Exploring Organizational Designs with TÆMS: A Case Study of Distributed Data Processing*

M.V. Nagendra Prasad[1], Keith Decker[2], Alan Garvey[3] and Victor Lesser[1],

[1]Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{nagendra,lesser}@cs.umass.edu

[2]Department of CIS
University of Delaware
103 Smith Hall
Newark, Delaware 19716
decker@cis.udel.edu

[3]Department of Computer Science
Pacific Lutheran University
Tacoma, WA 98447

Abstract

When we design multi-agent systems for realistic, worth-oriented environments, solving the coordination problems they present involves understanding the intricate and sophisticated interplay between the domain and the various system components. In this paper, we examine this interplay using TÆMS and the GPGP (Generalized Partial Global Planning) approach as a tool for quickly exploring multiple alternative organizations/coordination strategies. We illustrate the flexibility of TÆMS-style task structures for representing interesting multi-criteria coordination problems, and the use of a new grammar-based generation tool to allow quicker task structure experimentation. Using Distributed Data Processing as an example domain, we show how different coordination algorithms of varying sophistication give rise to subtle effects in functionally-structured agent organizations with interrelationships between functions. The experiments presented in our paper serve to illustrate our claim about the generality and flexibility of the TÆMS representation for coordination problems and the grammar-based generators for modeling domain semantics and experimenting with different aspects of such domains.

Introduction

It has often been observed in human organizations that there is a strong effect of environmental factors such as dynamism and task uncertainty on the necessary coordination actions taken by organization participants to produce organizationally acceptable outcomes (Lawrence & Lorsch 1967; Galbraith 1977; Stinchcombe 1990). We and others have often pointed out that this applies to purely computational organizations as well (Fox 1981; Decker & Lesser 1995). Therefore it is important to develop tools and methodologies for:

- representing interesting, complex coordination problems. By this we mean problems with deadlines, positive and negative interactions, and outcome uncertainties. These problems are situated in worth-oriented domains (Rosenschein & Zlotkin 1994) with multiple decision criteria.
- building such representations fairly quickly. Otherwise empirical studies become drawn-out exercises in extensive, knowledge-intensive modeling.
- instantiating environment-specific coordination mechanisms.
- meaningfully comparing system performance under these different mechanisms.

This paper will describe just such an approach, starting with a brief review of our standard TÆMS task structure representation for complex multi-agent coordination problems and a description of a new graph-grammar-based stochastic task structure description language and generation tool. Stochastic graph grammars are used to capture the morphological regularities in a domain. Grammars are augmented with attributes to capture additional aspects of domain semantics. Using such grammar-based generators, we model functionally-structured agent organizations with interrelationships between functions in the domain of distributed data processing. We simulate three approaches to coordination of different levels of sophistication: no commitments, tacit a priori commitments and dynamically generated commitments. The results of conducting a number of experiments led to insights about the required sophistication of coordination of such agent organizations operating in resource limited and time constrained environments. More importantly, these experiments serve to bolster our claim about the generality and flexibility of the TÆMS representation for coordination problems and the grammar-based generators for modeling domain semantics and experimenting with different aspects of such domains.

*This material is based upon work supported by the National Science Foundation under Grant Nos. IRI-9523419 and the Department of Navy, Office of the Chief of Naval Research under Grant No. N00014-95-1-1198. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

TÆMS: Task Analysis, Environment Modeling, and Simulation

The TÆMS framework (Task Analysis, Environment Modeling, and Simulation) (Decker & Lesser 1993; Decker 1995) represents coordination problems in a formal, domain-independent way. In the simplest terms, a TÆMS model of a task environment specifies what actions are available to agents and how those actions relate to one another and to the performance of the system as a whole. We have used it to represent coordination problems in distributed sensor networks, hospital patient scheduling, airport resource management, distributed information retrieval, pilot's associate, local area network diagnosis, etc. (Decker 1995). TÆMS models problem-solving activities of intelligent agents operating in complex environments where:

- Responses to some tasks are required by specific deadlines.
- The problem-solving environment involves a *worth-oriented domain* (Rosenschein & Zlotkin 1994). In a worth-oriented domain goals are encoded as functions that rate the acceptability of states, and the agents strive for the best solution possible (but not necessarily an optimal solution). We use the term *quality* to refer to all acceptability characteristics other than temporal characteristics.
- Optimal performance on individual computational tasks may not be possible due to missing or incomplete information, or lack of processing time.
- Agents may concurrently contribute to the achievement of multiple goals involving potentially different subsets of agents. The computational results from multiple agents may need to be integrated to form a complete solution.
- Subproblems are interdependent, that is, cannot be solved independently in isolation.

