposition's holding, then the agent will adopt a potential intention to do δ.

A third axiom provides a basis for helpful behavior specifically in the SharedPlan context, i.e., for helping a collaborative partner. For example, if Seller2 needs to deliver some goods to Buyer1, and the buyer that belongs to Seller2's enterprise (we denote this buyer as Buyer2) needs to deliver some goods from the city of Buyer1's enterprise. Seller2 might offer to deliver these goods to Buyer1. The situation this axiom covers is one in which Buyer2 believes that the overall cost to him of the maintain-the-enterprise will be less if he does this delivery than otherwise.

As we mentioned, our environment includes both human and automated agents. Most of the current trader agents [659, 177, 594], which help users with all aspects of online shopping, do not keep track either of what the user is trying to acquire or of the dialogue context. As a result, trader agents do little to support user-agent collaboration on a transaction. Most of the work of keeping track of the context of their communication is thus left to the user. Those trader agents that do maintain some kind of history of what the user has done record a linear history list of the user's commands to the system. They do not represent the intentional structure of the dialogue nor track intentional state. As a result, they can not take the advantage of the structure of the user's work, and hence grow linearly with it. For example, the intentional context of buying a book for a present is different from the intentional context of buying a book for learning for an exam as well as buying a book for learning in the classroom. In the first case, if the book is not in the stock, the agent may suggest alternative books in the same price. But, in the second case the alternative option is suggesting a book with similar subjects. In the last case, since the user must have the specific book which he had indicated, the only alternative option is to inform the user of the date that this book will be in stock.

To provide the basis for more collaborative trader agents, we enable the users to communicate with the seller agents when SharedPlans are used for providing the collaborative task context. Our seller agents take an active role and can, for example, work with the users to identify a product which is relevant to their needs. Again, for space limitation we cannot demonstrate the efficacy of the human seller's collaboration.

9.3.2 Comparison with Alternative Implementation of the Electronic Commerce Domain

In this section, we compare our multi-agent system, for buying and selling, to alternative works which have created intelligent agent-based markets or retail outlets.

Takahanshi et al. [659] introduce an information tool called DION that collects shop and service information. This tool consists of several agents. One of them supports the user when creating queries. The other agents search and locate information, fetch the information and organize the search result. Their approach uses telephone-dictionary information as a well-organized index. However, their tool helps users find items to buy, and does not try to automate the process of buying and selling as we do.

Doorenbos et al. [177] design, implement, and analyze shopping agents that can help users with all aspects of online shopping. Their initial focus has been the design, construction, and evaluation of a scalable comparison shopping agent called ShopBot. Like DION, the ShopBot does not include any automated processes of buying and selling, and it focuses on extraction of information.

Schrooten [594] in his work presents an example of a pilot application concerned with the production of electronic consumer catalogs and their delivery to customers through the Internet. The technology behind the application is based on software agents. Schrooten's work illustrates new opportunities for electronic commerce and discusses the contributions of an agent-based approach, but he does not illustrate any cooperation model which we believe is necessary for the development of beneficial agents for electronic-commerce.

Chavez et al. [120] designed the Kabash system. Kasbah is a Web-based multi agent classified ad system where users create buying agents and selling agents to help transact goods. A user who wants to buy or sell goods creates an agent, gives it strategic direction, and sends it off into a centralized agent marketplace. Kasbah agents pro-actively seek out potential buyers or sellers and negotiate with them on behalf of their owners. Their work is focused on negotiation "strategies". That is, unlike our system, the buyer and seller agents in the Kasbah system do not include any planning capabilities or collaboration capabilities which are based on a cooperation model.

Another system is the Michigan Internet AuctionBot [741], which is an online auction server. When a user would like to sell an item it put it for sale through the AuctionBot. The user chooses a type for his auction from a selection of auction types and specifies its parameters (e.g., clearing times, permitted, etc.). AuctionBot manages the auction according to the user's specifications. As the Kasbah system the AuctionBot is only focused on negotiation skills (e.g., auctioning skills).

Klaus et al. [210] in their paper advocate the use of intelligent agents as a useful metaphor and as a software engineering methodology for the design and operation of virtual enterprises. They focus on how agents can support the cooperative process of setting up virtual enterprises through the Internet by performing tasks such as presentation, information retrieval and extraction, and the participation in auctions in electronic markets. In their paper they only offer a perspective of the high potential of agent-based technology, by presenting the main objectives of the research project AVE (Agent in Virtual Enterprises). The agents in AVE are designed according to the INTERRAP agent model. These agents include three layers. One of these layers is the cooperative planning layer (CPL) which extends the planning functionality of an agent to joint plans, i.e., plans by and/or for multiple agents that allows conflict resolution and cooperation. This cooperation is based on a general negotiation model. In our implementation, the cooperation is based on the SharedPlan model. Thus, our implementation includes more methods for achieving cooperation.

