

Towards a General Implementation of SharedPlans

Minh Hoai Nguyen and Wayne Wobcke

School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia

ABSTRACT: SharedPlans is an agent teamwork model which provides a formalisation of the conditions under which a group of agents has a collaborative plan. This paper describes some important steps towards a general implementation of SharedPlans theory. The implementation yields a flexible framework for building systems of collaborative agents.

INTRODUCTION

Building a group of agents that can work effectively as a team is more than just merely putting a number of individual agents together. Agents working in a team require abilities to plan, communicate and coordinate with each other. Furthermore, systems in which agents are simply equipped with precomputed coordination plans will fail to work in complex, dynamic environments. Thus it is extremely difficult to design and build a system of collaborative agents from scratch.

To see the difficulties in developing team-based applications, consider the scenario where a team of several scouting helicopters and an infantry platoon need to move to a battlefield, following Tambe [Tam97]. A sub-team of helicopters is responsible for scouting the path ahead while the infantry platoon needs to construct a bridge across the river. In addition, the infantry platoon needs to wait for the completion of scouting and bridge building activities before starting reallocation. This scenario illustrates several difficulties. First, the involvement of many agents complicates the coordination of team activities. Second, the existence of uncertainty, such as the enemy troop locations, requires agents to be equipped with flexible plans. Third, agents are possibly faced with incomplete and inconsistent information. Furthermore, many potential situations could lead to a failure of the team activity. For example, a helicopter agent might think a common plan is established but, in fact, it is not. Thus the agent might fly off to the battlefield alone leaving its teammates behind. Another example is when the team goal fails because the infantry platoon fails to construct the bridge, but because the scouting team is not informed appropriately, it continues with the plan unnecessarily. Thus the team of agents need to communicate with one another to synchronise actions, monitor team goals on behalf of the other agents and sub-teams, and coordinate with one another to establish and abandon team goals as appropriate.

One attempt to formalise a notion of collaborative activity is the SharedPlans theory presented by Grosz and Kraus [GK96, GK99], who proposed a specification of the conditions under which a group of agents has a shared plan. The formalisation gives conditions on the mental attitudes an agent must have to engage in collaborative activity. It also identifies the responsibilities and commitments of agents acting in a group activity. Furthermore, SharedPlans theory also provides an indication of how to decompose complex group actions into individual or simpler group actions.

Several applications have been built based on the theoretical foundation of SharedPlans, including an e-commerce system, Hadad and Kraus [HK99], a distance learning tool, Ortiz and Grosz [OG02] and a human computer interaction system, COLLAGEN, Rich and Sidner [RS97]. However, these applications are hard-coded implementations of the theory for specific problems, and there is no general framework that supports SharedPlans theory.

Therefore, there is an emerging need for building such a framework. This paper describes our initial attempt in this direction. Our system captures most of the concepts formalised in SharedPlans, such as shared and individual plans, partial and full plans. Furthermore, our framework supports rapid development and code reusability.

The remainder of this paper is organised as follows. The first section describes the SharedPlans theory in more detail. The second section discusses the structure and features of our framework. An evaluation of our framework will be described in the third section. Finally, we review and compare our system to related work.

SHAREDPLANS THEORY

SharedPlans theory is a formalisation of the mental attitudes of agents engaging in group activities. In SharedPlans theory, a group of agents have a collaborative plan when they each hold certain beliefs, desires and intentions. Thus the formalisation attempts to define some complex concepts, such as full shared plans and partial shared plans, based on these basic mental attitudes. The formalisation is given in first order logic enhanced with several primitive predicates, modal operators, meta-predicates and action functions. Some axioms also govern the commitments and behaviour of agents.

This section provides a brief summary of SharedPlans theory. Before the definition of shared plans can be understood, some supporting concepts such as 'mutual belief', 'intention to' and 'intention that' need to be described.

Mutual Belief

A group of agents are said to have *mutual belief* about a proposition α if each agent believes α , each agent believes the other agents believe α , each agent believes that any other agent believes all other agents believe α , etc. The beliefs of each agent about any other agent's beliefs may be nested to any arbitrarily large depth.

