Integrating High-Level and Detailed Agent Coordination into a Layered Architecture [*]

XiaoQin Zhang, Anita Raja, Barbara Lerner
Victor Lesser, Leon Osterweil, Thomas Wagner
Department of Computer Science
University of Massachusetts at Amherst

June 28, 2000

1 Introduction

Coordination, which is the process that an agent reasons about its local actions and the (anticipated) actions of others to try to ensure the community acts in a coherent fashion, is an important issue in multi-agent systems. Coordination is a complicated process that typically consists of several operations: exchanging local information; detecting interactions; deciding whether or not to coordinate; proposing, analyzing, refining and forming commitments; sharing results, and so on. We argue that facets of these different operations can be separated and bundled into two different layers.The lower-layer pertains to *feasibility* and *implementation* operations, i.e., the detailed analysis of candidate tasks and actions, the formation of detailed temporal/resource-specific commitments between agents, and the balancing of non-local and local problem solving activities. In contrast, the upper-layer pertains to *domain specific* coordination tasks such as the formation of high-level goals and objectives for the agent, and decisions about whether or not to coordinate with other agents to achieve particular goals or bring about particular objectives. Detailed domain state is used at this level to make these high-level coordination decisions. In contrast, decisions at the lower-level do not need to reason about this detailed domain state. However, reasoning about detailed models of the performance characteristics of activities, such as their temporal scope, quality, affects of resource usage on performance, is necessary at this level. In this view, the layers are interdependent activities that operate asynchronously.

[*] Effort This material is based upon work supported by the National Science Foundation under Grant No. IIS-9812755 and the Air Force Research Laboratory/IFTD and the Defense Advanced Research Projects Agency under Contract F30602-97-2-0032. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory/IFTD, National Science Foundation, or the U.S. Government.

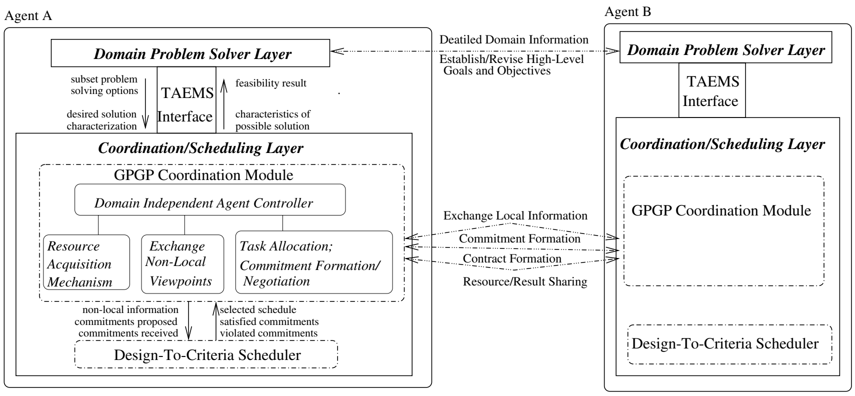
Agent A
Domain Problem Solver Layer
subset problem solving options
desired solution characterization
TAEMS Interface
feasibility result
characteristics of possible solution
Coordination/Scheduling Layer
GPGP Coordination Module
Domain Independent Agent Controller
Resource Acquisition Mechanism
Exchange Non-Local Viewpoints
Task Allocation; Commitment Formation/ Negotiation
non-local information
commitments proposed
commitments received
selected schedule
satisfied commitments
violated commitments
Design-To-Criteria Scheduler
Deatiled Domain Information
Establish/Revise High-Level Goals and Objectives
Exchange Local Information
Commitment Formation
Contract Formation
Resource/Result Sharing
Agent B
Domain Problem Solver Layer
TAEMS Interface
Coordination/Scheduling Layer
GPGP Coordination Module
Design-To-Criteria Scheduler

Figure 1: Integrating Coordination Agent Framework

In this paper, we describe the layered coordination model and introduce a general agent architecture based on the model (Section 2). In Section 3, we explore the layered model by integrating the Design-to-Criteria (DTC) [3] scheduler and the Generalized Partial Global Planning (GPGP) [2] coordination system as the lower layer with the Little-JIL framework as the higher layer.

