Chapter 16

TÆMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms

[1]

Keith S. Decker

The design of coordination mechanisms for groups of computational agents, either interacting with one another or with people, depends crucially on the task environment of which they are a part. Such dependencies include the structure of the environment (the particular kinds and patterns of interrelationships that occur between tasks) and the uncertainty in the environment (both in the *a priori* structure of any episode within an environment and in the outcomes of an agent's actions). Designing coordination mechanisms also depends on properties of the agents themselves (arising in turn from the intentional design of their internal architecture)—a large body of work already exists that describes the principled construction of agents that act based on what are variously termed beliefs, desires, intentions, and goals (see Chapters 2 and 5). The central idea presented here is that the design of coordination mechanisms cannot rely on the principled construction of agents alone, but must rely on the structure and other characteristics of the task environment—for example, the presence of uncertainty and concomitant high variance in a structure. Furthermore, this structure can and should be used as the central guide to the design of coordination mechanisms, and thus must be a part of any comprehensive theory of coordination.

This chapter will first present an outline of the TÆMS framework for building

[1]To appear in G. O'Hare and N. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, Wiley Inter-Science, 1994.

models of task environments at multiple levels of abstraction, along with examples of such models. The next two sections will also describe several short examples of our experiences in using the framework for modeling domains, analyzing a simple distributed sensor network, and generalizing the partial global planning algorithm. Finally the chapter will step back to discuss the current state of TÆMS, and its strengths and weaknesses.

16.1 The TÆMS Framework

TÆMS (Task Analysis, Environment Modeling, and Simulation) is a framework with which to model complex computational task environments that is compatible with both formal agent-centered approaches and experimental approaches. The framework allows us to both analyze and quantitatively simulate the behavior of single or multi-agent systems with respect to interesting characteristics of the computational task environments of which they are part. We believe that it provides the correct level of abstraction for meaningfully evaluating centralized, parallel, and distributed control algorithms, negotiation strategies, and organizational designs. No previous characterization formally captures the range of features, processes, and especially interrelationships that occur in computationally intensive task environments. TÆMS exists as both a language for stating general hypotheses or theories and as a system for simulation. The simulator supports the graphical display of generated task structures, agent actions, and statistical data collection in CLOS (the Common Lisp Object System) on the TI Explorer Lisp machine and DEC Alpha.

The basic form of the task environment framework—the execution of interrelated computational tasks—is taken from several domain environment simulators (Carver & Lesser 1991, Cohen, Greenberg, Hart & Howe 1989, Durfee, Lesser & Corkill 1987)[2]. If this were the only impetus, the result might have been a simulator like Tileworld (Pollack & Ringuette 1990). However, formal research into multi-agent problem solving has been productive in specifying formal properties, and sometimes algorithms, for the control process by which the mental state of agents (termed variously: beliefs, desires, goals, intentions, etc.) causes the agents to perform particular actions (Cohen & Levesque 1990, Shoham 1991, Zlotkin & Rosenschein 1991). This research has helped to circumscribe the behaviors or actions that agents can produce based on their knowledge or beliefs. The final influence on TÆMS was the desire to avoid the individualistic agent-centered approaches that characterize most AI and Distributed AI. The concept of agency in TÆMS is based on simple notions of *execution*, *communication*, and *information gathering*. An agent is a locus of belief (state) and action. By separating as much as possible the notion of agency from the model of task environments, we do

[2] See also the discussion in Chapter 5.

not have to subscribe to particular agent architectures (which one would assume will be adapted to the task environment at hand), and we may ask questions about the inherent social nature of the task environment at hand (allowing that the concept of society may arise before the concept of individual agents (Gasser 1991)). Thus we might study how the concept and architecture of agents arises naturally from their task environment, rather than by starting with a pre-designed agent architecture. Another example is the search for so-called 'social laws' (Shoham & Tennenholtz 1992) that can be derived from a task environment to reduce coordination overhead by reducing some forms of uncertainty. The form of the framework is more detailed in structure than many organizational-theoretic models of organizational environments, such as Thompson's notions of pooled, sequential, and reciprocal processes (Thompson 1967), Burton and Obel's linear programs (Burton & Obel 1984), or Malone's queueing models (Malone 1987), but is influenced by them, and by the importance of environmental uncertainty, variance, and dependency that appear in contingency-theoretic and open systems views of organizations (Lawrence & Lorsch 1967, Galbraith 1977, Stinchcombe 1990, Scott 1987).

16.1.1 General Framework

The principle purpose of a TÆMS model is to analyze, explain, or predict the performance of a system or some component. While TÆMS does not establish a particular performance criteria, it focuses on providing two kinds of performance information: the temporal intervals of task executions, and the *quality* of the execution or its result. *Quality* is an intentionally vaguely-defined term that must be instantiated for a particular environment and performance criteria. Examples of *quality* measures include the precision, belief, or completeness of a task result. We will assume that *quality* is a single numeric term with an absolute scale, although the algebra can be extended to vector terms. In a computationally intensive AI system, several quantities—the quality produced by executing a task, the time taken to perform that task, the time when a task can be started, its deadline, and whether the task is necessary at all—are affected by the execution of other tasks. In real-time problem solving, alternate task execution methods may be available that trade-off time for quality. Agents do not have unlimited access to the environment; what an agent believes and what is really there may be different.

The model of environmental and task characteristics proposed has three levels: *objective*, *subjective*, and *generative*. The *objective* level describes the essential, 'real' task structure of a particular problem-solving situation or instance over time. It is roughly equivalent to a formal description of a single problem-solving situation such as those presented in (Durfee & Lesser 1991), without the information about particular agents. The *subjective* level describes how agents view and interact with the

problem-solving situation over time (e.g., how much does an agent know about what is really going on, and how much does it cost to find out—where the uncertainties are from the agent's point of view). The subjective level is essential for evaluating control algorithms, because while individual behavior and system performance can be measured objectively, agents must make decisions with only subjective information. In organizational theoretic terms, subjective *perception* can be used to predict agent actions or *outputs*, but unperceived, objective environmental characteristics can still affect performance (or *outcomes*) (Scott 1987). Finally, the *generative* level describes the statistical characteristics required to generate the objective and subjective situations in a domain. A generative level model consists of a description of the generative processes or distributions from which the range of alternative problem instances can be derived, and is used to study performance over a range of problems in an environment, avoiding single-instance examples.