Action, Recipe and Plan

Actions are abstract complex entities. Some examples of actions are 'cook' and 'shoot'. Actions can be basic or complex. Complex actions consist of several other sub-actions which could, in turn, be basic or complex. For example, 'Cook Pasta' is a complex action which comprises several sub-actions like 'Boil pasta', 'Drain pasta', 'Mix pasta with sauce'.

A *recipe* for an action is a term which refers to a set of sub-actions together with their constraints and orders of execution designed to achieve the action. For each action, there might be none, one or more than one applicable recipe. For instance, the action 'Go to Melbourne' might have two different recipes: the first might be 'Catch a cab to the airport then catch a plane to Melbourne', the second could be 'Catch a bus to central station and catch a train to Melbourne'. The concept of recipe in SharedPlans is analogous to the concept of plan in traditional artificial intelligence. The plan concept in SharedPlans, however, is quite different. Plans in SharedPlans refer to recipes with some appropriate beliefs and intentions.

Intention

Intention is a unique mental attitude of an agent which is regarded as the commitment of the agent to some choice of action. According to Bratman [Bra90], intention has three functional roles. First, prior intentions frequently pose problems for means-end deliberation. For example, if an agent in Sydney intends to go to Melbourne tomorrow, he must gradually fill in his plan by figuring out how to get there. Second, prior intentions constrain the adoption of new intentions that conflict with the existing ones. For instance, an agent who intends to stay home to study for an exam cannot consistently adopt a new intention to go to the cinema with friends on the same night. Third, intentions control the conduct of agents; an agent eventually acts on his intentions.

The SharedPlans theory distinguishes between two kinds of intentions: *intention to* (IntTo) and *intention that* (IntThat). IntTo is an intention to perform an action, similar to Bratman's conception. IntThat is used to represent an agent's expectation that some proposition will hold or some actions will be performed (possibly by other agents). IntThat is similar to IntTo in the sense that it rules out the adoption of conflicting intentions and it constrains replanning in case of failure. There is, however, a significant difference between IntThat and IntTo. IntTo commits an agent to means-end reasoning and acting. In contrast, IntThat does not necessarily entail this commitment.

SharedPlans enforces some constraints on agents' beliefs and commitments if they have some intentions. If an agent intends to do a basic level action, the agent must believe she can do the action and commit herself to doing the action. If the agent intends to do a complex action, she must have a recipe for the action and intend to do all constituent sub-actions. The recipe might be partial. In this case, the agent must intend to elaborate the recipe.

Shared Plans

A group of agents are said to have a *shared plan* for doing an action if they mutually believe that all members of the group are committed to having the action done. In addition, there exists a recipe such that the group mutually believe the need to perform all sub-actions. Furthermore, each agent must believe every sub-action is catered for by a capable agent or sub-group of agents.

A more formal definition of a full shared plan is briefly given here. Let $FSP(P, GR, \alpha, T_p, T_\alpha, R_\alpha)$ denote that a group Gr has a shared plan P at time T_p to do action α at time T_α using recipe R_α .

$FSP(P, GR, \alpha, T_p, T_\alpha, R_\alpha)$ holds if and only if the following conditions are satisfied:

1. The group Gr has a mutual belief that all members are committed to (IntThat) the group success of doing α .
2. The group Gr has a mutual belief in the intention to perform the recipe R_α and the need to perform the constituent sub-actions in R_α .
3. For each sub-action β in R_α , there is an agent or a sub-group of agents GR_k which has an individual plan or shared plan to do the sub-action β . Everyone else in the group must believe that GR_k can do the sub-action using an appropriate recipe (however, other agents are not required to know the recipe). Moreover, every agent must commit to (IntThat) the success of GR_k in doing β .

SharedPlans theory also provides a definition of partial plans. Partial plans are plans in which the recipes for the actions might be incomplete or some sub-actions have not been assigned to any agent or any sub-group of agents. In the case of a partial plan, the group must have a full plan for elaboration of the plan to a full plan.