One implication of the deadline requirement is that the representation must specify when there are multiple ways to accomplish a goal that trade off the time to produce a result for the quality (acceptability) of the result. The possibility of incomplete or missing information will, by necessity, lead to agents working in a satisficing mode in which problem solving is structured to operate effectively even with missing information. For this reason, the representation includes what we will call "soft" coordination relationships that define the implications, in terms of both result quality and computation duration, of specific information arriving prior to the start of the computational task. Additionally, the effect of an agent's activities may not be quantifiable from a local perspective; instead it may need to be measured from the perspective of how it contributes to the solution of a high-level goal. Thus, there needs to be a representation of how progress towards the achievement of a goal occurs incrementally as the result of multiple activities. Finally, the representation of agent activity should allow for the possibility that individual agent's activities contribute to different and independent high-level goals.

TÆMS models

We will present a brief summary of the objective (external environment) definition of TÆMS models. Interested readers will find many more details and examples in (Decker 1995). The following description is top-down, starting with the environment and ending with what the agent perceives. Thus we do not define a coordination problem until the final subsection. TÆMS models are discrete state-based models, where the state of all agents in the system at time $t+1$ is computed from their state at all previous times.

An *environment* Ψ is a generator function that takes arbitrary set of parameters (including the set of agents), and produces *episodes* $\mathbf{E}$. This is done according to the generative model for the specific environment; this paper will discuss a new grammar-based generator. An episode is a particular identifiable collection of problem instances—later in this paper we will be looking at an episode as a collection of problems that occur at a distributed data processing center over the course of one full day.

We will now describe the objective task structure of a problem-solving episode. An *episode* $\mathbf{E}$ consists of a set of *task groups* $\mathcal{T}$; $\mathbf{E} = \langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n \rangle$. Each task group has an arrival time $\mathrm{Ar}(\mathcal{T})$, and a deadline $\mathrm{D}(\mathcal{T})$. A task group represents a set of computationally related actions. A *task group* is represented by a directed acyclic graph. The nodes of the graph are called *tasks* T. One task is denoted the root task, and is usually simply indicated by the symbol for the entire task group, $\mathcal{T}$. Tasks that have no children are called *executable methods*, or just *methods* M for short. Tasks that do have children, but that are not the root task, are straightforwardly called *subtasks*. The structure of a task group is meant to reflect the problem's task structure.

The edges of this graph form the subtask relationship. Task or task group quality at a given time ($Q(T,t)$) is based on the *subtask* relationship. This quality function is constructed recursively. Formally, the subtask relationship is defined as $\mathsf{subtask}(T, \mathbf{T}, Q)$, where $\mathbf{T}$ is the set of all direct subtasks of T and Q is a quality function $Q(T,t)$: [tasks $\times$ times] $\mapsto$ [quality] that returns the quality associated with T at time t. The quality accrual semantics of a particular environment are modeled by the appropriate choice of the quality function Q (e.g., minimum, maximum, summation, or the arithmetic mean). In particular, we will often write of the quality achieved at a task group at time t, $Q(\mathcal{T}, t)$, meaning the current quality at the root task.

Executable methods represent domain actions, like executing a blackboard knowledge source, running an instantiated plan, or executing a piece of code with its data. Executable methods have several functions defined on them. $\mathbf{q}(M,t)$ is the *current maximum quality* that can be achieved by executing M at time t for its duration $\mathbf{d}(M,t)$. $\mathbf{d}(M,t)$ is the *current duration* of method M at time t. Progress(M,t) is the number of time units spent executing M. The definitions of $\mathbf{q}(M,t)$, $\mathbf{d}(M,t)$, and Progress(M,t) are fixed by TÆMS. $Q(M,t)$ is the quality at M at time t. This function is available for modeling a particular environment, but it is constrained to obey the identity:

$$Q(M,t) = \mathbf{q}(M,t)$$

$$\text{if } [\text{Progress}(M,t) = \mathbf{d}(M,t)] \wedge [\text{Finish}(M) \leq \text{D}(M)]$$

(i.e. the quality is the maximum quality if the executable method was completed before its deadline).