Albayrak et el. [7] present the REkos project (intelligent agents for realization of electronic market services). The REkos project consists of creating and realizing tools for the implementation of cooperating complex services. Their architecture includes management module processes. During the execution, the management module accesses the knowledge base. In doing this the agent is controlled by its intentions and goals. The management module includes several components. One of the components is the cooperation manager, which supervises the execution of the cooperation protocols and scripts, in which the agent is involved. Another module that REkos architecture includes is the intentionality module. In the intentionality module, goals and intentions of the agent are represented. Although REkos architecture includes a cooperation manager, this cooperation manager refers to the communication between the agents. That is, REkos architecture does not consist of SharedPlans which plays a central role in the cooperation of our implementation. As a result, REkos architecture does not provide, for example, capabilities for helpful behavior and avoidance of conflicting intentions.

9.4 The General SharedPlan System

In this section, we describe a general system for the implementation of a multi-agent collaborative system that is based on the SharedPlans formalization which is discussed in section 9.2. We demonstrate the behaviour of the system using a specific implementation of agents that sell and buy in an electronic-commerce environment. The system was implemented using Alegro Common-Lisp running on a Solaris Workstation.

In order to enable the agents to act in a realistic environment we provide each agent in the system with the ability to plan several actions simultaneously. Each agent in our system is able to plan in parallel several actions, $(\alpha_1, \ldots, \alpha_n)$, where its plan for each such action may be only partial (see Figure 9.3). For example, the agent can begin to perform several "transact" actions with several different enterprises, negotiate the price and then select the best bid. We first describe the data structures of the system, and then discuss the system's processes. The system demonstrates the ability to use the general architecture of collaborative agents based on the theory.

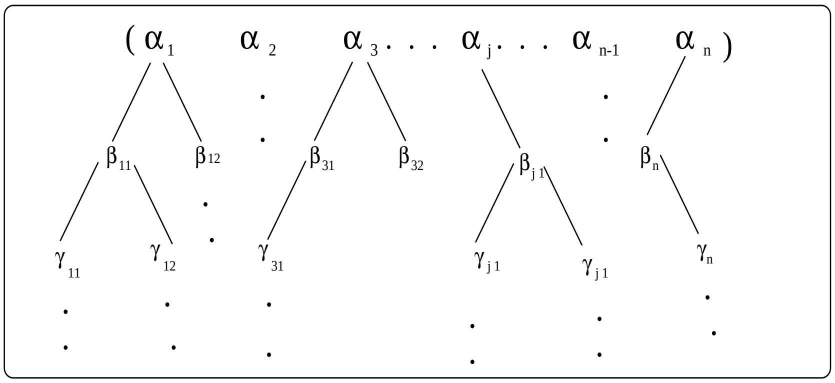

Fig. 9.3. Example of several actions which agent G plans simultaneously. While G detects that it is not able to continue working on the plan of action α_j, G can continue working on the plan for another action, α_i (i $\neq$ j), until it will be able to continue working on α_j's plan. $\beta_{j,k}$ are the subactions of R_{α_j} and $\gamma_{j,l}$ are the subactions of $R_{\beta_{j,l}}$.

9.4.1 The Information and Data Structures of the Agent

To help explain the necessity of information and data structures which are given in this section we will examine a scenario based on the example which is presented in section 9.2. Suppose that Buyer1 and Seller2 from the "transact" action example (see Figure 9.1), have agreed on the supply date of the selected good, T_{supply}, and they agreed that Seller2 will do the delivery and Buyer1 will execute the payment when she will get the good. Thus, Buyer1 has intention to [Int.To] do the "execution-payment" subaction at T_{supply}, intention that [Int.That] Seller2 will do "execution-delivery" subaction at T_{supply}, a belief that she will be able to do so, and an individual plan for doing so; likewise, Seller2 has an intention to [Int.To] "execution-delivery" at T_{supply}, an intention that [Int.That] Buyer1 will do the "execution-payment" subaction at T_{supply}, a belief he can do it, and an individual plan for doing so. We also assume that Buyer1 and Seller2 do not have enough knowledge in order to continue working on their plan for the "transact" action, until the supply date, T_{supply}, will be reached. Thus, Buyer1 and Seller2 do not do anything with respect to their SharedPlan of "transact" in the time interval ($T_{present}$, T_{supply}), while $T_{present}$ denotes the present time. Suppose that another buyer of another enterprise, Buyer3, also wants to buy an item from the enterprise of Seller2. Seller2 can begin a new "transact" action with Buyer3.