2 Integrating Coordination Approaches

Figure 1 presents a general agent framework based on the layered model.The domain problem solver layer models the domain problem, manages the system state, reasons and plans on how to solve problems, and establishes the performance criteria for the agent.Different domain problem solvers may use different description languages, modeling structures, reasoning and analyzing strategies to solve problems. The coordination/scheduling layer evaluates the feasibility of performing goals/subgoals recommended by the domain problem solver layer, and, based on detailed resource constraint analysis, sets up the detailed temporal sequence and choice of local activities so that the multi-agent system meets its performance objectives. It has the following functions:

- **Reasoning about the feasibility of activities**
- **Choosing from and sequencing possible activities**
- **Assisting the task allocation**
- **Assisting the resource allocation**

The TÆMS task modeling language [1] is a domain-independent framework used to model the agent's candidate activities. It is a hierarchical task representation language that features the ability to express alternative ways of performing tasks, statistical characterization of methods via discrete probability distributions in three dimensions (quality, cost and duration), and the explicit representation of interactions between tasks.

Figure 5 contains an example of a TÆMS task structure. The TÆMS framework serves as a bridge that we use to connect the Domain Problem Solver Layer and the domain independent Coordination/Scheduling Layer.

This agent framework works as follows: the domain problem solver analyzes its current problem solving situation and establishes high level goals it want to achieve; also through the communication with other agents, it may decide some goals/tasks need to be cooperatively performed. The coordination/scheduling layer reasons about possible solutions to achieve the goal and sequences local activities. In this reasoning process, the criteria requirements such as the balance between achieving a good result quickly versus achieving a high quality result in a longer time, resource requirement and interaction with other agents are all considered. The communication and reasoning process in the coordination/scheduling layer are transparent to the domain problem solver. The GPGP coordination module communicates with other potential participant agents and builds proposed commitments for the common goal. The DTC scheduler reasons about local activities and these proposed commitments and verifies the feasibility of these commitments. If the proposed commitments are not suited for the current objective, the GPGP module refines the commitments after negotiating with other agents. The GPGP module may also receive requests from other agents to establish commitments to achieve a particular result.The DTC scheduler also reasons about these requests given the current scheduled activities and verifies if these commitments are feasible. The scheduled activities and established commitments are returned to the domain problem solver for execution. In short, the idea is that the domain problem solver decides what to do, the coordination/scheduling layer decides how to do it and when to do it.

3 Coordination in Little-JIL

3.1 Our Work With Little-JIL

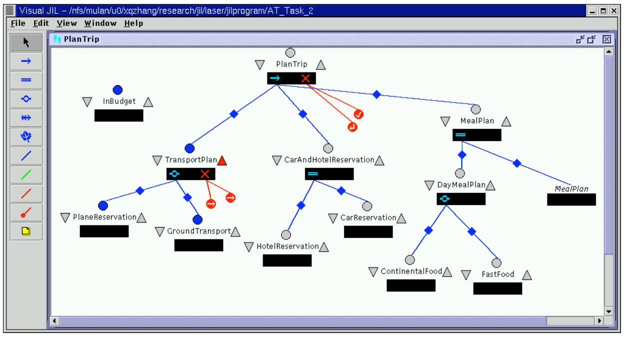

Figure 2: the personal assistant agent's task in Little-JIL

Little-JIL [4] is a process-programming language that is used to describe software development process and other processes. Little-JIL represents processes as compositions of steps, which may be divided into substeps. The specification of a step is defined in terms of a number of elements. Each element defines a specific aspect of step semantics, such as data, control, resource usage, or consistency requirements. Little-JIL language provides a description of a multiagent process. It describes control flows, data flows, resource requirements of a process. However, if there are alternative ways of accomplishing subtasks, the agent needs to reason about the effects of the choice on the overall process's characteristics. This problem is more difficult when this process is distributed among multiple agents, agents need to coordinate with each other to find a solution that meets the global criteria function. Furthermore, there are interactions among steps that agents need to coordinate over, which poses problems for multi-agent coordination.