16.1.2 Mathematical Framework

The *objective* level describes the essential structure of a particular problem-solving situation or instance over time. It focuses on how task interrelationships dynamically affect the *quality* and *duration* of each task. The basic model is that *task groups* (problem instances) appear in the environment at some frequency, and induce tasks T to be executed by the agents under study. Task groups are independent of one another (except for the use of computational or physical resources), but tasks within a single task group have interrelationships. Task groups or tasks may have deadlines $D(T)$. The *quality* of the execution or result of each task influences the *quality* of the task group result $Q(T)$ in a precise way; these quantities can be used to evaluate the performance of a system.

An individual task that has no subtasks is called a method M and is the smallest schedulable chunk of work (though some scheduling algorithms will allow some methods to be preempted, and some schedulers will schedule at multiple levels of abstraction). There may be more than one method to accomplish a task, and each method will take some amount of time and produce a result of some *quality*. Quality of an agent's performance on an individual task is a function of the timing and choice of agent actions ('local effects'), and possibly other (previous or future) task executions ('non-local effects'). When local or non-local effects exist between tasks that are known by more than one agent, we call them *coordination relationships* (Decker & Lesser 1993*a*). The basic purpose of the objective model is to formally specify how the execution and timing of tasks affect this measure of quality.

Local Effects: The *Subtask* Relationship. Task or task group quality ($Q(T)$) is based on the *subtask* relationship. This quality function is constructed recursively—each

task group consists of tasks, each of which consists of subtasks, etc.—until individual executable tasks (methods) are reached. 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 $Q(T,t) : [\text{tasks} \times \text{times}] \mapsto [\text{quality}]$ that returns the quality associated with T at time t. In a valid model, the directed graph induced by this relationship is acyclic (no task has itself for a direct or indirect subtask). The semantics of a particular environment are modeled by the appropriate choice of the quality function Q (e.g., minimum when all tasks need to be done, maximum for just one, summation, etc.).

Local Effects: Method Quality. Each method M at a time t will potentially produce (if executed by an agent) some *maximum quality* $\mathbf{q}(M,t)$ after some amount of elapsed time $\mathbf{d}(M,t)$. The execution of methods is interruptible, and if multiple methods for a single task are available, the agent may switch between them (typically, alternative methods tradeoff time and quality). We model the effect of interruptions, if any, and the reuse of partial results as non-local effects.

Let $\mathrm{Progress}(M,t)$ be the amount of progress at time t on the execution of M. If M were not interruptible and $\mathrm{Start}(M)$ and $\mathrm{Finish}(M)$ were the execution start time and finish time, respectively, of M, then:

$$\mathrm{Progress}(M,t) = \begin{cases} 0 & t \leq \mathrm{Start}(M) \\ t - \mathrm{Start}(M) & \mathrm{Start}(M) < t < \mathrm{Finish}(M) \\ \mathrm{Finish}(M) - \mathrm{Start}(M) & t \geq \mathrm{Finish}(M) \end{cases}$$

We could model the quality produced by a method $Q(M,t)$ using a linear growth function Q_{lin}:

$$Q_{\mathrm{lin}}(M,t) = \begin{cases} \frac{\mathrm{Progress}(M,t)}{\mathbf{d}(M,t)}(\mathbf{q}(M,t)) & \mathrm{Progress}(M,t) < \mathbf{d}(M,t) \\ \mathbf{q}(M,t) & \mathrm{Progress}(M,t) \geq \mathbf{d}(M,t) \end{cases}$$

Other models (besides linear quality functions) have been proposed and are used, to represent anytime (Boddy & Dean 1989), contract anytime (Russell & Zilberstein 1991), mandatory/optional (Liu, Lin, Shih, Yu, Chung & Zhao 1991), or design-to-time approachs (Garvey & Lesser 1993).

Non-local Effects. 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 $\mathsf{nle}(T, M, e, p_1, p_2, \ldots)$, 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 $e(T, M, t, d, q, p_1, p_2, \ldots) : [\text{task} \times \text{method} \times \text{time} \times \text{duration} \times \text{quality} \times \text{parameter 1} \times \text{parameter 2} \times \ldots] \mapsto [\text{duration} \times \text{quality}]$.

Some effects depend on the availability of information to an agent. For defining effects that depend on the availability of information, we define the helper function $Q_{\text{avail}}(T, t, A)$ that represents the quality of a task T 'available' to agent A at time t. If T was executed at A, $Q_{\text{avail}}(T, t, A) = Q(T, t)$. If T was executed (or is being executed) by another agent, then the 'available' quality is calculated from the last communication about T received at agent A prior to time t. Thus the local, subjective quality of a task result received by a remote agent may be different from the current quality at another agent—this is one of the important types of environmental uncertainty that our framework can express and for which we can develop coordination algorithm support.

Non-local effects are the most important part of the TÆMS framework, since they supply most of the characteristics that make one task environment unique and different from another. Typically a model will define different classes of effects, such as *causes*, *facilitates*, *cancels*, *constrains*, *inhibits*, and *enables* (Decker & Lesser 1992). This section contains a definition for one common class of effect that has been useful in modeling different environments; another definition will be presented in Section 16.1.4; see Decker & Lesser (1993*e*) for a more complete set of definitions. When non-local effects occur between methods associated with different agents, we call them *coordination relationships* (Decker & Lesser 1993*a*, Decker & Lesser 1992).