STEPS TOWARDS AN IMPLEMENTATION OF SHAREDPLANS

This section describes our implementation of SharedPlans. First, the language, platform and agents' structure will be discussed. Then we explain how SharedPlans theory is mapped to our system.

Language and Platform

Our framework is built on top of JACKTM Intelligent Agents [AOS05a] and Java. JACK is a toolkit which supports the development of agent-based systems. JACK is capable of generating Java code from an agent-oriented design. JACK is used for the implementation of the individual agents, including their beliefs, for event processing and inter-agent communication, and to implement the processes described below. Our agents do not use the plan library for representing recipes; rather recipes are represented in a logical language and, when adopted, are executed by an intention process.

Note that JACK provides an additional library component called JACKTeams [AOS05b] which supports team-oriented programming. However, JACKTeams is not used in our framework. This is because the structure and coordination mechanism of teams created using JACKTeams are different from those of the teams that we are building and are less flexible than those based on SharedPlans. In JACKTeams, team members are controlled and directed by team agents. Team agents are unique entities which are not part of the teams. Team members in JACKTeams are not necessarily aware of one another. In contrast, our framework creates decentralised teams in which team members are not commanded by any external entity. Team members must know one another in order to participate in team activities.

Structure of Agents

Figure 1 depicts the high level view of events flowing among processes inside each agent. There are four main types of process, namely monitor processes, message receiver processes, elaborator processes and intention processes. We now describe each of these processes in more detail.

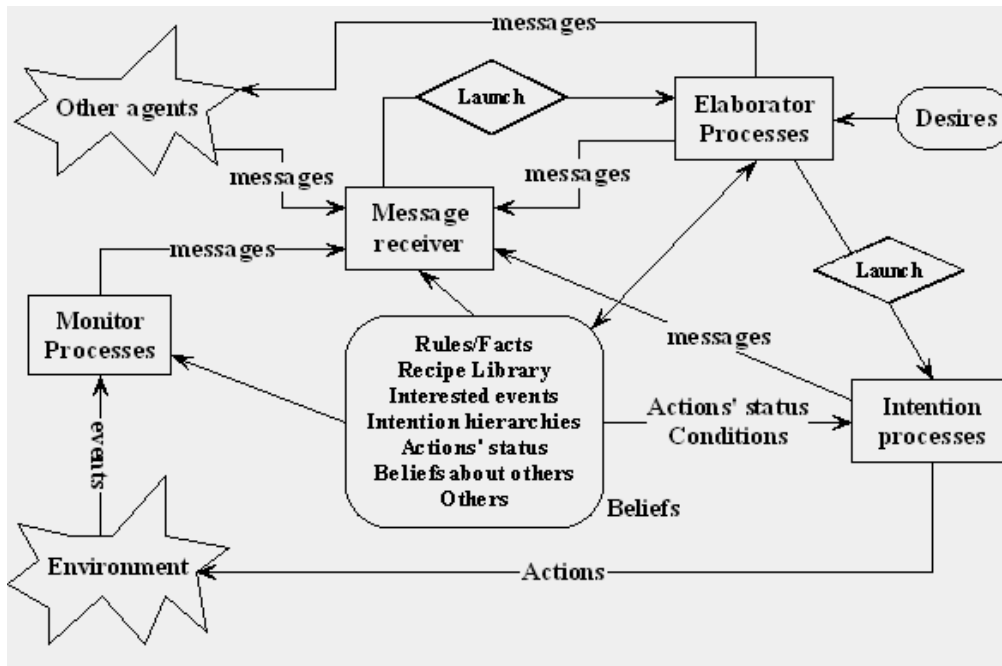


Figure 1. Internal structure of agents

The message receiver accepts external messages from other agents or internal messages from monitor processes, elaborator processes and intention processes. Upon receiving a message, the message receiver seeks the most appropriate elaborator process to handle the message. It does so by looking up the explicit rules in the belief set. The most appropriate elaborator process will then be launched and the responsibility of handling the message will be passed to it. The message receiver can invoke as many elaborator processes as needed to handle messages concurrently.

