BAA 96-40

Proposal

Technical Topic Area: Adaptive Architectures for Survivable Systems

Enhancing Survivability with Distributed Adaptive Coordination

Technical Point of Contact:	Professor Victor R. Lesser Phone: 413-545-1322; Fax: 413-545-1249. Email: lesser@cs.umass.edu Computer Science Department, Box 34610 LGRC University of Massachusetts Amherst, MA 01003
Administrative Point of Contact:	Carol Sprague, Acting Associate Director Phone: 413-545-0698; Fax: 413-545-1202 Office of Grant and Contract Administration, Munson Hall University of Massachusetts Amherst, MA 01003
Type of Business:	Other Educational

Period of Performance: 36 Months
Total Cost: $1,174,888
Year 1: $392,479
— Extend existing simulation testbed for studying survivability
— Extend coordination approach for dynamic adaptation
— Develop learning, detection, and diagnosis algorithms
Year 2: $394,335
— Design and analyze new coordination/organizational mechanisms
— Design and analyze new learning, detection, and diagnosis algorithms
— Produce initial integrated prototype shell
Year 3: $388,074
— Delivery of all algorithms, analyses, evaluations, and reference implementations
— Delivery of final simulation testbed
— Delivery of final integrated prototype shell

Section II. Summary

A. Innovative Claims

We view information systems as computational organizations consisting of *agents*. Each agent performs particular *tasks* using one or more *methods*. An agent may carry out an entire task itself, or it may coordinate with other agents to perform parts of the task. We view *survivable* information systems as computational organizations that redesign themselves in response to threats and opportunities: organizations that learn, reason about, and adapt to intrusions, failures, and new resources.

We will develop an architecture that integrates three key technologies for organizational design. *Distributed coordination algorithms* adapt to changing situations and respond to intrusions by creating new computational organizations. Using these algorithms, agents accomplish a large number of interrelated tasks and schedule the order of those tasks to achieve goals for the quality of results and the time required to return those results. Because agents select among available methods, they can exploit variability, diversity, and redundancy in available methods. Distributed coordination is difficult to disrupt and degrades gracefully as agents are compromised. We will build on our previous work [16, 21] to produce coordination algorithms specifically for designed to enhance survivability. These algorithms will exploit knowledge about the system's goals and characteristics of the current agents, methods, and resources.

A second technology, *learning algorithms*, model the characteristics of agents, tasks, methods, and resources. Such models are vital for effective coordination, because coordination algorithms rely on accurate estimates of parameters such as the duration of certain tasks and the quality of their results. Many parameters will change when portions of the system come under attack. We will build on our previous work in machine learning and statistics [19, 48, 35] to produce algorithms that learn to accurately predict important parameters.

A final technology, *detection and diagnosis algorithms*, use learned models to detect changes in the behavior of agents and environments and diagnose the causes of those changes. Such techniques are important to direct the search for suitable organizational structures. Without such reasoning, coordination would be far less effective and efficient. Detection and diagnosis algorithms also can pinpoint the cause of behavioral changes and identify where an intrusion has occurred. We will build on our previous work on modeling and diagnosis in single agents [33] and our recent work on dynamic adaptation of coordination strategies for multiagent network diagnosis using explanation-based learning [55].

Finally, the system architecture itself will integrate the actions of these and other survivability technologies in a sophisticated way. Our agent-based approach will allow us to integrate the results of sophisticated intrusion detection and diagnosis components developed by other survivability researchers. Our work represents an important point in the design space that, to our knowledge, no one else is pursuing. It will produce systems that learn, reason, and respond to intrusions in a domain-independent way. The systems will exploit both learned and given knowledge about agents, tasks, methods, and resources. Even when significant portions of such systems are compromised, the remaining portions will quickly adapt and continue to perform critical functions.

B. Deliverables

Our research will be a medium-scale experimental and prototyping effort. The deliverables fall into four areas: 1) An integrated system architecture prototype; 2) Algorithms and prototype implementations of key technologies for coordination, learning, detection, and diagnosis; 3) A testbed for modeling and simulation of survivability issues in complex systems; and 4) Extensive evaluation of individual technologies and the integrated system prototype. *We make no proprietary claims with respect to this research.*

Integrated Architecture Prototype

The key technologies each can enhance survivability individually, but they also have important synergies. We will integrate all the technologies into a single prototype agent shell in order to evaluate their interactions and demonstrate how their synergies can enhance survivability. The integrated prototype agent shell will be delivered along with a detailed design document, illustrating the technical issues involved in integration. In addition, we will provide detailed specifications and sample implementations of wrappers that integrate legacy systems. These wrappers will provide the necessary functions to turn standard applications into cooperative, adaptive agents that take advantage of the newly developed technologies.