When Seller2 begins this new "transact", action he has to maintain his intention to "execution-delivery" at T_{supply}, and his intention that [Int.That] Buyer1 will do "execution-payment" subaction at T_{supply} in the context of his SharedPlan with Buyer1. Seller2 cannot adopt new intentions that are conflicting with the intentions that he currently has (unless he reconciles the intentions and decides to discard the old one). He also has to remember to continue working on his joint plan with Buyer1 for the "transact" action at T_{supply}, and he has to continue this planning from the point where he had stopped coordinating his planning with Buyer1. Thus, the agent needs structures which will help him to "remember" what he did not finish with respect to the "transact" action, and structures for maintaining its intentions and the planning actions, and a method for enabling him to perform the "execution-delivery" subaction at T_{supply}. At the time T_{supply} Seller2 has to perform the "execution-delivery" but he also has to remember to continue its planning for the new "transact" action it has begun.

Suppose that Buyer1 and Seller2 decided together that Buyer1 would pay cash to Seller2 at T_{supply} when performing "execution-payment," but that Buyer1 did not succeed to get enough cash by that date. Because Buyer1 knows that Seller2 expects to get the cash at T_{supply} she has to communicate to Seller2 and inform him about the problem and together they can change their plan. However, in some situations they may discard the SharedPlans and cancel the transaction. In order to re-plan the subaction "execution-payment", the agent needs to save all the relevant information which will help her to return to the "execution-payment"'s planning.

To allow such complex multi-agent partial planning, each agent uses structures and data which are discussed briefly below. In particular, each agent in the system applies three major modules: (1) Domain module; (2) Planning module; and (3) Communication module. The **domain module** contains the information which is known by the agent about its specific domain as beliefs of the agent, domain actions and the recipes library. The **planning module** contains structures which are used by the agent while it plans its actions. This module consists of: (a) domain actions: $(\alpha_1, \ldots, \alpha_n)$ that the agent plan in parallel; (b) the agenda which includes all the types of the intentions which have been adopted by the agent; (c) a schedule and a queue: the schedule contains basic domain actions and the queue contains all the processes that the agent needs to activate; and (d) the context which refers to the intentional context, IC, in the formalism. The agent associates such context with the following objects: with each intention in the agenda, with each basic-level domain action in the schedule and with each planning action in the queue. The **communication module** handles the communication processes which are different from the other processes in the system.

9.4.2 Actions and Recipes

Actions, in the model (see section 9.2), have been described as abstractions with some properties such as action type, agents, time and other objects involved in performing the action, likewise, constraints. This approach works equally well for *action types*. That is, the action types include the same properties of actions. But, they also include the information as to whether the action is basic, as well as action preconditions, effects and results.

A *recipe* has been described as a specification of a set of action-types and constraints. The recipe's set of actions may be at different levels of abstraction. It also has been noted in the model that a recipe may include variables and constraints on these variables. The following sections describe the structures of an action type and a recipe as has been used in our system.

Action Type Structure. As observed above, the action includes an action type as one of its properties. In our system, the action types are also associated with various properties, but unlike actions the action types in the system are abstract. For example, the action-type "transact" is an abstract entity which represents a general activity of buying an item. When agents act to execute purchase, then this actual activity is referred to as the "action transact". In general, an action type structure consists of: (1) name of the action; (2) parameters of the action; (3) some constraints; (4) the action consequences; (5) action level; and (6) result variables.

```
(setq transact (make-action-type
 :name 'transact
 :agents '(A1 A2)
 :time 't1
 :params '(item-description)
 :param-constraints 'nil
 :time-constraints '()
 :agent-constraints '(
     (not (eq (?enterprise ?A1) (?enterprise ?A2))
     (member ?A1 ?capability-sell)
     (member ?A2 ?capability-buy))
 :preconditions '()
 :results '(
     (increase-present-amount ?item-description ?A1)
     (increase-money ?A2))
 :effects '(
     (decrease-money ?A1)
     (decrease-present-amount ?item-description ?A2))
 :basicp 'nil
 :result-var '() ))
```

Fig. 9.4. Action Type Example.

In Figure 9.4 the action type structure of "transact" is presented. The variables and functions are prefixed by ?; we use this prefix to refer to the values of the instantiated variables and the values are returned by functions. The action's name slot contains the name of the action-type (e.g., "transact"). The action type structure includes three types of parameters' names: the first type are agents, the second is the time of the action, and the third type includes all the other parameters' names that are involved in performing the action. The action-type's constraints may be of two types: One type, can restrict the actions' agent, actions' parameters, and the actions' time, i.e., the *param-constraints*, *time-constraints*, and *agent-constraints* fields in Figure 9.4 respectively. The action consequences also may be of two types: The first type is its result, i.e., the 'intended' consequences of the action, e.g., in the "transact" example the intended result of the buyer is to increase the amount of the products in the stock. The second type is side effects: in this aspect the agent does not intend the consequences, e.g., in the "transact" example, the agent has to pay for the product, as such she loses money. The action level is a 'boolean' value that defines whether the action is basic or complex. Sometimes, the values of the action parameters can be identified by other subactions. For example, in order to do "execution-the-payment" the agents have to identify the price of the item, but the price of the item is the value that is returned by the action "determine-price-and-payment". The result-var field consists of the names of the parameters which will contain the values to be returned to the agents that perform this action type after the performance of this action type.