The elaborator processes are responsible for planning and forming intentions of agents. Upon being launched, elaborator processes investigate the messages assigned to them and reason about agents' beliefs and desires. After that, different elaborator processes might take different actions. Some elaborator processes might simply update the belief set and terminate. Some others might launch several intention processes to execute the intended actions. Some elaborator processes could indirectly create some peer and child processes to handle their assigned messages with them. They do so by creating and posting appropriate messages to the message receiver. Of course, there also can be processes which involve all the tasks mentioned above.

Intention processes are created by elaborator processes to perform intended actions. The actions here could be external actions which relate to the environment such as 'go' and 'drive'. The actions could also be internal actions such as elaboration or planning actions. For external actions, the intention processes directly execute the action. The intention processes execute internal actions by sending appropriate messages to the message receiver. An intention process constantly checks the agent's beliefs to see if it is the right moment to execute an action. The right moment could be a boolean combination of triggering events and time events. Dropping an intention will result in the termination of all related intention processes. Consequently, all related actions being executed or being intended to be executed will be aborted.

Monitor processes constantly monitor the environment. They have a reasoning capability of their own, though limited. They can access the belief set and determine what events the agent is currently interested in. If a monitor process detects an interesting event, it sends appropriate messages to the message receiver, which is responsible for invoking relevant elaborator processes to handle the event. This occurs through the use of JACK's mechanism for triggering plans.

Mapping from SharedPlans

In SharedPlans theory, a group of agents have a shared plan if certain conditions about intentions, individual beliefs and mutual beliefs hold. In our system, establishing a shared plan requires an equivalent set of conditions. Some intermediate processing steps are applied to transform the set of conditions in SharedPlans theory to the set of conditions used in our system. First, the mutual belief conditions found in the theory are reinterpreted as a set of individual beliefs and communication requirements. From that, a set of conditions for having a shared plan from the point of view of an individual agent is derived. This is contrast to the set of conditions for having a shared plan from the external perspective of an omniscient being that has all information about all the agents including the agents' beliefs. The final set of conditions obtained is used to guide reasoning procedures in elaborator processes.

One of the advantages of SharedPlans is the inclusion of partial plans in the formalisation. With partial plans, the recipes might be incomplete or some sub-actions not resolved. Our system also caters for the case of partial plans. For example, an agent might think a partial shared plan is established even though some sub-actions have not been assigned to anyone or any subgroup. The difference between having a plan (either partial or full) and not having a plan is the adoption of intentions to do sub-actions. Once the agent thinks a plan is established, she will invoke processes to do the sub-actions assigned to her. With partial plans, the agent might start executing her assigned sub-actions while some other unresolved sub-actions are still being elaborated.

The way the elaborator processes work is inspired by the conditions of IntTo and IntThat in SharedPlans. IntTo and IntThat lead agents to do some main activities such as means-end reasoning, committing to group activities, avoiding conflicts and satisfying mutual beliefs. Elaborator processes for complex actions reason about how to achieve the actions and form appropriate intentions to do constituent sub-actions. Agents are not only committed to the success of their own tasks in group activities but also to those of others. This can be illustrated by the fact that agents take into account the other agents' activities in planning their actions. IntTo and IntThat also rule out the possibility of adopting new intentions which conflict with existing ones. The agents' belief sets maintain hierarchies of intentions and provide appropriate information for elaborator processes to avoid creating incompatible intentions. In order to satisfy mutual belief requirements, elaborator processes issue appropriate messages to communicate with other agents.

The commitments of agents in executing intended actions are exhibited in the intention processes. In SharedPlans theory, once agents intend to do some actions, they must commit to doing the actions. For instance, if an agent intends to go swimming at 5pm, she will indeed go swimming at 5pm unless she revokes her intention. In our system, if an elaborator process thinks a new intention is formed, it launches a relevant intention process. Each intention process will execute the intended action when the right moment comes unless the intention is dropped or suspended before this time. Intention processes in our system demonstrate the commitments of agents in performing intended actions.

Although SharedPlans relies heavily on the notion of mutual belief, no indication for how mutual beliefs can be achieved is provided. Using the definition given is not a practical approach for representing mutual beliefs. In practice, reasoning agents (including humans) attain mutual belief based on some simplified versions of the above definition obtained by making additional assumptions about the environment.