Any task T containing a method that starts executing before the execution of another method M finishes may potentially affect M's execution through a *non-local effect* e. We write this relation (a labeled arc in the task structure graph) as $\mathsf{nle}(T, M, e, p_1, p_2, \ldots)$, where the p's are parameters specific to a class of effects. There are precisely two possible outcomes of the application of a non-local effect on M under our model: *duration effects* where $\mathbf{d}(M,t)$ (duration) is changed and *quality effects* where $\mathbf{q}(M,t)$ (maximum quality) is changed. An effect class e is thus a function $e(T, M, t, d, q, p_1, p_2, \ldots)$: [task $\times$ method $\times$ time $\times$ duration $\times$ quality $\times$ parameter 1 $\times$ parameter 2 $\times \ldots$] $\mapsto$ [duration $\times$ quality].

Each method has an initial maximum quality $\mathbf{q}_0(M)$ and duration $\mathbf{d}_0(M)$ so we define $\mathbf{q}(M,0) = \mathbf{q}_0(M)$ and $\mathbf{d}(M,0) = \mathbf{d}_0(M)$. If there is only one non-local effect with M as a consequent $\mathsf{nle}(T, M, e, p_1, p_2, \ldots)$, then $[\mathbf{d}(M,t), \mathbf{q}(M,t)] \leftarrow e(T, M, t, \mathbf{d}(M, t-1), \mathbf{q}(M, t-1), p_1, p_2, \ldots)$. If there is more than one non-local effect, then the effects are applied one after the other in an order specified in the model. We have defined at least sixteen example non-local effects (Decker 1995).

We can define a performance measure $\mathcal{P}(\mathbf{E})$ for the system (or for an agent) that is a function of the episode. The default is the sum of the task group qualities ($\mathcal{P}(\mathbf{E}) = \sum_{\mathcal{T} \in \mathbf{E}} Q(\mathcal{T}, \text{D}(\mathcal{T}))$). We can also define a control function (alternately, a "strategy", decision rule, or control policy) for each agent that given the agent's current beliefs and the time will return the action that the agent should take at that time. One statement of a cooperative coordinated control problem (similar in spirit to the specification of a team theory decision problem) is then to find a set of such control functions, one for each agent in an environment, so as to attempt to maximize the expected performance measure for the entire system of agents.

Grammar-based Task Structure Generation

Decker and Lesser(Decker 1995; Decker & Lesser 1995) illustrate the importance of extensive empirical studies in determining the role of different coordination algorithms in different task environments. However, these studies relied on a weak task environment generator where the experimenter was limited to setting certain numerical parameters like mean of the task structure depth or mean and variance of the number of interrelationships in task structures. This often gives rise to a wide range of task structures and a huge variance in the types of capabilities needed by the system to effectively handle them. More importantly, it is unlikely that most real applications involve an infinite variety of task structures. The domain semantics dictate and limit morphology of the task structures. While there is bound to be some randomness in these structures, it is highly unlikely that the only regularity that can be modeled in the task structure representations of a coordination problem instance are a few parameters like its mean depth or branching factor. Below we introduce a graph grammar based task structure specification language that is powerful enough to model the topological relationships occurring in task structures representing many real life applications. We first briefly introduce graph grammars (GGs) and our extensions to traditional graph grammars to capture domain semantics beyond the topological relationships. We then show how a data-flow model of data processing can be captured by task structures and their grammatical specifications.

Graph Grammars Graph grammars are a powerful tool used in a number of domains(Mullins & Rinderle 1991; Nagl 1979) to capture and characterize the underlying structural regularities. They offer a structured way of describing topological relationships between entities in a domain. Graph grammars are fundamentally similar to string grammars(Chomsky 1966; Nagl 1979) with the difference lying in the productions. A *graph production* is a triple p = $(g_l, g_r, \mathcal{E})$ where g_l is the subgraph to be replaced (left hand side) and g_r is the subgraph to be inserted in its place in the host graph. E is the *embedding transformation*. A number of schemes for graph grammar have been proposed and the primary differences between them arise from the differences in the embedding schemes. Much of the traditional literature in graph grammars does not deal with attribute valued nodes and edges and stochastic productions. We need attributes to capture a number of other aspects of domain semantics in addition to the topological relationships between entities. Stochasticity in the productions adds more power to the modeling potential of these grammars by capturing aspects of uncertainty in the domain. Accordingly, we call our grammars Attribute Stochastic Graph Grammars (ASGGs).