An important effect, used by the Partial Global Planning (PGP) algorithm (see Chapter 8 and Durfee & Lesser (1991)) but never formally defined, is the *facilitates* effect. Computationally, facilitation occurs when information from one task, often in the form of constraints, is provided that either reduces or changes the search space to make some other task easier to solve. For example, searching for a new book in the library will be faster after the book has been properly shelved, but you could search in the unordered stack of new books if you needed to. In our framework, one task may provide results to another task that *facilitate* the second task by decreasing the duration or increasing the quality of its partial result. Therefore the *facilitates* effect has two parameters (called *power* parameters) $0 \leq \phi_d \leq 1$ and $0 \leq \phi_q \leq 1$, that indicate the effect on duration and quality respectively. The effect varies not only through the power parameters, but also through the quality of the *facilitating* task available when work on the *facilitated* task starts (the ratio R). Remember that $\mathbf{q}(M, t)$ refers to the maximum quality possible to achieve at method M, and note that before work is started on a method, $\text{Start}(M) = t$ (i.e. formulae are evaluated as if execution were about to start).

$$\begin{aligned} R(T_a, s) &= \frac{\text{Q}_{\text{avail}}(T_a, s)}{\mathbf{q}(T_a, s)} \\ \textbf{facilitates}(T_a, M, t, d, q, \phi_d, \phi_q) &= [d(1 - \phi_d R(T_a, \text{Start}(M))), \\ &\quad q(1 + \phi_q R(T_a, \text{Start}(M)))] \end{aligned} \tag{16.1}$$

So if T_a is completed with maximal quality, and the result is received before M is started, then the duration $\mathbf{d}(M, t)$ will be decreased by a percentage equal to the duration power

ϕ_d of the *facilitates* effect, and similarly the maximum quality $\mathbf{q}(M, t)$ will be increased by a percentage equal to the quality power ϕ_q. The use of $\mathtt{Start}(M)$ in the definition indicates that communications about T_a received after the start of processing have no effect. In other work (Decker & Lesser 1993*a*) we explored the effects on coordination of a *facilitates* effect with varying duration power ϕ_d, and with $\phi_q = 0$.

Much more detail on the formal definitions of TÆMS objective, subjective, and generative structures, including the representation of the use of physical resources, can be found in Decker & Lesser (1993*e*).

16.1.3 Distributed Sensor Network Example

This example grows out of the set of single instance examples—Distributed Vehicle Monitoring Testbed (DVMT) scenarios—presented in (Durfee et al. 1987) (see also the discussion of the DVMT in Chapter 5). The authors of that paper compared the performance of several different coordination algorithms on these examples, and concluded that no one algorithm was always the best—not a very surprising result and one that can be viewed as the central tenet of contingency theory. This is the classic type of result that the TÆMS framework was created to address—we wish to *explain* this result, and better yet, to *predict* which algorithm will do the best in each situation. Here we do this by extending the analysis to take into account a statistical view of the environmental characteristics. The level of detail to which you build a model in TÆMS will depend on the question you wish to answer—here we wish to identify the characteristics of the Distributed Sensor Network (DSN) environment, or the organization of the agents, that cause one coordination algorithm to outperform another.

In a DSN problem like the DVMT (see Chapter 3), the movements of several independent vehicles will be detected over a period of time by one or more distinct sensors, where each sensor is associated with an agent. The performance of agents in such an environment is based on how long it takes them to create complete vehicle tracks, including the cost of communication. The organizational structure of the agents will imply the portions of each vehicle track that are sensed by each agent.

In our model of DSN problems, each vehicle track is modeled as a task group (problem instance). Several task groups (vehicle tracks) may occur simultaneously in a single problem solving *episode*. The simplest objective model is that each task group $\mathcal{T}_i$ is associated with a track of length l_i and has the following objective structure, based on a simplified version of the DVMT: (l_i) Vehicle Location Methods (VLM) that represent processing raw signal data at a single location resulting in a single vehicle location hypothesis; ($l_i - 1$) Vehicle Tracking Methods (VTM) that represent short tracks connecting the results of the VLM at time t with the results of the VLM at time $t + 1$; and one Vehicle-track Completion Method (VCM) that represents merging

all the VTMs together into a complete vehicle track hypothesis. *Non-local effects*, which relate the executions of tasks to one another, exist as shown in Figure 16.1—two VLMs *enable* each VTM, and all VTMs *enable* the lone VCM. This structure is fairly common in many other environments, where a large amount of detailed work needs to be done, the results of which are collected at a single location or agent and processed again (integrated), and so on up a hierarchy. The question is: to achieve a particular level of performance, how many agents are needed and how should they be organized in their unique and/or overlapping capabilities for accomplishing the necessary tasks. Coordination is used not only to accomplish the necessary transfer of results from one agent to another, but to avoid redundant work on the part of agents with overlapping capabilities, and also to potentially balance the workloads.

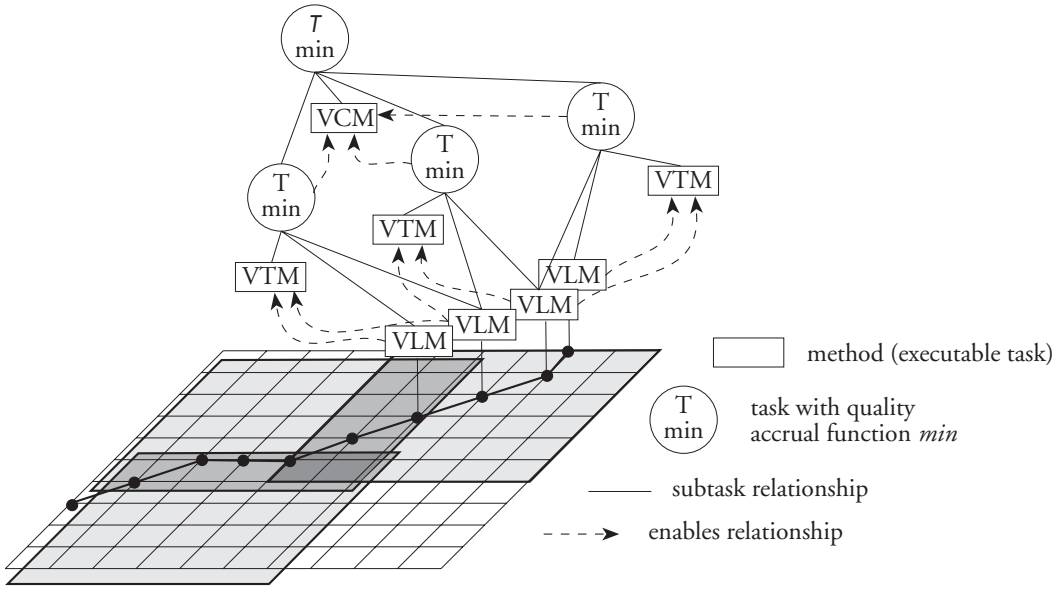

Figure 16.1: *Objective task structure associated with a single vehicle track.*

We have used this model to develop expressions for the expected value of, and confidence intervals on, the time of termination of a set of agents in any arbitrary DSN environment that has a static organizational structure and coordination algorithm (Decker & Lesser 1993*b*, Decker & Lesser 1993*c*). We have also used this model to analyze a dynamic, one-shot reorganization algorithm (and have shown when the extra overhead is worthwhile versus the static algorithm). In each case we can predict the effects of adding more agents, changing the relative cost of communication and computation, and changing how the agents are organized (in the range and overlap of their capabilities). These results were achieved by direct mathematical analysis of the model and verified through simulation in TÆMS.