Algorithms and Reference Implementations

We will develop and deliver reference implementations and technical papers for:

- *Coordination algorithms* that use modular coordination mechanisms to create systems of agents capable of distributed organizational design. Some mechanisms will be predefined to respond to general, domain-independent survivability problems, others will be learned responses to particular domain-specific situations.
- *Learning algorithms* that statistically model important parameters of agents, tasks, methods, and resources and the relationships among these parameters. This learned knowledge will improve the efficiency and reliability of coordination.
- *Detection and diagnosis algorithms* that use learned models to detect changes in system operation and diagnose their causes. Detection involves monitoring important parameters and identifying statistically significant deviations from expected behavior. Diagnosis identifies likely causes of such changes.

Testbed for Modeling and Simulation

We will extend our existing framework for modeling and simulation of multiagent systems to encompass all of these technologies. This framework, Task Analysis, Environmental Modeling, and Simulation (TÆMS), already encompasses coordination algorithms. TÆMS underlies much of our work in multiagent systems. We will apply TÆMS to model and evaluate systems constructed with our algorithms.

We will also extend our existing simulation testbed specifically to model mixed-resource environments and to evaluate system survivability. During its development, we will work with

other contractors to be sure our simulations are realistic and that our conceptions of legacy systems, tasks, shared resources, and intrusions are accurate. The use of such a testbed will greatly enhance our ability to carefully and completely evaluate solutions in many different (and potentially hard to recreate) situations.

Evaluation

We will extensively evaluate individual prototypes and integrated systems. Much of our previous work on multiagent coordination and learning is based on close attention to evaluation — a phase all too often overlooked in AI research [4, 5]. The TÆMS framework was specifically designed to support the characterization and evaluation of multiagent systems, and we will make use of the enhanced version of TÆMS for this purpose. We will produce additional analyses and analytical models of algorithms where possible. We will continue our commitment to applying statistical analysis techniques to evaluating our systems. Such techniques include paired-response studies made possible by sophisticated simulation techniques. While such techniques are only the beginning of good evaluation, they often can reveal subtle indicators of performance and tease apart complex inter-relationships.

C. Cost, Schedule, and Milestones

The proposed effort is for three years at a total cost of $1,174,888 (first year $392,479; second year $394,335; third year $388,074).

Schedule and Milestones

Year 1:

- Extend existing simulation testbed to model survivability issues and complex resource constraints. *Milestone: Demonstrate extended testbed that can handle resource constraints and model typical survivability issues* ($130,827)
- Extend distributed coordination approach to support both procedural and declarative mechanisms for learning. *Milestone: Demonstrate declarative mechanisms for using declarative organizational knowledge.* ($130,827)
- Begin to develop algorithms for univariate and multivariate learning. ($65,413)
- Begin to develop algorithms for detection and diagnosis of behavior during normal operation (e.g., use of shared resources). ($65,413)

Year 2:

- Extend distributed coordination algorithms for task sharing, self-interested agents, and organizational design. *Milestone: demonstration, technical papers* ($131,445)
- Complete development of algorithms for multivariate learning. *Milestone: demonstration, technical papers.* ($65,722)
- Develop algorithms for detection and diagnosis of behavior during abnormal operation. *Milestone: Demonstration of intrusion detection, technical papers.* ($65,722)
- Produce and evaluate integrated prototype shell with coordination, learning, detection, and diagnosis *Milestone: initial deployment of integrated prototype.* ($131,445)

Year 3:

- Enhance simulation testbed for additional realism with a greater range of agents, tasks, resources, and intrusions. Information for this effort will be sought from other program participants. Complete documentation. *Milestone: Delivery of the final testbed.* ($59,360)
- Complete extension and evaluation of distributed coordination algorithms for organizational design. *Milestone: Deliver final algorithms, reference implementations, and analyses.* ($97,018)
- Complete extension and evaluation of learning, detection, and diagnosis algorithms for organizational design. *Milestone: Deliver final algorithms, reference implementations, and analyses.* ($97,018)
- Create interface specifications for legacy subsystems. *Milestone: demonstration.* ($32,339)
- Extend integrated prototype with final coordination, learning, detection, and diagnosis algorithms. ($70,000)
- Evaluate and document integrated prototype. *Milestone: Deliver final integrated prototype and documentation* ($32,339)