Let a graph $G = (V, E)$, where V is the set of vertices (also referred to as nodes) and E is the set of edges. Nodes and edges can have labels. An Attribute Stochastic Graph Grammar is defined as a 8-tuple $\langle \Sigma_n, A_n, \Sigma_t, A_t, \Delta, A_\Delta, S, P \rangle$ where the nonterminal node alphabet (Σ_n), the terminal node alphabet (Σ_t), and the edge-label alphabet(Δ) are finite, non-empty, mutually disjoint sets, A_n, A_t, and A_Δ are the respective sets of attributes, $S \in \Sigma_n$ is the start label (can be a node or a graph) and P is a finite nonempty set of graph production rules(Sanfeliu & Fu 1983). A graph production is a 4-tuple $p_i = \langle g_l^i, g_r^i, \mathcal{E}^i, Pr(p_i) \rangle$ where $\sum Pr(p_j) = 1 \mid (p_j \in P \textit{ and } g_l^j \textit{ are isomorphic})$

Let G' be a graph derived from S using P. Rewriting this graph involves what is called a LEARRE method(Mullins & Rinderle 1991): <u>L</u>ocate a subgraph g' that is isomorphic to the lefthand side, g_l^i of production $p_i \in P$, establish the <u>E</u>mbedding <u>A</u>rea in G', <u>R</u>emove g' along with all the edges incident on it, <u>R</u>eplace g' with g_r^i and <u>E</u>mbed it to the host graph $G' - g'$. Figure 1 shows a simple example.

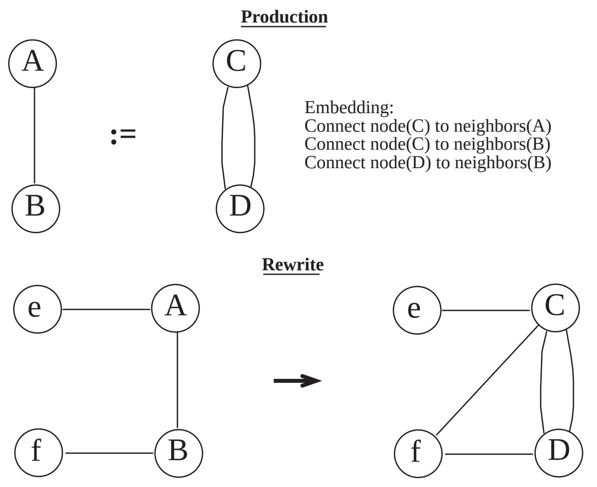
Production
A
B
:=
C
D
Embedding:
Connect node(C) to neighbors(A)
Connect node(C) to neighbors(B)
Connect node(D) to neighbors(B)
Rewrite
e
A
f
B
e
C
f
D

Figure 1: Example of a Graph Grammar Rewriting

The rewriting stops when every node label is in the terminal alphabet set Σ_t and the resultant graph is considered a member of the family of graphs generated by the ASGG. For the sake of expository clarity the example doesn't deal with attributes. Distributed data processing examples in the next section will show instances of attributes.

Another extension to our definition of graph grammars involves partial edges. A production embedding can specify a link between a local label and non-local label (in the host graph) that may be non-existent at the time of the application of the production. This "hanging" edge gets linked if and when a node with the corresponding label arises at a later time due to rewriting in the host graph. If the termination graph is reached without a node with the corresponding label, this hanging edge is removed at that point.

ASGGs and Distributed Data Processing Our example of a functional agent organization is derived from the domain of distributed data processing (DDP) that involves a number of geographically dispersed data processing centers (agents). Each center is responsible for conducting certain types of analysis tasks on streams of satellite data arriving at its site: "routine analysis" that needs to be performed on data coming in at regular intervals during the day, high priority "crisis analysis" that arises due to the stochastic occurrence of certain events and "low priority analysis" that arises at the beginning of the day with a probability. Low priority analysis involves performing specialized analysis on specific archival data. Different centers need to perform different kinds of tasks for a given type of analysis. Different types of analysis tasks have different priorities. A center should first attend to the "crisis analysis tasks" and then perform "routine tasks" on the data. Time permitting, it can handle the low-priority tasks. The processing centers have limited resources to conduct their analysis on the incoming data and they have to do this within certain deadlines. Results of processing data at a center may need to be communicated to other centers due the interrelationships between the tasks at these centers. For example, in order begin analyzing a particular stream of satellite data, a center may need the results of a related analysis from another center. The question that we considered was whether appropriate coordination among centers could be provided by (1) a fairly simple data-flow driven coordination algorithm, by (2) precomputed standard operating plans, or whether (3) a more complex but time-consuming algorithm using non-local commitments was necessary.