Decker & Lesser (1993*d*) adds significant complexity to this basic model, making it more like the DVMT. For example, it adds the characteristic that each agent has two methods with which to deal with sensed data: a normal VLM and a 'level-hopping' VLM (the level-hopping VLM produces less quality than the full method but requires less time; see (Decker, Lesser & Whitehair 1990, Decker, Garvey, Humphrey & Lesser 1993) for this and other approximate methods; a similar technique can be used to model agents who have different capabilities such as processor speed). It also adds a representation of faulty sensors (a subjective level effect), and result sharing, as well as modeling the relationships that low quality results early on tend to make things harder to process at higher levels, that the first VLM execution provides information that slightly shortens the executions of other VLMs in the same vehicle track[3], and that a similar *facilitation* effect occurs at the tracking level (tracking gets easier as more of the track is completed). We can also represent subjective features such as faulty sensors, external noise, and 'ghost' tracks (like the example in Chapter 5). In general, having different subjective mappings for different agents or classes of agents allows us to model situations where some agents are more, less, or simply differently 'informed' than others.

16.1.4 Hospital Patient Scheduling Example

Let's look at a second example in a completely different domain, hospital patient scheduling. The following description is from an actual case study (Ow, Prietula & Hsu 1989):

> *Patients in General Hospital reside in* units *that are organized by branches of medicine, such as orthopedics or neurosurgery. Each day, physicians request certain tests and/or therapy to be performed as a part of the diagnosis and treatment of a patient. [. . .] Tests are performed by separate, independent, and distally located* ancillary departments *in the hospital. The radiology department, for example, provides X-ray services and may receive requests from a number of different units in the hospital.*

Furthermore, each test may interact with other tests in relationships such as *enables*, *requires* (must be performed after), and *inhibits* (test A's performance invalidates test B's result if A is performed during specified time period relative to B). Note that the unit secretaries (as scheduling agents) try to minimize the patients' stays in the hospital, while the ancillary secretaries (as scheduling agents) try to maximize equipment use (throughput) and minimize setup times.

[3] This is because the sensors have been properly configured with the correct signal processing algorithm parameters with which to sense that particular vehicle.

Figure 16.2 shows an subjective TÆMS task structure corresponding to an episode in this domain, and the subjective views of the unit and ancillary scheduling agents after four tests have been ordered. Note that quite a bit of detail can be captured in just the 'computational' aspects of the environment—in this case, the tasks use people's time, not a computer's. However, TÆMS can model in more detail the physical resources and job shop characteristics of the ancillaries if necessary (Decker & Lesser 1993*e*). Such detail is not necessary to analyze the solution presented by Ow et al. (1989), who propose a primary unit-ancillary protocol and a secondary ancillary-ancillary protocol.

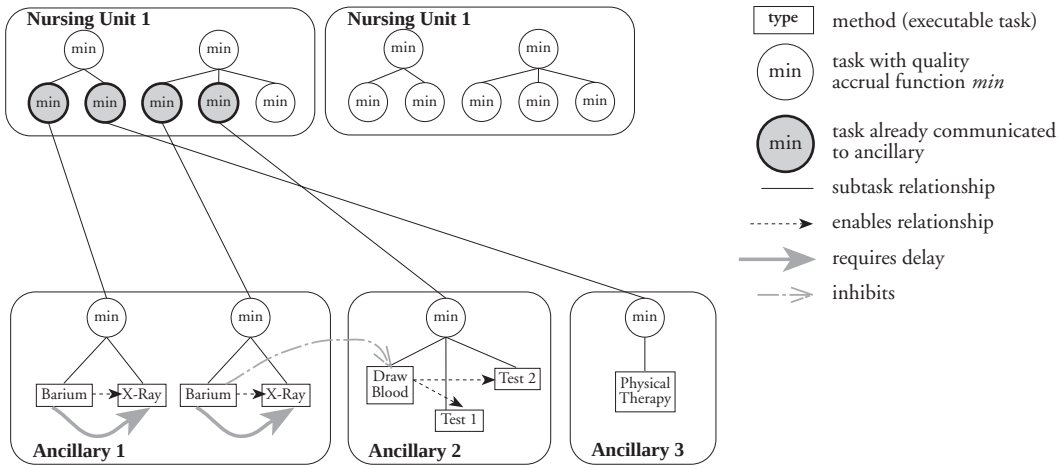

Figure 16.2: *High-level, subjective task structure for a typical hospital patient scheduling episode. The top task in each ancillary is really the same objective entity as the unit task it is linked to in the diagram.*

We use *min* (AND) to represent quality accrual because in general neither the nursing units nor ancillaries can change the doctor's orders—all tests must be done as prescribed. We have added two new non-local effects: *requires-delay* and *inhibits*. The first effect says that a certain amount δ of time must pass after executing one method before the second is enabled. We could define it mathematically as follows:

$$\textbf{requires-delay}(T_a, M, t, d, q, \delta) = \begin{cases} [\mathbf{d}_0(M), 0] & \mathtt{Start}(M) < \mathtt{Finish}(T_A) + \delta \\ [\mathbf{d}_0(M), \mathbf{q}_0(M)] & \mathtt{Start}(M) \geq \mathtt{Finish}(T_A) + \delta \end{cases} \tag{16.2}$$

The second relationship, A *inhibits* B, means that B will not produce any quality if executed in a certain window of time relative to the execution of A, and can be defined in a similar manner to the previous example.