Recipe Structure. In our system, recipes are associated with action-types. An agent may know several recipes for the same action type, but for each action the action type is unique. Figure 9.5 presents a recipe for the the multi-agent action "transact". In general, a recipe structure consists of: (1) unique identifier; (2) a set of subactions; (3) result variables; (4) constraints; (5) new parameters; (6) the recipe consequences (see, for example, Figure 9.5).

The *applic-constraints* field refers to the applicable constraints. Performing the subactions of the recipe under the appropriate constraints may cause changes in the domain as well as in the agents mental states. The recipe *consequences* describe those changes. In the "transact" example, if the buyer using the 'I do the delivery' recipe for the execution delivery subaction, it has the consequence of making his car occupied, and using the 'hire carrier' recipe has the consequence of spending money.

9.4.3 Getting from Individual Plan to SharedPlan

There are several situations in which a recipe for a single-agent action includes multi-agent subactions, as in the "maintain-the-stock" example (which is discussed in section 9.2.1). Thus, while an individual agent detects that she cannot complete her individual plan alone, she can join other agents and form a SharedPlan. The agent which detects the need for a SharedPlan becomes the organizer. Being the organizer does not yield any privileges in deciding on recipes, roles etc.

```
(make-recipe
 :action-type transact
 :name 'transact1
 :applic-constraints '(?catalog-number (? 1))
 :subactions '(
    0. (find-the-cloth ?A1 (? 1) ?t1 (?the-cloth))
    1. (verification ?A2 (? the-cloth) ?t2 (?verify))
    2. (determine-date-supply ?A1 (? the-cloth) ?t4 (? date))
    3. (determine-price-and-payment ?A1 (? the-cloth) ?t5 (?price ?payment))
    4. (execution-payment ?A1 (?price ?payment) ?t6)
    5. (execution-delivery ?A2 (?date) ?t7))
 :new-params '(the-cloth verify date price payment)
 :param-constraints '()
 :time-constraints '(
    (≤ (end-time ?t1) (start-time ?t2))
    (≤ (end-time ?t2) (start-time ?t3))
    (≤ (end-time ?t5) (start-time ?t6))
    (≤ (end-time ?t4) (start-time ?t7)))
 :agent-constraints '(
    (member ?A1 ?capability-sell)
    (member ?A2 ?capability-buy))
 :consequences '()
 :result-var '() )
```

Fig. 9.5. Multi-agent Recipe Example.

The roles of the organizer are to: (1) identify other community members who will be willing to participant in the SharedPlan and send them a message that SharedPlan is required; (2) determine a unique name for the common action; and (3) determine the "common-recipe-tree-path". The "common-recipe-tree-path" of a subaction includes all the actions leading to the need for the given action from the first time that any other agent or the organizer in the system began to plan this action. For example, if in an agent's plan δ is a subaction in a recipe R_β for β and β is a subaction in a recipe R_α for doing α and α is the highest level action, then the "common-recipe-tree-path" of δ is (α, β, δ)). The "common-recipe-tree-path" is used by the agents, as unique identifiers in the messages that are exchanged between them and in order to uniquely identify the parameters of the actions.

9.4.4 The Processes in the System

The Grosz and Kraus' model (see section 9.2.3) has demonstrated the need for planning actions, and offered a sketch of some of these actions. Although all the necessary planning actions have been defined, the detailed design of each of them is not obvious. In this section, we specify the system's processes which are consistent with the theory.

The Controller Process. Figure 9.6 shows the main constituent steps of the controller process of agent G. It is comprised of three major constituents:

1. Check for messages, and allocate any newly received messages to the relevant processes.
2. Check the schedule. If it is time to run a process associated with a domain action, do so.
3. Choose α_i from the actions-list which consists the domain actions, $(\alpha_1, \ldots, \alpha_n)$, that G plans in parallel (see Figure 9.3) and then check the queue of α_i. If possible, run the next process in queue(α_i). The processes of the queue are associated with planning actions.