In our system, we make some assumptions relating to communication and trust. First, the communication channel is reliable. In the other words, an agent Alice sending a message to another agent Bob can assume that Bob receives the same message within a reasonable amount of time after the message is sent. Conversely if Bob does not receive any message from Alice, Bob can assume that Alice does not send him any message. Another assumption of our system is that an agent can trust and does trust other agents' words about their beliefs. In the other words, Alice would trust what Bob says about Bob's beliefs and Bob believes that Alice trusts what he tells her about his beliefs. The above assumption does not require total trust of one agent in other agents' abilities. For instance, the fact that Bob sends a message 'I can do α ' to Alice does not entail that Alice believes Bob can do α . What the above assumption implies is that Alice would believe that Bob believes Bob can do α .

With the assistance of the above assumptions, from the point of view of an individual agent, mutual belief about a proposition α in a group \mathbf{Gr} can be achieved if:

- i. She, herself, believes in α .
- ii. She tells everyone in the group \mathbf{Gr} that she believes in α . Moreover, in every message she sends, she includes a list of recipients to whom she sends the same message, and the list includes everyone in the group \mathbf{Gr} .
- iii. She is told by everyone in the group \mathbf{Gr} that they believe in α . Moreover, the list of recipients of each message which she receives contains everyone in the group \mathbf{Gr} .

Agents in our system employ this new set of conditions instead of the official definition in reasoning activities pertaining to mutual beliefs.

EVALUATION

Our implementation closely follows SharedPlans theory. This is demonstrated in the previous section where a mapping from SharedPlans theory to our implementation is outlined. As a more direct implementation of SharedPlans, our system inherits most of its advantages. Some advantageous features of SharedPlans are the inclusion of both full and partial plans, the mechanism for decomposing complex actions and the decentralised team structure.

Our system is not a hard-coded solution for any specific problem. It is a reusable framework which facilitates the rapid development of team-based applications in a wide range of problem domains. The development process using our framework involves defining the recipe library and agents' capabilities. Developers also need to identify the initial beliefs of the agents about their teammates' abilities. Agents' beliefs about the capabilities of others can be updated and learned over time. At this point, the agents are ready to be integrated with the JACK platform. Using our framework, developers can focus more on designing their systems instead of worrying about how agents can coordinate and communicate with one another.

As a demonstration, a prototype application has been built using the framework. The application involves a team of three scouting helicopters and an infantry platoon. The mission is to get the majority of the infantry platoon to the battlefield; the mission is still considered successful even if some scouting helicopters or individual soldiers are shot down. Upon receiving a request to join the mission, each agent needs to engage in activities to establish a shared plan. These activities include establishing the mutual belief that everyone commits to the mission, deciding on a common recipe and assigning sub-actions to individual agents or sub-teams. Agents use elaborator processes to engage in these activities. In the prototype, the agents agree on a common recipe which is composed of four sub-actions: *BuildingBridge*, *Moving*, *Scouting1* and *Scouting2*, where *Scouting2* is backup plan for *Scouting1* and should only be executed if *Scouting1* fails.

RELATED WORK

In this section, we compare our framework to some other implementations of SharedPlans as well as several other frameworks for collaborative systems.

Currently, there are several existing implementations of SharedPlans. These include an electronic commerce system, Hadad and Kraus [HK99], a collaborative interface for distance learning (DIAL), Ortiz and Grosz [OG02] and a multi-agent system for collaboration of heterogeneous groups of people and computer systems (GigAgent), described in Grosz, Hunsberger and Kraus [GHK99]. SharedPlans is also used as the basis of the COLLAGEN dialogue system, Rich and Sidner [RS97]. However, these systems are all hard-coded implementations of the theory for specific problems.