16.2 Using TÆMS: Experiences

The first step in using TÆMS is to build a model of the environment of interest. Often the easiest way to go about this is to first model a single, detailed episode at the objective and subjective levels, and then to use that as a basis for a generative level model. TÆMS supplies several quality accumulation functions (*min*, *max*, *mean*, and *sum*) and predefined non-local effects (*enables*, *precedes*, *facilitates*, and *hinders* for tasks; *uses* and *limits* for resources), but you can extend the set. If you are interested only in general coordination problems, comparing architectures, etc., there is also a random structure generator. It takes parameters such as mean branching factor (Poisson), mean depth (Poisson), mean duration (exponential), user defined quality accumulation function distributions, etc. This basic structure can then be extended by *patterns*, such as 'random but consistent hard non-local effects' or 'fast fallback methods', each of which will take other parameters. The set of pre-defined patterns can also be extended. Several different task group templates can be defined with specified inter-arrival times for each, and the random structures can be recreated for paired-response experiments (running several algorithms or architectures on the same randomly-generated episode). At this point the analysis of the endogenous features of individual episodes, given generative parameters, can take place. If the model is of a real domain, a verification phase would also be mandatory.

TÆMS assumes only a simple view of agents as loci of belief (state) and action. A mathematical analysis will usually proceed with a formal specification of what actions an agent will take based on its current beliefs (for example, the agent-oriented programming model of Shoham (1991)). TÆMS provides a meta-structure for the agent's state-transition function that is divided into the following 4 parts: control, information gathering, communication, and method execution. First the control mechanisms assert (commit to) information-gathering, communication, and method execution actions and then these actions are computed one at a time, after which the cycle of meta-states repeats. Predefined information gathering actions in TÆMS (which trade computational time for information about the current task structure) include accessing newly arrived subjective task structures and uncovering coordination relationships to other agents' tasks. Predefined communication actions include trasmitting the current result of a method execution (or task), transmitting parts of the task structure, and meta-level communications. Method execution actions are, of course, the execution of methods in the structure by agents. TÆMS currently supports simple sequential execution, execution with monitoring, suspension, and preemption of methods, and is being extended to interruptable execution and internal agent parallelism.

We will briefly describe some of our experiences with TÆMS in distributed problem solving situations; we have also been using it to examine issues in single agent real-time scheduling and parallel scheduling (Garvey, Humphrey & Lesser 1993, Garvey &

Lesser 1993).

16.2.1 Analyzing a Simple Distributed Sensor Network

We have been developing a methodology for analyzing, explaining, and predicting behavior that focuses on chaining models of the environment, of coordination mechanisms, and of principled agent construction together and examining the flow of various uncertainties from the external environment through the models to the collective agent behaviors. We illustrated this with a mathematical analysis of a simplified distributed sensor network problem, verified through simulation (Decker & Lesser 1993*b*, Decker & Lesser 1993*c*).

The methodology we have been building uses the TÆMS framework and other DAI formalisms to build and chain together statistical models of coordination behavior that focus on the sources of uncertainty or variance in the environment and agents, and their effect on the (potentially multi-criteria) performance of the agents. We have used this methodology to develop expressions for the expected value of, and confidence intervals on, the time of termination of a set of agents in any arbitrary simple DSN environment that has a static organizational structure and coordination algorithm (Decker & Lesser 1993*b*). This paper shows how the distributions of objective parameters such as "the number of VLM methods seen by the maximally loaded agent" ($\hat{S}$) and "the max number of task groups seen by the same agent" ($\hat{N}$) can be defined from just the generative parameters $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$.[4]

For example, the total time until termination for an agent receiving an initial data set of size $\hat{S}$ is the time to do local work, combine results from other agents, and build the completed results, plus two communication and information gathering actions:

$$\hat{S}\mathbf{d}_0(\text{VLM}) + (\hat{S} - \hat{N})\mathbf{d}_0(\text{VTM}) + (a-1)\hat{N}\mathbf{d}_0(\text{VTM}) + \hat{N}\mathbf{d}_0(\text{VCM}) + 2\mathbf{d}_0(I) + 2\mathbf{d}_0(C) \quad (16.3)$$

We can use Eq. 16.3 as a predictor by combining it with the probabilities for the values of $\hat{S}$ and $\hat{N}$. Decker & Lesser (1993*b*) verify this model using the simulation component of TÆMS.

We have also used this model to analyze a dynamic, one-shot negotiated reorganization algorithm (and have shown when the extra overhead is worthwhile versus the static algorithm) (Decker & Lesser 1993*c*). In each case we can predict the effects of adding more agents, changing the relative cost of communication and computation, and changing how the agents are organized (in this case, by changing the range and

[4] A DSN environment $\mathcal{D}$ can be described by the following parameters: A, the number of sensor agents; η, the average number of vehicle tracks in an episode; r, the range of the agents' sensors; o, the amount of overlap between agent sensors; $\mathcal{T}$, which describes the task structure induced by each track given its length (derived from the other parameters).

overlap of their capabilities). These results were achieved by direct mathematical analysis of the model and verified through simulation in TÆMS. The interested reader will find the details in our papers (Decker & Lesser 1993*b*, Decker & Lesser 1993*c*).

16.2.2 Generalizing the Partial Global Planning Algorithm

The partial global planning (PGP) approach to distributed coordination (Durfee & Lesser 1991), also described in Chapter 8, increased the coordination of agents in a network by such methods as scheduling the timely generation of partial results, avoiding redundant activities, shifting tasks to idle nodes, and indicating compatibility between goals. It achieved this by recognizing certain *coordination relationships* (non-local effects across agents) among tasks in the Distributed Vehicle Monitoring Testbed (DVMT) environment and producing the appropriate scheduling constraints. In fact, because the local scheduler was so simple, the PGP mechanism supplanted it, recording and responding to many of the appropriate scheduling constraints itself. This work has had the characteristic of showing coordination techniques that are helpful, but not providing a deep analysis of when and why they are appropriate.

Generalized Partial Global Planning (GPGP) is really a family of algorithms; it is a extendable set of cooperative (team-oriented) coordination mechanisms built of modular components that work in conjunction with, but do not replace, a fully functional agent with a local scheduler. Each component can be added as required in reaction to the environment in which the agents find themselves a part. We have analyzed the performance of several GPGP family members through simulation in conjunction with a heuristic real-time local scheduler and randomly generated abstract task environments.