We used the GPGP approach(Decker & Lesser 1995; Decker 1995) to model these three different coordination algorithms of varying sophistication. We first discuss how a data processing problem can be represented as a TÆMS task structure and how ASGGs can be used to represent domain semantics of a data processing problem. In the following discussion data processing center and agent are used interchangeably.

Figure 2a shows the data flow representation of an example DP problem. It consists of three tasks: T_1, T_2, and T_3. T_1 facilitates T_2, and T_2 enables T_3.

"Facilitates" is a soft interrelationship which says that if the results of processing task T_1 are available at T_2 when it starts processing, then the duration of the method for achieving T_2 is reduced and its quality is increased. "Enables" is a hard interrelationship that implies that the results of processing T_2 must be available at T_3 before it can start processing.

A TÆMS representation of the same problem can be seen in Figure 2b. The leaves of the task structure show available methods for the tasks. Figure 3 shows a graph grammar for this task structure. To avoid cluttering, not all attributes are shown in the figure. There are a number of attributes like the quality accrual function, termination time, earliest start time for tasks, and type, duration, quality, deadline, termination time, etc. for methods. The quality and duration attributes of a method are specified as distributions representing certain aspects of uncertainty in the underlying domains. This feature of the grammar leads to a large variety of semantically different task structures but their structural variety is limited by the graph productions.

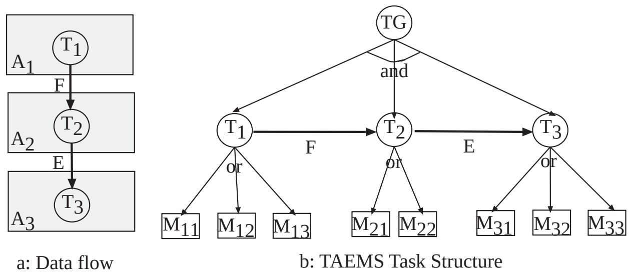

Figure 2: Example of a Data Processing problem and its TÆMS representation

We modeled the DDP domain using three graph grammars, each one generating task structures representing the type of operations that need to be performed on the incoming data. A grammar is activated at regular intervals (representing arrival times for data) with a probability corresponding to its type. For example, the low priority task structure grammar is activated with probability "low_priority_tasks_probability_parameter" only if the time is 0 (beginning of the day). Achieving all crisis tasks gets higher final quality than routine tasks and achieving routine tasks gets higher quality than low priority tasks. The task deadlines are basically determined by the policies at a center and they represent the amount of processing time available to a center to process the data it sees. Figure 4 shows the interrelationships of the three types of processing tasks we investigated in the following experiments. The tasks are distributed across agents as shown by the rectangles demarcating the

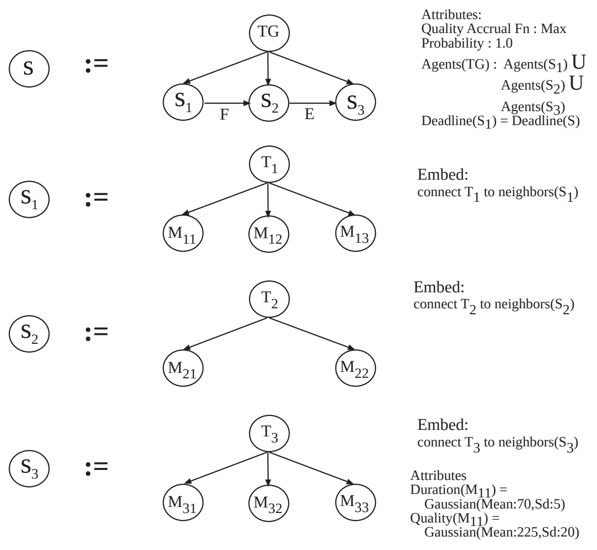
S
:=
TG
S1
F
S2
E
S3
Attributes:
Quality Accrual Fn : Max
Probability : 1.0
Agents(TG) : Agents(S1) U Agents(S2) U Agents(S3)
Deadline(S1) = Deadline(S)
S1
:=
T1
M11
M12
M13
Embed:
connect T1 to neighbors(S1)
S2
:=
T2
M21
M22
Embed:
connect T2 to neighbors(S2)
S3
:=
T3
M31
M32
M33
Embed:
connect T3 to neighbors(S3)
Attributes
Duration(M11) = Gaussian(Mean:70,Sd:5)
Quality(M11) = Gaussian(Mean:225,Sd:20)