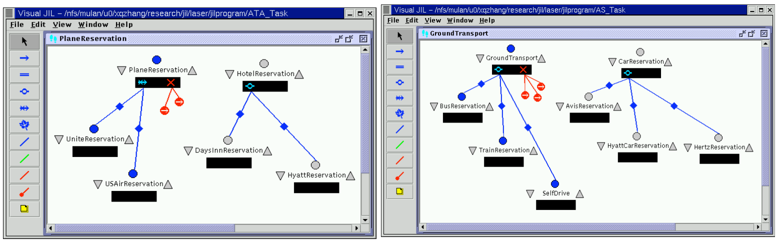

Figure 3: the secretary agent's task and the travel agent's task in Little-JIL

Our solution to these problems is to integrate our coordination module with the Little-JIL process problem solver. Figure 4 describes the infrastructure of an agent that works on a Little-JIL process program. The Little-JIL process program is generated by the Little-JIL editor(outside of the agent) or is received from another agent(task assignment). The Little-JIL problem solver receives and executes this process program.This Little-JIL process program is sent to the Little-JIL unwinder, which opens this process

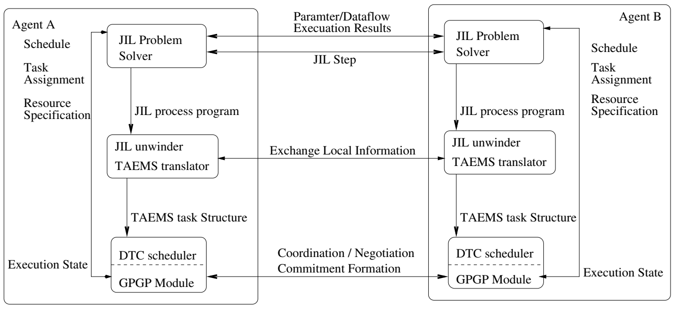

Figure 4: Little-JIL Agent Architecture

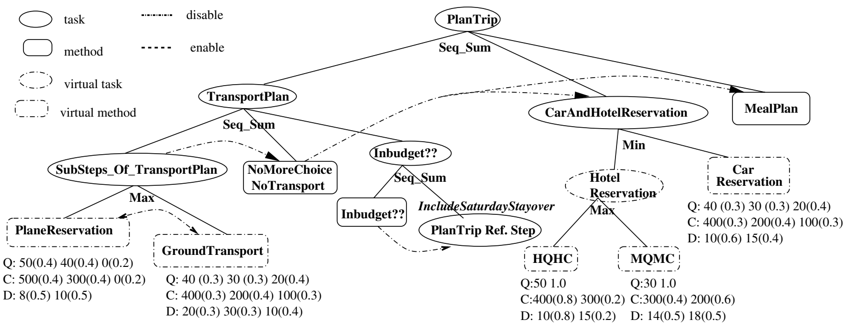

Figure 5: the personal assistant agent's task structure

program, discovers the interactions among agents, and extracts the resource requirement information. In the unwinding process, some steps are detected as non-local tasks (for example, the *PlaneReservation* step should be executed by a TravelerAgent), these non-local tasks are treated as virtual tasks: they will be analyzed but not executed locally. The TÆMS translator takes this process program and the information provided by the Little-JIL unwinder and generates a TÆMS task structure. The Design-To-Criteria scheduler uses this TÆMS structure to generate an end-to-end schedule to meet the global criteria requirement. Based on this first round of scheduling, the agent negotiates with other agents to find appropriate commitments for the non-local tasks through the GPGP module. Because the other agent may not be able to perform the task as specified by the local scheduler, re-scheduling or re-assignment may be needed to achieve a satisfactory commitment. In this negotiation process, the agent also should take the resource requirements into consideration, making sure resources are available when they are needed.The "final" schedule with task assignments and resource specifications are returned to the Little-JIL problem solver and the process is executed as scheduled.