Assumption:
Agent G is currently working on action $\alpha_j \in (\alpha_1, \ldots, \alpha_n)$.
We denote the agenda of the action α_j by *agenda(α_j)*. agenda(α_j) includes all the types of the intentions which have been adopted by G in order to perform the action α_j. We denote the queue of α_j by queue(α_j). queue(α_j) contains all the processes that G needs to activate in order to perform the action α_j.
Controller-Loop:

Check for arriving messages; send each message to its relevant process;
While time of first action in schedule is greater or equal to the current time **do:**
    execute the corresponding action;
If agenda(α_j) is not empty, **then:**
    If queue(α_j) is not empty **then:**
        Choose an item from queue(α_j) to establish;
    Else, the queue(α_j) is empty and
    the time of the first action on the schedule has not arrived yet **then:**
        If the actions-list includes other actions **then:**
            abandon α_j, select another action α_i $(i \neq j)$ from actions-list;
        Else, the actions-list does not include other actions **then:**
            Wait, until a new message will arrive;
Else, if G believes that her plan to do action α_j is complete, **then**:
    delete α_j from the actions-list;
    If actions-list includes other actions **then:**
        abandon α_j, select another action α_i $(i \neq j)$ from actions-list;
    Else, the actions-list is empty **then:**
        Wait, until a new message will arrive;
Start the Controller-Loop again;

Fig. 9.6. Pseudo code for the Controller Process.

We will discuss each of the above tasks of the controller in the following: As the first step in Figure 9.6, the controller process handles the inter-agent communication. Messages are exchanged between the planning process of a sender agent, and those of the recipient. The need to send a message usually arises during the execution of a joint planning process, and thus messages are sent out by the planning processes themselves. Making sure that it reaches the right process is a non-trivial task. The decision on what the relevant process is for each message received, making this process aware of the message, is the controller's job.

A process is associated with each of the actions the agent has committed to (that is, with each of the actions for which it has adopted either an Int.To or Int.Th). A process is initiated when its Int.To (or the Int.Th) is created, and spends the lifespan of the Int.To (resp. Int.Th) either running or waiting. However, there is a difference in the way basic level domain actions and the way complex domain actions are handled: a basic domain action is associated with a process 'make-basic-action' and is maintained in the schedule. A complex domain action is associated with a Select-Rec planning action and is kept in a queue with the other planning actions[6] All the scheduling is handled by the controller process: The controller loops:

1. Check the schedule and if the time of the earliest 'make-basic-action' process in the schedule has arrived the controller initiates this process. In particular, if the specific time of the earliest basic domain action has not arrived the agent can continue its planning by starting the controller loop again.
2. Check the queue and if the queue contains any process, the controller takes out the first process from the queue and initiates this process in order to continue the planning of the agent.

Processes for Planning. In order to develop mechanisms for expanding partial plans to more complete ones, the model provides complex planning actions. In our system, each planning action is implemented via a process. Each process is associated with either Int.To or with Int.That. Below we describe the processes in our system and demonstrate the usage of these processes in the Electronic-Commerce environment. For limited space reasons and since the processes are complex, we will briefly describe a selected subset of them[7].

Processes that are associated with Int.To. Since our Electronic-Commerce is a dynamic environment, our system requires a mechanism for efficiency backtracking and modularity. Thus, the elaborating process, which is used in our system for extending partial plans to complete ones, initiates several processes (see Figure 9.7) such as a process for selecting recipes which identifies ways to perform (domain) actions.

Elaborate-Individual: this process is needed when an intention to perform a single agent action α exists but, the plan for that action is incomplete. Elaborate-Indiv is responsible for the completion and the construction

[6] The processes in the queue that are associated with complex actions are handled differently from the processes in the schedule that are associated with basic-level actions: First, the processes in the schedule are scheduled for a specific time and are ordered by an increasing order of the performance time, while planning processes are kept in a queue according to their arrival time. Second, while there is a separate sub-queue for each action in the actions-list, there is only one schedule for all the actions.

[7] The other complex processes of the system are described in [267].

of the plan for α. For example, in order to perform the single agent action "maintain-the-stock", Buyer1 initiates an Elaborate-Indiv process for "maintain-the-stock".

Select-Recipe-Individual: the main responsibilities of the Select-Recipe-Individual process include: (a) selection of an applicable recipe; (b) for each single agent subaction in the recipe being used, ensure that an attempt will be made to execute it by initiating an Elaborate-Indiv process for this subaction; (c) for each multi agent subaction in the recipe being used, ensure that an attempt will be made to execute it by forming an intention-that the proposition "Initiate-SP" of that subaction will hold. For instance, in the "maintain-the-stock" example Buyer1 forms an intention-that the proposition "Initiate-SP(transact)" will hold in order to perform the multi-agent subaction "transact".