Figure 3: Example of a Graph Grammar Rewriting

agents.

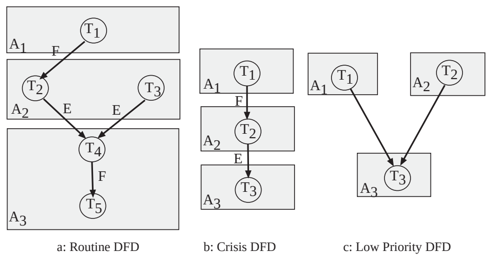

Figure 4: Data Processing Problems

Instantiating Environment-specific Coordination Mechanisms

In order to bring to bear different collections of coordination mechanisms for different multi-agent problem-solving environments, we use the Generalized Partial Global Planning (GPGP) approach(Decker & Lesser 1995; Decker 1995). The GPGP approach consists of an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. This approach has several unique features:

- Each mechanism is defined as a response to certain features in the current subjective task environment. Each mechanism can be removed entirely, or parameterized so that it is only active for some portion of an episode. New mechanisms can be easily defined.
- The GPGP coordination component works by supplying information and constraints to the

local scheduler, and to other agent's coordination components. The most common constraint is a *commitment*(Cohen & Levesque 1990; Jennings 1993) to achieve a certain level of quality at a task by a certain time. The empirical results in this paper were achieved using a 'design-to-time' soft real-time local scheduler developed by Garvey (Garvey & Lesser 1995).

- GPGP is characterized by its use of commitments as a powerful and flexible way to achieve domain independent coordination among agents.

The coordination module in GPGP consists of several coordination mechanisms, each of which notices certain features in the task structures locally known, and responds by taking certain communication or information gathering actions, or by proposing new commitments. The coordination module keeps track of local commitments and commitments received from other agents, and chooses from among multiple schedules if the local scheduler returns multiple schedules.

The important point is that we can use the flexibility of the GPGP approach to easily introduce new coordination mechanisms for different environments. By using the TÆMS-based simulator and GPGP together, we can derive a better empirical understanding of what the effect of different coordination mechanisms is on important performance characteristics in a problem solving environment. This gives us a powerful tool for posing and answering questions about coordination and organization of multi-agent systems in complex environments. In the next section, we will use these tools to explore the Distributed Data Processing environment. Specifically we will be investigating the effect of three coordination algorithms in this domain:

1. Data Flow Algorithm: An agent communicates the result of performing a task to all the agents and the other agents can exploit these results if they still can. There are no commitments from any agent to any other agent. In the case of facilitates interrelationships, the results need to arrive before the start of the facilitated task. If a recipient agent has a task that is enabled by the result, it can start executing it only after receiving that result. In the GPGP approach, these characteristics can be obtained by turning off all coordination relationship exploitation mechanisms, turning on non-local view detection and communication of all results.

2. Modular(or dynamic-scheduling) represents the other extreme where agents coordinate through commitments and relevant results that are committed to other agents are communicated. Agents schedule dynamically trying to exploit as best as possible the interdependencies among tasks. The agents have the relevant non-local view of the coordination problem, detect coordination relationships, form commitments and communicate the committed results.

3. Rough coordination is similar to modular but commitments do not arise out of communication between agents. Each agent has an approximate idea of when the other agents complete their tasks and communicate results based on its past experience. "Rough commitments" are a form of tacit social contract between agents about the completion times of their tasks. However, it is unrealistic to expect the commitments on low probability crisis tasks and low priority tasks to follow such tacit a priori rough commitments. So this coordination type uses rough commitments for routine tasks but behaves just like data flow for the non-routine crisis and low priority tasks. The agents have the relevant non-local view of the coordination problem, detect coordination relationships, but use rough commitments for routine tasks and communicate the committed results. It might be possible to view rough commitments as precompiled social laws (Shoham & Tennenholtz 1992).