This approach views the coordination mechanism as *modulating* local control, not supplanting it — a two level process that makes a clear distinction between coordination behavior and local scheduling (Corkill & Lesser 1983). By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in partial global planning that occurs when there are multiple plans. By separating coordination from local scheduling, we can also take advantage of advances in real-time scheduling algorithms to produce CDPS systems that respond to real-time deadlines. We can also take advantage of local schedulers that have a great deal of domain scheduling knowledge already encoded within them. Finally, we can rely on humans as well in making local scheduling decisions. Our approach allows consideration of termination issues that were glossed over in the PGP work (where termination was handled by an external oracle).

The GPGP approach specifies three basic areas of the agent's coordination behavior: how and when to communicate and construct non-local views of the current problem solving situation; how and when to exchange the partial results of problem solving;

how and when to make and break *commitments* to other agents about what results will be available and when. The GPGP approach of recognizing and reacting to the characteristics of certain coordination relationships is shared with Von Martial's work on the *favor* relationship (v. Martial 1992). The use of commitments in the GPGP family of algorithms is based on the ideas of many other researchers (Cohen & Levesque 1990, Shoham 1991, Castelfranchi 1993, Jennings 1993). Each agent also has a heuristic local scheduler that decides what actions the agent should take and when, based on its current view of the problem solving situation (including the commitments it has made), and a utility function. The coordination mechanisms supply non-local views of problem solving to the local scheduler, including *what* non-local results will be available locally, and *when* they will be available. The local scheduler creates (and monitors the execution of) schedules that attempt to maximize system-wide quality through both local action and the use of non-local actions (committed to by other agents) without resorting to a complete global problem view.

One question that we have examined in TÆMS is the effects of agents exchanging non-local views (one of the GPGP coordination mechanisms—see Figure 16.3), and the decomposability of tasks in the environment (expressed as a probability on non-local effects), on system performance in terms of communication, total agent workloads, overall solution quality, and termination time. We showed that there were significant differences in communication and agent workloads due to both task decomposability and exchange of non-local views (but no interaction effects).

Another question we have examined is the effect of task structure variance on the performance of load balancing algorithms. This work is a logical follow-on to the analysis of static and dynamic negotiated reorganization summarized in the last section. A *static* organization divides the load up *a priori*—in this case, by randomly assigning redundant tasks to agents. A *one-shot dynamic* reorganization, like that analyzed by Decker & Lesser (1993*c*), negotiates the handling of redundant tasks on the basis of the *expected* load on other agents. A *meta-level communication* (MLC) reorganization negotiates the handling of redundant tasks on the basis of actual information about the particular problem-solving episode at hand. Because it requires extra communication, the MLC reorganization is more expensive, but the extra information pays off as the *variance* in static agent loads grows. Figure 16.4 shows how the probability of terminating more quickly with the MLC load balancing mechanism grows as the standard deviation in the total durations of redundant tasks at each agent grows.

More information about Generalized Partial Global Planning, including the formal definition of five example coordination mechanisms and experimental methodology for determining when a mechanism is appropriate, can be found in Decker & Lesser (1994).

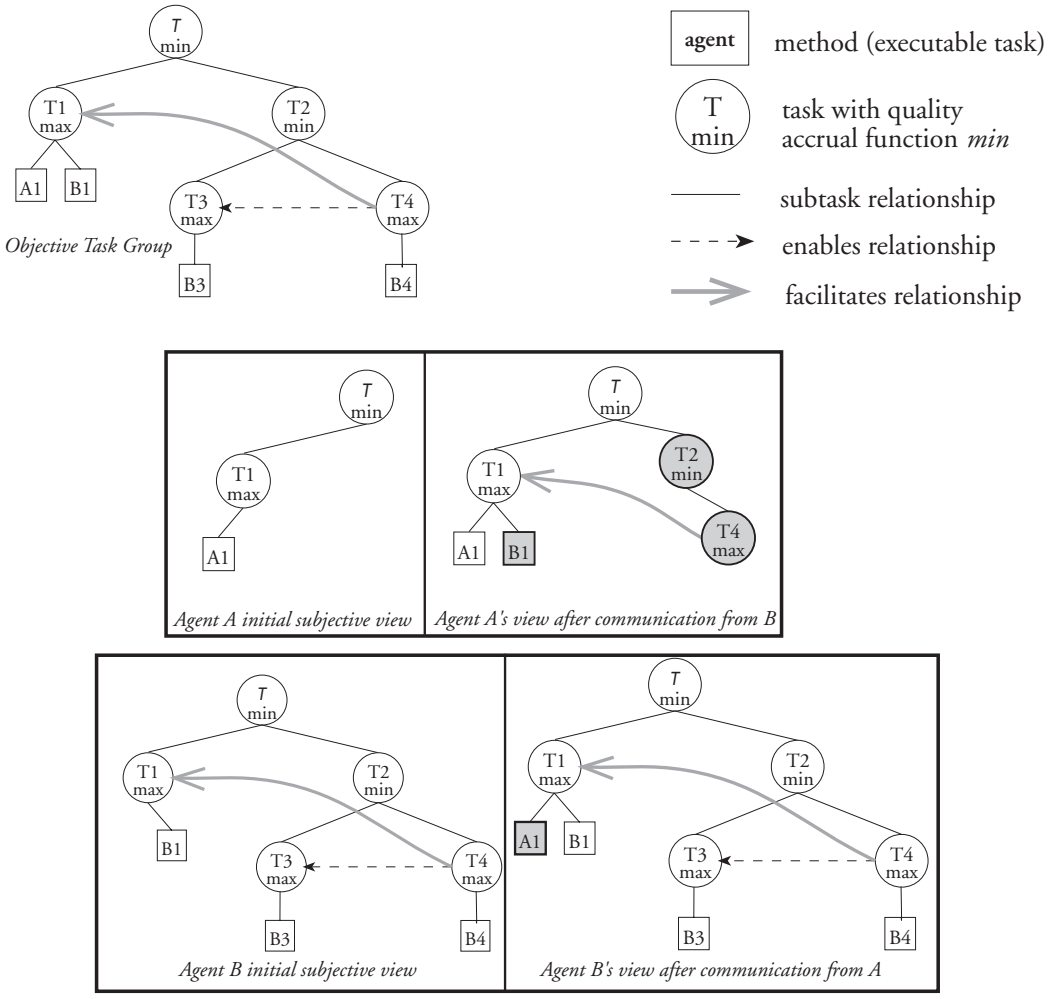

Figure 16.3: *Example of the local view at Agents A and B before and after the team shares private information to create a partial non-local view. Method boxes are labelled by the name of the agent that has the method.*