More generally, there are some agent frameworks that support the development of collaborative systems, such as JACKTeams [AOS05b], ZEUS [NN98] and the Open Agent Architecture [MCM99]. However none of these support SharedPlans theory directly. The team structures and coordination protocols between team members in these systems are therefore different from our system. Teams in JACKTeams have a hierarchical structure with the existence of independent team entities. Agents in JACKTeams do not need to reason about their teammates, as team coordination is controlled by the

team agents. ZEUS agents work in a more distributed fashion. Agents in ZEUS can be provided with different coordination protocols drawn from a standard library. However, there is no ZEUS protocol which supports SharedPlans. The Open Agent Architecture is designed to integrate heterogeneous agents. In the Open Agent Architecture, agents do not need to know each other in order to work together, and coordination and communication are handled by a special facilitator agent.

Probably the closest work to our system is STEAM [Tam97]. STEAM is influenced by both SharedPlans theory and the Joint Intention theory of Cohen and Levesque [CL91]. STEAM, however, does not follow the formalisation of SharedPlans closely. The reasoning processes and mental attitudes of STEAM agents are steered and governed by Joint Intention theory rather than by SharedPlans. STEAM only uses a few aspects of SharedPlans such as the possibility of partial plans and the hierarchical structure of intentions.

CONCLUSION AND FUTURE WORK

SharedPlans is an important agent teamwork model which provides a theoretical foundation for team-based applications. However, most existing applications of SharedPlans are just *ad hoc* implementations for specific problems. This paper has briefly described the SharedPlans theory and its advantages. We also discussed the structure and implementation of our flexible, reusable framework for SharedPlans.

SharedPlans provides a formalisation of the conditions under which a group of agents has a collaborative plan; it, however, does not directly address some computational issues which are essential for such a framework, including communication, monitoring and coordination. In future work, we will examine how to incorporate additional components into the framework to address these issues in a modular, robust way.

REFERENCES

- [AOS05a] JACKTM Intelligent Agents Agent Manual. Version 5.0. Agent Oriented Software Group, June, 2005.
- [AOS05b] JACKTM Intelligent Agents Teams Manual. Version 5.0. Agent Oriented Software Group, June, 2005.
- [Bra90] Bratman, M.E. (1990) What is Intention? In Cohen, P.R., Morgan, J. & Pollack, M.E. (Eds) *Intentions in Communication*, pp. 15–31, MIT Press, Cambridge, MA.
- [CL91] Cohen, P.R. & Levesque, H.J. (1991) Teamwork. *Nous*, **25**, 487–512.
- [GHK99] Grosz, B.J., Hunsberger, L. & Kraus, S. (1999) Planning and Acting Together. *AI Magazine*, **20**(4), 23–34.
- [GK96] Grosz, B.J. & Kraus, S. (1996) Collaborative Plans for Complex Group Action. *Artificial Intelligence*, **86**, 269–357.
- [GK99] Grosz, B.J. & Kraus, S. (1999) The Evolution of SharedPlans. In Wooldridge, M. & Rao, A. (Eds) *Foundations and Theories of Rational Agency*, pp. 227–262, Kluwer Academic Publishers, Dordrecht.
- [HK99] Hadad, M. & Kraus, S. (1999) SharedPlans in Electronic Commerce. In Klusch, M. (Ed.) *Intelligent Information Agents*, pp. 204–230, Springer-Verlag, Berlin.
- [MCM99] Martin, D.L., Cheyer, A.J. & Moran, D.B. (1999) The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence*, **13**, 91–128.
- [NN98] Nwana, H.S., Ndumu, D.T., Lee, L.C. & Collis, J.C. (1998) ZEUS: A Toolkit for Building Distributed Multi-agent Systems. *Applied Artificial Intelligence*, **13**, 129–185.
- [OG02] Ortiz, C.L. & Grosz, B.J. (2002) Interpreting Information Requests in Context: A Collaborative Web Interface for Distance Learning. *Autonomous Agents and Multi-Agent Systems*, **5**, 429–465.
- [RS97] Rich, C. & Sidner, C.L. (1997) COLLAGEN: When Agents Collaborate with People. *Proceedings of the First International Conference on Autonomous Agents*, 284–291.
- [Tam97] Tambe, M. (1997) Agent Architectures for Flexible, Practical Teamwork. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 22–28.