Experiments

We performed a series of experiments varying a number of parameters in our model to see the effectiveness of different coordination mechanisms used to manage the activities at different centers. The experiments reported below study the performance of the three coordination algorithms discussed previously.

Effect of Crisis Tasks

The probability of the centers seeing crisis tasks was varied from 0 to 1.0 as shown in Table 1. Each cell in the table shows quality averaged over 100 runs at the particular parameter settings. Low priority tasks arrived with a probability of 0.5 and the routine tasks were always seen at the time of new arrivals. The deadline was fixed at 140 time units for these experiments. At lower values of crisis tasks probability (0.0 and 0.1) , modular performed better than both rough and data flow (significantly better using the Wilcoxon matched-pair signed-ranks test at significance level 0.05). However, as the probability increases rough and data flow algorithms perform significantly better at crisis tasks probability 1.0. This initially looked counter intuitive to us but a closer examination of the runs revealed the reasons. The commitment mechanisms implemented as of now in GPGP are "one-way". There are commitments from the predecessor end of the interrelations to the agents at the successor end but not the other way. For example, agent A_1 commits to doing T_1 of the crisis task structure (see Figure 4b) by a certain time and agent A_2 tries to take this commitment into consideration while scheduling for its crisis tasks. However, there are no commitments from agent A_3 about the latest time it needs the results of execution of task T_2 so that it can schedule its methods. This results in Agent A_2 trying to take advantage of commitments from A_1 and in the process sometimes delaying execution of its methods until it is too late for agent A_3 to execute its methods enabled by the results from A_2. These experiments lead us to an important general observation that just being sophisticated may not guarantee better performance of a coordination algorithm. It has to be sufficiently and appropriately sophisticated—in this case, commitments need to flow in both directions between predecessors and successors. Otherwise, the coordination algorithm may not only be wasteful in computation and communication resources but it may also lead to inappropriate behavior.

Crisis TG Probability	Data Flow	Modular	Rough
0.0	0	147.6	91.5
0.1	64.2	184.5	149.9
0.5	243.1	282.5	295.2
1.0	437.4	406.1	439.77

Table 1: Average Quality for deadline 140

Effect of Deadlines

We ran another set of experiments with deadline being varied. The crisis tasks probability was set to 0.1 and low priority tasks probability was set to 0.5. New routine tasks were always seen at the time of new data arrival. The results of these experiments are shown in Table 2 where each cell in the table shows quality averaged over 100 runs at the particular parameter settings. Modular performed significantly better than the other two coordination mechanisms at higher deadlines (140, 200, 300, 500). However, much to our initial surprise, increasing the deadline from 140 time units to 200 led to a decrease in average quality for both modular and rough. This arises due certain subtle interactions between the coordination mechanism and the design-to-time scheduler. When the deadline is 140 time units, agent A_2 chooses lower quality methods that have lower durations for performing its tasks. However, when the deadline increases to 200 time units, it chooses higher quality methods for its schedule of activities. However, since the agents are coordinated using only predecessor commitments, A_2 now delays its higher duration methods long enough to adversely effect the ability of A_3 to execute its enabled methods by not providing results in time. As deadline time increases above 200, it can be seen that the average quality increases again (more time is available for enabled methods in A_3 to complete execution).

Deadline	Data Flow	Modular	Rough
100	0	0	0
120	58.5	24.8	27.2
140	64.2	184.5	149.9
200	73.6	124.0	92.6
220	80.3	297.5	214.2
300	104.8	342.3	120.0
500	108.1	378.0	126.5

Table 2: Average Quality for for different deadlines

Discussion

The actual results of these experiments are not as important as the fact that the TÆMS representation with a graph grammar based modeling framework provides a flexible and powerful way to experiment extensively and tease out subtle and intricate interactions between the components of a complex system, and between the domain and the system. The importance of such studies, especially for designers building complex systems, cannot be overstated. Our own understanding of the GPGP family of coordination algorithms (even after working with it for a number of years) has benefited from such an exercise.