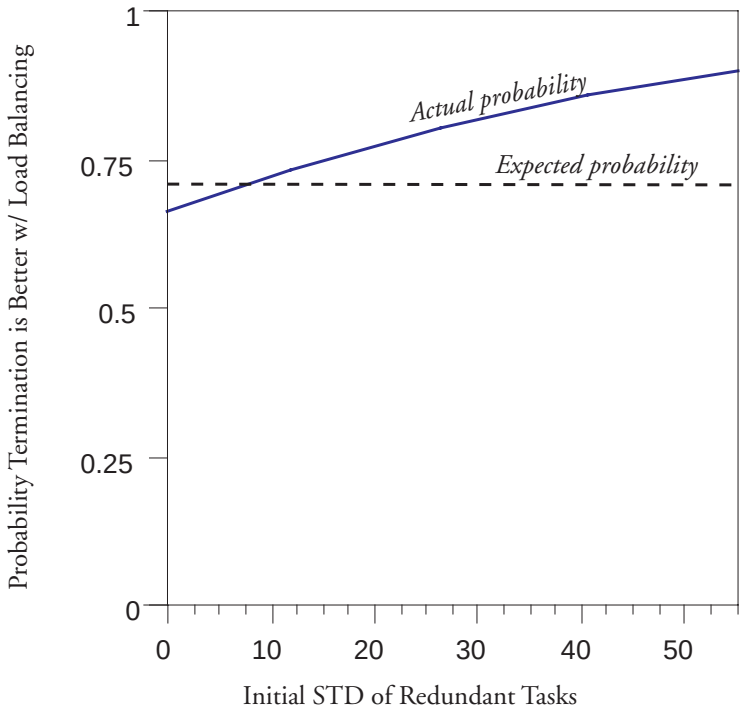

Figure 16.4: *Probability that MLC load balancing will terminate more quickly than static load balancing, fitted using a loglinear model from actual* TÆMS *simulation data.*

16.3 Summary and Future Work

TÆMS is a framework for modeling complex task environments. TÆMS exists as both a language for stating general hypotheses or theories and as a system for simulation. The important features of TÆMS include its layered description of environments (*objective* reality, *subjective* mapping to agent beliefs, *generative* description of the other levels across single instances); its acceptance of any performance criteria (based on temporal location and *quality* of task executions); and its non-agent-centered point of view that can be used by researchers working in either formal systems of mental-state-induced behavior or experimental methodologies. TÆMS provides environmental and behavioral structures and features with which to state and test theories about the control of agents in complex computational domains, such as how decisions made in scheduling one task will affect the utility and performance characteristics of other tasks.

TÆMS is not only a mathematical framework, but also a simulation language for executing and experimenting with models directly. The TÆMS simulator supports the graphical display of generated subjective and objective task structures, agent actions, and statistical data collection in CLOS on the TI Explorer and DEC Alpha. These features help in both the model-building stage and the verification stage. The TÆMS simulator is being used not only for research into the coordination of distributed problem solvers, but also for research into real-time scheduling of a single agent (Garvey & Lesser 1993), scheduling at an agent with parallel processing resources available, learning coordination relationships, and computational organizational design.

TÆMS does not at this time automatically learn models or automatically verify them. While we have taken initial steps at designing a methodology for verification (see (Decker & Lesser 1993*b*)), this is still an open area of research (Cohen 1991). Work now being done includes building new models of different environments that include physical resource constraints, such as hospital patient scheduling and airport resource scheduling. Other extensions we are now working on include more helpful facilities for specifying dynamic objective and subjective models that change structure as the result of agent actions (like the Tower of Babel (Ishida 1992)). Such models add yet another important source of uncertainty that can be a factor in influencing agent organization.

Current applications being developed using TÆMS and the GPGP approach include distributed support for human scheduling in concurrent engineering environments like the ARCADIA software engineering development environment (Taylor, Belz, Clarke, Osterweil, Selby, Wileden, Wolfe & Young 1988). Part of the ARCADIA environment is directly concerned with representing and tracking the state of: software development processes (including interrelationships), the products being produced, and the resources available. This information can be used to assist users in task selection by developing a User Coordination Assistant Agent (UCAA) that keeps track of a workstation user's

current agenda of tasks and presents a possible schedule (ordering) of these tasks according to user- and domain-directed preferences. Such an agenda is not developed in isolation, but rather through a distributed coordination process using multiple coordination mechanisms triggered by the coordination relationships between the task structures of the different agents involved.

Another application being developed is one using TÆMS to represent and reason about multi-agent information retrieval on the Internet. The idea here is to have teams of agents search for useful information. The results of some team member's searches will have an impact on other team members in a way that can be modeled with /tems/. Algorithms, such as new GPGP mechanisms, can then be developed to take advantage of these opportunities. Another direction we are pursuing is to use /tems/ to build computational versions of organizational theories, such as Williamson's Transaction Cost Economics (Williamson 1975). Finally, we wish to expand our analyses beyond the questions of scheduling and coordination to questions about negotiation strategies, emergent agent/society behavior, and organizational self-design.

Acknowledgements

Many people have worked on the development of TÆMS including Alan J. Garvey, Marty A. Humphrey, and Victor R. Lesser. This work was supported in part by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the U. S. Government and no official endorsement should be inferred.

Bibliography

Boddy, M. & Dean, T. (1989), Solving time-dependent planning problems, *in* 'Proceedings of the Eleventh International Joint Conference on Artificial Intelligence', Detroit, pp. 979–984.

Burton, R. M. & Obel, B. (1984), *Designing Efficient Organizations: Modelling and Experimentation*, North Holland, Amsterdam.

Carver, N. & Lesser, V. (1991), A new framework for sensor interpretation: Planning to resolve sources of uncertainty, *in* 'Proceedings of the Ninth National Conference on Artificial Intelligence', pp. 724–731.

Castelfranchi, C. (1993), Commitments:from individual intentions to groups and organizations, *in* M. Prietula, ed., 'AI and theories of groups & organizations: Conceptual and Empirical Research', AAAI Workshop. Working Notes.