3.2 Example

For example , there are three agents which work together to plan a trip. They are the personal assistant agent, the travel agent and the secretary agent. To plan a trip(*PlanTrip*) the personal assistant agent needs to do three things in sequence: first plan the transportation (*TransportPlan*), then reserve a car and a hotel (*CarAndHotelReservation*), then plan the meal(*MealPlan*). To plan the transportation, there are two choices, either ask a travel agent to make the plane reservation (*PlaneReservation*) or ask the secretary agent to plan the ground transport. To make the plane reservation(Figure 3, right), the travel agent will try United Airlines(*UnitedReserveration*) first, if it fails, then try USAir (*USAirReserveration*). To plan the ground transportation(Figure 3, left), the secretary agent has three choices, either take a bus (*BusReservation*) or take the train(*TrainReservation*), or drive a car (*SelfDrive*). Similarly, there are two choices to reserve the hotel - *DaysInnReservation* or *HyattReservation*; there are three ways to reserve car - *AvisReservation*, *HertzReservation*, and *HyattCarReservation*. The personal assistant agent has the high level global view of the *PlanTrip* task(Figure 2), but it has no detailed information of each process. Both the travel agent and the secretary agent have detailed process information about their local tasks, but they don't have a global view of the *PlanTrip* task and they don't know the context of their local tasks.The context is constructed when the Little-JIL unwinder opens the process program, exchanges local information and discovers those interactions among agents.

One kind of communication is caused by the supertask/subtask relationship. For example, the personal assistant agent recognizes it is the travel agent who really performs the *PlaneReservation* task, which is a non-local task for itself. In this case, the personal assistant agent would like to use some quantitative information on the performance of non-local tasks from the travel agent and the secretary agent. This information may describe possible different approaches to do the non-local task, each approach has different performance characteristics, as the *HotelReservation* task in Figure 5, or the information is only the estimation of the quality, cost, duration. This information is used to help the personal assistant agent construct a reasonable plan that best matches its criteria requirement. The NLE(Non Local Effects) relationship is discovered though the information exchanging. For example, the Hyatt car can be reserved only if the Hyatt hotel has been reserved. This restriction is represented as an *enables* edge from the *HyattReservation* task to the *HyattCarReservation* task in TÆMS task structure.

The TÆMS translator takes the process program and those interaction relationships discovered by the unwinder and translates the Little-JIL process program to the TÆMStask structure. In the translation process more NLE relationships are discovered and recorded. They come from the context of Little-JIL steps. For example, because the *TransportPlan* step is a *choice* step, this means only one step of the *PlaneReservation* and the *GroundTransportation* steps needs to be successfully completed. This relationship is represented as a disables edge from the *PlaneReservation* task to the *GroundTransportation* task and vice versa. In the translation process, the resource requirements also are translated and recorded in the TÆMS language. For instance, the *DaysInnReservation* step requires a computer because the reservation needs to be done on the Internet.

After the unwinding and the translating process, every agent has a TÆMS task structure that includes related NLE relationships and necessary resource specifications(Figure 5, 6). It should be noted that the agent may also have other local tasks besides those tasks in this *PlanTrip* task. The environment we are studying is a multiagent, multitask environment. The GPGP module and the DTC scheduler will work on the agent's local task group to find out a reasonable and efficient local scheduler, with negotiation and coordination with other agents. We will use the following examples to explain how this module performs the functions described in section .

Reasoning about the feasibility of activities The personal assistant schedules its local task based on its criteria function (i.e.High Quality and High cost: a trip with higher quality accommodation while is more costly is acceptable) and on the performance characteristics of each step; th following schedule is recommended by the DTC scheduler(Figure 7) From this schedule, the