Processes that are associated with Shared Plans. In this section we describe the complex processes **Elaborate_Group** and **Select_Rec_GR** which play a role in expanding partial SharedPlans to complete ones.

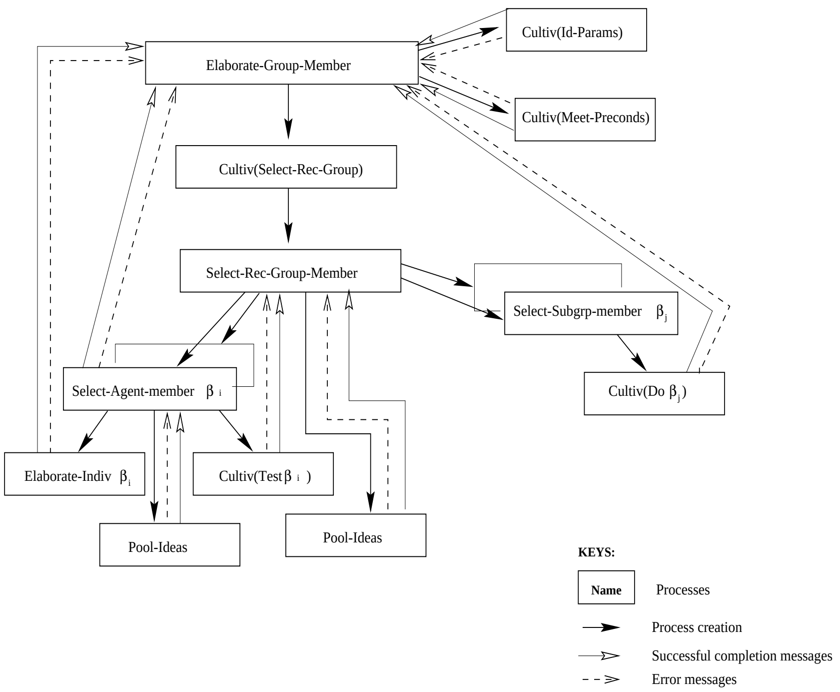

Fig. 9.7. Elaborate-Group-Member and Select-Rec-Group-Member

Elaborate-Group-Member: A basic claim of the SharedPlans formalization is that collaborative activity is rooted in the individual mental actions and domain actions of individual agents. This constraint holds of complex group planning actions as well as of domain actions. Thus, Elaborate_Group (see 9.4.4) comprises individual planning actions of the group members. We use the process Elaborate-Group-Member to represent these individual actions. Elaborate-Group-Member processes are the multi-agent counterparts of Elaborate-Indv processes. That is, they are responsible for building up plans for multi-agent actions. The Elaborate-Group-Member process must perform each of the following tasks:

1. *Make sure that α's parameters have been identified by initiating a cultivate process to identify parameters* (see section 9.4.4). For example, the parameter which has to be defined in the multi agent subaction "transact" which is included in the recipe of "maintain-the-stock" example, are the products that Buyer1 intends to buy. In this case, the parameter is identified by the cultivate process of Buyer1 and is given by Buyer1 to Seller2, but there may be situations in which the group members jointly identify it.
2. *Verify that all the preconditions of α have been met by initiating cultivate process for meeting the preconditions* (see section 9.4.4). For example, the precondition for doing the "determine-price-and-payment" subaction is that Buyer1 knows exactly what she wants to buy. In this case, Buyer1 checks if the precondition is satisfied and only then they can perform "determine-price-and-payment" subaction.
3. *Reconcile the potential intention that the group perform α with the intentions already held.* Reconciling a potential intention involves making sure that no conflicting intentions are already being held, and then replacing the potential intention-that with an intention-that. If there is an existing conflicting intention, a decision should be made whether to keep the old intention, or to discard it and to adopt the new one [492]. The Reconcile-int-that is the major locus for agents to assist one another. For example, in our domain, the buyer and seller agents that work for the same enterprise will avoid conflicts in resource use (and hence conflicting intentions).
4. *Initiate a process to select and setup a recipe for α, this process is described below.*
5. *Inform the parent process of its success or failure, as needed.* For example, in "maintain-the-stock" action, if the multi agent action "transact" fails, then the job of the Elaborate-Group-Member process of each participant is to inform its parent, the Cultivate process of the failure. This will lead to backtracking or to cancelation of the action performance.
6. *Deal with failure and success messages sent by other processes* (mostly its children).
7. *Inform the collaborating agents of its success or failure, as needed.*

8. *Deal with failure and success messages sent by other agents.* For example, in the "transact" action when Seller2 finishes the execution of "find-the-item," the Elaborate-Group-Member process of Seller2 will send a message to Elaborate-Group-Member process of Buyer1, informing Buyer1 if the execution of "find-the-item" has succeeded or failed.