Cohen, P., Greenberg, M., Hart, D. & Howe, A. (1989), 'Trial by fire: Understanding the design requirements for agents in complex environments', *AI Magazine* **10**(3), 33–48. Also COINS-TR-89-61.

Cohen, P. R. (1991), 'A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart?', *AI Magazine* **12**(1), 16–41.

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

Corkill, D. D. & Lesser, V. R. (1983), The use of meta-level control for coordination in a distributed problem solving network, *in* 'Proceedings of the Eighth International Joint Conference on Artificial Intelligence', Karlsruhe, Germany, pp. 748–755.

Decker, K. S. & Lesser, V. R. (1992), 'Generalizing the partial global planning algorithm', *International Journal of Intelligent and Cooperative Information Systems* **1**(2), 319–346.

Decker, K. S. & Lesser, V. R. (1993*a*), 'Analyzing a quantitative coordination relationship', *Group Decision and Negotiation* **2**(3), 195–217.

Decker, K. S. & Lesser, V. R. (1993*b*), An approach to analyzing the need for meta-level communication, *in* 'Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence', Chambéry, France, pp. 360–366.

Decker, K. S. & Lesser, V. R. (1993*c*), A one-shot dynamic coordination algorithm for distributed sensor networks, *in* 'Proceedings of the Eleventh National Conference on Artificial Intelligence', Washington, pp. 210–216.

Decker, K. S. & Lesser, V. R. (1993*d*), Quantitative modeling of complex computational task environments, *in* 'Proceedings of the Eleventh National Conference on Artificial Intelligence', Washington, pp. 217–224.

Decker, K. S. & Lesser, V. R. (1993*e*), 'Quantitative modeling of complex environments', *International Journal of Intelligent Systems in Accounting, Finance, and Management* **2**(4), 215–234. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

Decker, K. S. & Lesser, V. R. (1994), Designing a family of coordination algorithms, *in* 'Proceedings of the Thirteenth International Workshop on Distributed AI', Seattle, WA, pp. 65–84. Also UMass CS-TR-94-14.

Decker, K. S., Garvey, A. J., Humphrey, M. A. & Lesser, V. R. (1993), 'A real-time control architecture for an approximate processing blackboard system', *International Journal of Pattern Recognition and Artificial Intelligence* **7**(2), 265–284.

Decker, K. S., Lesser, V. R. & Whitehair, R. C. (1990), 'Extending a blackboard architecture for approximate processing', *The Journal of Real-Time Systems* **2**(1/2), 47–79.

Durfee, E. & Lesser, V. (1991), 'Partial global planning: A coordination framework for distributed hypothesis formation', *IEEE Transactions on Systems, Man, and Cybernetics* **21**(5), 1167–1183.

Durfee, E. H., Lesser, V. R. & Corkill, D. D. (1987), 'Coherent cooperation among communicating problem solvers', *IEEE Transactions on Computers* **36**(11), 1275–1291.

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

Garvey, A. & Lesser, V. (1993), 'Design-to-time real-time scheduling', *IEEE Transactions on Systems, Man and Cybernetics* **23**(6), 1491–1502.

Garvey, A., Humphrey, M. & Lesser, V. (1993), Task interdependencies in design-to-time real-time scheduling, *in* 'Proceedings of the Eleventh National Conference on Artificial Intelligence', Washington, D.C., pp. 580–585.

Gasser, L. (1991), 'Social conceptions of knowledge and action', *Artificial Intelligence* **47**(1), 107–138.

Ishida, T. (1992), Tower of Babel: Towards organization-centered problem solving, *in* 'Proceedings of the 11th Workshop on Distributed Artificial Intelligence', The Homestead, Michigan, pp. 141–153.

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. & Lorsch, J. (1967), *Organization and Environment*, Harvard University Press, Cambridge, MA.

Liu, J. W. S., Lin, K. J., Shih, W. K., Yu, A. C., Chung, J. Y. & Zhao, W. (1991), 'Algorithms for scheduling imprecise computations', *IEEE Computer* **24**(5), 58–68.

Malone, T. W. (1987), 'Modeling coordination in organizations and markets', *Management Science* **33**, 1317–1332.

Ow, P. S., Prietula, M. J. & Hsu, W. (1989), Configuring knowledge-based systems to organizational structures: Issues and examples in multiple agent support, *in* L. F. Pau, J. Motiwalla, Y. H. Pao & H. H. Teh, eds, 'Expert Systems in Economics, Banking, and Management', North-Holland, Amsterdam, pp. 309–318.

Pollack, M. E. & Ringuette, M. (1990), Introducing Tileworld: Experimentally evaluating agent architectures, *in* 'Proceedings of the Eighth National Conference on Artificial Intelligence', Boston, pp. 183–189.

Russell, S. J. & Zilberstein, S. (1991), Composing real-time systems, *in* 'Proceedings of the Twelfth International Joint Conference on Artificial Intelligence', Sydney, Australia, pp. 212–217.

Scott, W. R. (1987), *Organizations: Rational, Natural, and Open Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

Shoham, Y. (1991), AGENT0: A simple agent language and its interpreter, *in* 'Proceedings of the Ninth National Conference on Artificial Intelligence', Anaheim, pp. 704–709.

Shoham, Y. & Tennenholtz, M. (1992), On the synthesis of useful social laws for artificial agnet societies (preliminary report), *in* 'Proceedings of the Tenth National Conference on Artificial Intelligence', San Jose, pp. 276–281.

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

Taylor, R., Belz, F., Clarke, L., Osterweil, L., Selby, W., Wileden, J., Wolfe, A. & Young, M. (1988), Foundations for the Arcadia environment architecture, *in* 'Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments', Boston, MA, pp. 1–12.

Thompson, J. D. (1967), *Organizations in Action*, McGraw-Hill, New York.

v. Martial, F. (1992), *Coordinating Plans of Autonomous Agents*, Springer-Verlag, Berlin. Lecture Notes in Artificial Intelligence no. 610.

Williamson, O. E. (1975), *Markets and Hierarchies: Analysis and Antitrust Implications*, The Free Press, New York.

Zlotkin, G. & Rosenschein, J. S. (1991), Incomplete information and deception in multi-agent negotiation, *in* 'Proceedings of the Twelfth International Joint Conference on Artificial Intelligence', Sydney, Australia, pp. 225–231.