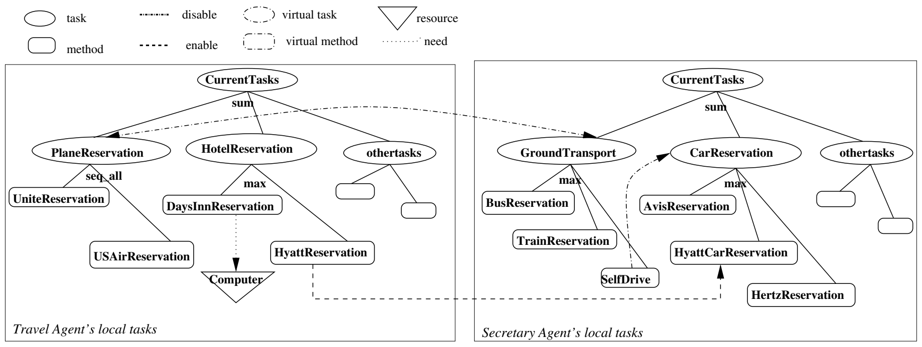

Figure 6: the travel agent and the secretary agent's task structure

travel agent and the secretary agent will obtain three kinds of information that guide their local activities. One is a time constraint on its tasks, specifying when is the earliest time they can start and when is the latest time they should be completed. For example, the travel agent will try to schedule its local activities so as to complete the *PlaneReservation* task by time 19. The second is the context information, i.e. the secretary agent can only execute the *GroundTransport* task if the *PlaneReservation* task fails. The third is the criteria related information, e.g. after *HotelReservation(HQHC)*(a high-quality, high cost approach) task is chosen by the person assistant, the secretary agent is likely to choose the *HyattReservation*(it has higher quality/cost) task instead of the *DaysInnReservation* task.

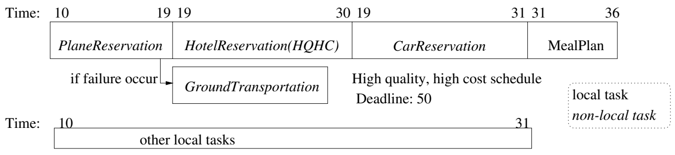

Figure 7: the high quality, high cost schedule of the *PlanTrip*

Choosing from and sequencing possible activities As discussed above, the high level schedule (HotelReservation(HQHC)) helps guide agents(the secretary agent) make local choices (*HyattReservation*) regarding how to meet the global criteria function. Also because an agent may have multiple local tasks, it is also important for the agent to sequence its local activities. This is a negotiation process. For example, the personal assistant agent wants the *PlaneReservation* to be done by time 19, the travel agent finds it is impossible to meet this deadline and instead offers time 35 as a completion time. The personal assistant thinks it is too late, then asks for an earlier time. The travel agent offers time 25 on the condition that it would do a narrower search exploring only a few airline companies. The personal assistant thinks it is OK because meeting the deadline is critical and the budget is not tight. It will reschedule the *PlanTrip* task based on the commitment that the *PlaneReservation* task is done by time 25. This negotiation process is done by the commitment formation/negotiation module in GPGP.

Assisting the task allocation After scheduling local activies, the agent may need to relocate some tasks in order to obtain higher global utility. For example, after the travel agent chooses the *HyattReseveration*, the secretary agent finds it is better to ask the travel agent to do the *HyattCarReservation* task also because there is an overlap between these two tasks. So the negotiation goes on between this two agents about this task allocation process. This is done by the task allocation module in GPGP.

Assisting the resource allocation When the agent sequences its local activities, the resource requirement should also be taken into account. For instance, the travel agent needs a computer to perform the *DaysInnReservation* task, and the computer is shared with other agents. The travel agent needs to negotiate with other agents or the manager of the computer to make sure the computer is available when it plans to do the *DaysInnReservation* task. If not, it needs to reschedule its local activities to allow the *DaysInnReservation* task to be performed when computer is available. This resource acquisition process is done by the resource acquisition module in GPGP.

4 Conclusion and Future Work

We view coordination as a multi-level process in which the higher level uses domain state to model coordination while the lower level uses quantitative information about the performance constraints to make decisions. In this paper, we have used the Little-JIL framework as an example of a high level coordination framework and GPGP/DTC as the lower level coordination framework. We have shown how these two frameworks can be integrated in order to develop a sophisticated approach to agent coordination.In the future, we are going to do some experiments to evaluate this intergration work, we would like to compare how the agents work differently with and without the coordination/scheduling module.

References

[1] Decker, K., Lesser, V. R. Quantative Modeling of Complex Environments. In International Journal of Intelligent Systems in Accounting, Finance and Management. Special Issue on Mathematical and Computational Models and Characteristics of Agent Behaviour., Volume 2, pp. 215-234, 1993.

[2] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proc. of the 1st Intl. Conf. on Multi-Agent Systems*, pages 73–80, June 1995. AAAI Press.

[3] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *Intl. Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998.

[4] Alexander Wise, Barbara Staudt Lerner, Eric K. McCall, Leon J. Osterweil, and Stanley M. Sutton Jr. Specifying Coordination in Processes Using Little-JIL. November 1999. Submitted to the International Conference on Software Engineering 2000.