Figure 9.7 presents briefly the relationships and the communication between the processes associated with a SharedPlan. For space limitation reasons, we only discussed the Select-Rec-Group-Member process (see [267] for a detailed description and examples of the system.)

Select-Rec-Group-Member: The group recipe selection process will cause individual agents to invoke the Select-Rec-Group-Member process. Select-Rec-Group-Member processes are the multi-agent counterparts of Select-Rec process. That is, they are responsible for the selection and initiation of the recipes for multi-agent actions. The main responsibilities of the Select-Rec-Group-Member process include:

1. Selection of an applicable recipe.
   The selection of an applicable recipe is more complicated than the individual case because of the need to coordinate with other agents to reach agreement. For example, in the multi agent action "determine-price-and-payment", although Buyer1 and Seller2 have the common general goal that the transaction will succeed, the interest of the buyer is that the price will be low while the interest of the seller is to agree upon a high price. In this case the agents may use negotiation in order to solve the conflict [357].
   Thus, to select the applicable recipe the following steps are executed:
   a) Each agent selects applicable recipes from its library that it prefers to use. Then, it sends these recipes to the other participants of the action.
   b) Each participant collects the proposed recipes.
   c) The agents applied an agreed upon decision making process (currently they randomly choose one, but negotiation or voting can be applied)[8].
2. For each single-agent domain subaction in the recipe being used, ensure that attempts will be made to select an agent that will execute it, by initiating a Select-Agent-member process which is responsible for the selection of the agent which will perform the single-agent subaction. In the "transact" example, although it is clear which agent will execute the subactions "find-the-item" "verification" and "execution-the-payment", it is not clear who, for example, will execute the "determine-date-supply" or "execution-delivery". This will be done through the Select-Rec-Group-Member processes of Buyer1 and Seller2.

[8] A more complex decision-making scenario, is one in which no single agent has a complete recipe. The agents work together to combine pieces of recipes that they discover individually.

3. For each multiple-agent domain subaction of the recipe being used, ensure that an attempt will be made to select the subgroup which will perform the subaction, by initiating a Select-Subgrp-member process which is responsible for the selection of the agents which will perform a multi-agent subaction.
4. Inform the parent process, Elaborate-Group-Member, of success or failure, as required.

Figure 9.7 illustrates how these tasks are performed by creation of processes, and the communication between these processes.

Processes that are associated with Int.That. **Cultivate:** The intentional attitude, intending-that, was introduced into the SharedPlan formalization to account for the commitment participants in a group activity make to one another's actions and to their joint activity. Intentions-that, like intentions-to, serves both to constrain the intentions an agent adopts and to affect (indirectly) its plan-based reasoning. In particular, the Cultivate process is the general process which is active whenever an agent has an intention-that. The Cultivate process determines the actions that an agent might take as a result of having an intention-that toward a proposition, i.e., creates potential-intentions to do them, as a result of having an intention-that toward a proposition. The following section describes briefly some of the propositions associated with the cultivate in the context of SharedPlans.

1. **Id-params:** The role of the cultivate process, which is associated with an Int.Th the proposition Id-Params(α) holds, is to ensure that the agent will identify the parameters of the act it performs. In order to verify this, the cultivate process, first, checks if the agent already knows the parameters. If the agent knows the parameters, then the cultivate process checks if the agent plans to identify these parameters[9]. If the agent does not have a plan for identifying these parameters the cultivate process initiates the Id-Params process. For instance, in the "transact" example the parameter that has to be identified in the single agent action "execution-the-payment" is the price of the item. In this case the parameter is identified when Buyer1 and Seller2 perform the multi-agent subaction "determine-price-and-payment", i.e., in this situation the group members jointly identify the parameter.
2. **Meet-Preconds:** The role of the cultivate process which is associated with an intention-that Meet-Preconds is to make sure that the preconditions will be satisfied. If the cultivate process reveals preconditions that are not satisfied, it activates the Meet-Preconds process. For instance, suppose that the precondition for the subaction "execution-delivery" is

[9] As described in [267] in several cases the parameters may be identified by other subactions

"the product existing in Seller2's enterprise" and suppose that it is decided that Buyer1 will perform the subaction "execution-delivery". In this case, although the action is performed by Buyer1 it is the role of the cultivate process of seller2 to make sure that the precondition of "execution-delivery" is satisfied. If Seller2 finds out that the preconditions cannot be satisfied he will update Buyer1.