In addition to the experiments discussed above, we have done a number of other experiments as well, but we will not discuss them here because of space. Following are some of the other insights gleaned from these experiments:

- In our experiments, an agent cannot interrupt the execution of a method once it has begun. This leads to some severe performance lapses on the part of the agent because it does not see task structures for a long time after their arrival - until it finishes executing the present method. Based on our observations, we hypothesize that interruptible method execution can lead to better performances in this domain.
- If the agents can somehow anticipate the task arrivals (for example, an expert may forecast the arrival of tasks based on the activity of the past few days and his or her past experience), coordinating for them up front can lead to performance gains for the agents.

We are currently working to confirm these hypotheses.

Conclusions

This paper has demonstrated the use of representational, experimental, and methodological tools such as TÆMS and GPGP in understanding questions of multi-agent organizational design. The use of these tools facilitates easy and powerful exploration of various coordination strategies for complex and strategically useful domains. Important highlights of our approach include the use of

- quantitative, hierarchical task structures as a flexible mechanism for representing coordination problems
- grammar-based generation techniques for capturing a relevant "domain theory" for an environment without extensive knowledge engineering
- the generalized partial global planning approach to constructing environment-centered collections of coordination mechanisms.

Details about the TÆMS representation and GPGP can be found in (Decker & Lesser 1993; 1995; Decker 1995).

In particular, this paper also considered a particular problem, functionally organized Distributed Data Processing (DDP) in some detail. We investigated the behavior of three different coordination algorithms for a set of task structures in DDP. We empirically demonstrated that the lack of coordination mechanisms for exploiting the successor side commitments in GPGP led to certain "unexpected" behaviors for a particular set of data processing tasks. However, the readers should be warned against generalizing beyond the set of task structures investigated. It is entirely conceivable that a different set of task structures, interrelated in a different way, can lead to wildly different results. On the other hand, the tools provided herein enable a designer to quickly and effectively model these new task structures, perform controlled experiments and determine the behavior of various coordination algorithms in such different environments.

In future, we would like to push this work in a number of different directions. Grammar-based generators have been used to study situation-specific learning of coordination algorithms and the initial results have been promising. We would like to further experiment with such learning in different computational environments. The present set of experiments have made us painfully aware of the lack of successor side coordination in GPGP. We are working to include these mechanisms. In addition, the TÆMS framework is being generalized to explicitly represent uncertainty in quality and duration, cost, and multiple outcomes.

References

Chomsky, N. 1966. *Syntactic Structures.* Mouton and Co.

Cohen, P. R., and Levesque, H. J. 1990. Intention is choice with commitment. *Artificial Intelligence* 42(3):213–261.

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 217–224.

Decker, K. S., and Lesser, V. R. 1995. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems.* San Francisco: AAAI Press. Longer version available as UMass CS-TR 94–14.

Decker, K. S. 1995. *Environment Centered Analysis and Design of Coordination Mechanisms.* Ph.D. Dissertation, University of Massachusetts.

Fox, M. S. 1981. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics* 11(1):70–80.

Galbraith, J. 1977. *Organizational Design.* Reading, MA: Addison-Wesley.

Garvey, A., and Lesser, V. 1995. Design-to-time scheduling with uncertainty. CS Technical Report 95–03, University of Massachusetts.

Jennings, N. R. 1993. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review* 8(3):223–250.

Lawrence, P., and Lorsch, J. 1967. *Organization and Environment.* Cambridge, MA: Harvard University Press.

Mullins, S., and Rinderle, J. R. 1991. Grammatical Approaches to Engineering Design, part i. *Research in Engineering Design* 2:121–135.

Nagl, M. 1979. A Tutorial and Bibliographic Survey on Graph Grammars. In Claus, V.; Ehrig, H.; and Rozenberg, G., eds., *Graph Grammars and their Application to Computer Science and Biology, LNCS 73.* Berlin: Springer-Verlag. 70–126.

Rosenschein, J. S., and Zlotkin, G. 1994. Designing conventions for automated negotition. *AI Magazine* 29–46.

Sanfeliu, A., and Fu, K. S. 1983. Tree-graph Grammars for Pattern Recognition. In Ehrig, H.; Nagl, M.; and Rozenberg, G., eds., *Graph Grammars and their Application to Computer Science, LNCS 153.* Berlin: Springer-Verlag. 349–368.

Shoham, Y., and Tennenholtz, M. 1992. On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 276–281.

Stinchcombe, A. L. 1990. *Information and Organizations.* Berkeley, CA: University of California Press.