3. **Initiate-SP:** When an individual agent detects that she cannot complete her plan for an action α alone, she can join other agents to form a SharedPlan by forming intention-that the proposition Initiate-SP will hold. The role of the cultivate process, which is associated with an Int.Th the proposition Initiate-SP(α), is to initiate the cultivate process which is associated with Do (this process is described below). For example, if Buyer1 detects that she cannot complete her plan for "maintain-the-stock" alone, then she joins Seller2 to form a SharedPlan.
4. **Do:** The intention-that toward the proposition Do in the formalism describes the intention of the agents to begin to do an action α together. The roles of the cultivate process which is associated with such an intention-that are:
   a) Establish that all the group members of the multi-agent action α hold the intention-that for doing α, and establish a mutual belief among the members of the group that each one will be a participant in the group doing α. That is, each agent in the group has to send a message informing its commitment to join in α performance. In addition a similar commitment message must be received from all the group members. This is done by the Establish-MB process that is described in [267]. For example, in the "transact" action this cultivate process of Buyer1 sends a commitment message that she joins the "transact" action to Seller2 and Seller2 and waits for a similar message from Seller2. Seller2 sends a commitment message that he joins the "transact" action to Buyer1 and waits for a commitment from Buyer1. Then, Buyer1 and Seller2 can hold the intention-that for doing the "transact" and establish a mutual belief that each of them is committed to the group performance of "transact".
   b) In order to form a SharedPlan (for doing α) group members must form a full SharedPlan to do the Elaborate-Group action. The cultivate process which is associated with the proposition Do also initiates the agent's adoption of an intention-that the group will undertake this planning activity. Since the collaborative activity is rooted in the individual mental actions the cultivate process which is associated with an intention-that Elaborate-Group, will cause individual agents to activate an Elaborate-Group-Member process; that is, the participants must each have their own internal process to complete the partial shared plan.

Figure 9.8 presents the creation of these processes and communication between them.

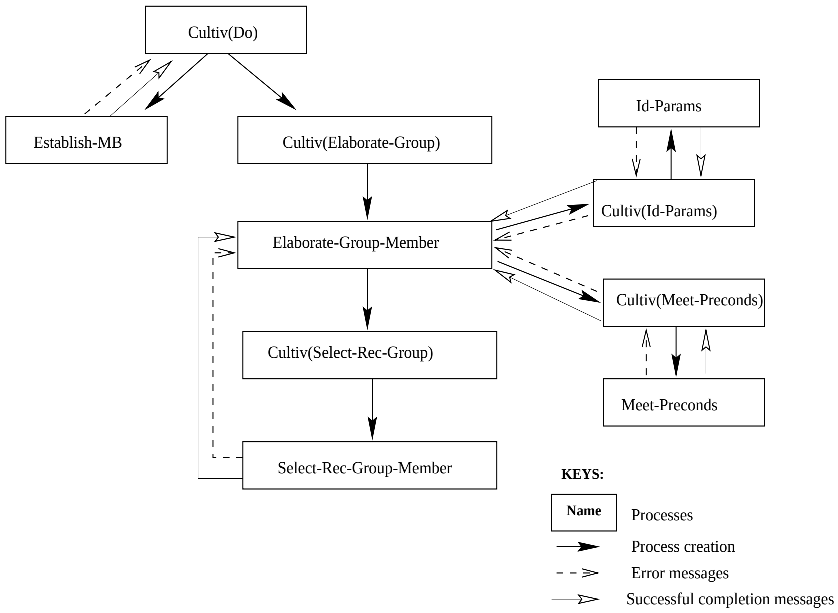

Fig. 9.8. Cultivate Do

Our SharedPlan system for buying and selling items which consists of the above planning processes provides automated purchasing and agent cooperation algorithms. Thus, in contrary to other agent-based markets or retail outlets on the Web (see section 9.3.2), which focus on either extraction of information or negotiation “strategies”, our system supports cooperative interactions of buying and selling goods. Using the SharedPlan formalization enables us to develop agents which are able to act in the dynamic Electronic Commerce environment where they are uncertain concerning their own actions and have incomplete information about the other agents and the environment.

9.5 Conclusion

In this chapter we focused on two major issues. First, we demonstrated the efficacy of collaboration in the Electronic Commerce environment and compared our implementation with other implementations for this environment. In this environment SharedPlans may be formed between agents belonging to the same enterprise to work together to maximize their enterprise's benefits and also among agents that are self-motivated and interested in collaboration because it may maximize their individual benefits.

Second, we have designed and implemented a collaborative multi-agent system based on SharedPlan formalization for selling and buying goods such as clothes and furniture on the Web. The Web-based buying and selling system requires complex SharedPlans of groups of more than two agents including both people and automated agents. Our Web-based buying and selling system demonstrates the usefulness of models of collaboration for multi-agent systems and tests the formalization.

Acknowledgement:

This research was supported in part by the National Science Foundation grant number IIS9724937.