D. Technical Rationale, Technical Approach and Constructive Plan

We view survivable systems as computational organizations that can redesign themselves in response to threats and opportunities. A central assumption of organizational design is that there exist alternative ways to accomplish tasks in terms of the agents, methods, and resources used. In systems of any complexity, such alternatives *do* exist, and systems constructed for survivability will intentionally contain them. Under these conditions, the central challenges of survivability are making effective use of the available alternatives, acquiring knowledge about those alternatives, and making inferences based on that knowledge,

The three technologies we will develop address each of these challenges. First, *distributed coordination* enables systems to organize themselves and accomplish critical tasks with available resources. Second, *learning* enables systems to improve with experience and model themselves and their environment. Learning creates the knowledge necessary for organizational design. Third, *detection and diagnosis* enables systems to use models that detect important changes and diagnose why those changes occurred. Detection and diagnosis provides the capability to effectively use learned knowledge for organizational design.

Below we provide an example of several representative problems, and how our technologies would be applied to enhance survivability. Finally, we indicate the technical bases for our confidence that such developments are feasible—we have already developed a few key technologies, albeit for different domains.

An Example: Distributed Data Processing

We can illustrate our proposed work most easily through an example. Consider a distributed information system where large amounts of raw satellite data are processed at geographically-dispersed data centers.

We view each center as a group of agents that coordinate among themselves and with agents at other centers to process the satellite data. Although the agents at each center have processing tasks for which they are best suited, centers can accomplish other tasks with lower efficiency and lower output quality. This redundancy and variability enhances survivability by insuring that critical tasks can still be accomplished even if one or more centers are impaired or unavailable. Processing tasks need not be monolithic; they can consist of several inter-related subtasks that can be accomplished by different agents in different ways. One large processing task may be accomplished by agents at several data centers, each of which may rely upon other agents for intermediate results.

Problem 1: Major failure The most obvious threat to the survivability of this system involves a major failure at one of the data centers. This could result from direct physical attack, complete communications disruption, or an electronic intrusion with catastrophic results.

Our work addresses this problem by using coordination and learning algorithms. First, we would use *distributed coordination algorithms* to shift processing from the failed center to other centers. The organization of the system would be redesigned in a distributed manner — agents would take on different tasks or would cooperate with different agents to accomplish the same tasks. The algorithms would effectively match agents to tasks, and schedule tasks so that high-priority processing is given precedence. Second, we would use *learning algorithms* to

improve the effectiveness of coordination by updating knowledge about agents, methods, tasks, and resources. Shifting processing to different centers is likely to change parameters that affect coordination. For example, some communication channels may become busier as agents take on tasks previous accomplished by agents at other centers. In such cases, learning algorithms would update the value of the parameters that characterize the speed of communication.

Problem 2: Spurious results A more subtle and potentially dangerous threat to survivability is if one or more agents are altered by an intruder to provide spurious results. Such an intrusion could affect an agent's resource use, the duration of its processing, or the quality of its results.

Our work addresses this problem by using *detection and diagnosis* techniques. First, we would use these techniques to discover that the characteristics of the agent have changed and to attempt a diagnosis of the cause. Detection might occur by monitoring the agent itself, or by monitoring the characteristics of higher-level task. In the latter case, our methods provide a way of tracing the pedigree of results in order to discover which subtask is responsible for the detected change. After the compromised agent is identified, it can be isolated from further participation in the system by updating the knowledge of relevant agents, who will shift processing away from the compromised agent as they participate in *distributed coordination*. This dynamic redesign of the agents' organization makes use of variation and redundancy in agents and the methods that they use to accomplish tasks. These alternatives will be used when a previously preferred agent or method becomes compromised. Subsequently, the compromised agent could be occasionally checked to see whether to reintroduce it to the organization.

Problem 3: Communications slowdown A third problem that might arise in the context of survivability involves a drop in the speed of communication. This could result from intentional interference (causing lost packets and requiring retransmission) or from resource contention.

Our work addresses this problem by using *detection and diagnosis* techniques to design a new organizational structures for the system. First, we would use the techniques to identify that a communication slowdown has occurred and to suggest possible causes for the slowdown. Based on those inferences, *coordination algorithms* would develop a new organizational design in order to minimize communication over the affected links. Such an organization might make each data center more autonomous, or communicate results at a higher level of detail, thus requiring less overall communication.

The TÆMS Framework

TÆMS [19, 14, 15] is designed to model the problem-solving activities of an intelligent agent operating in environments with deadlines and limits on resource usage, where the information required for the optimal performance of a computational task may not be available, where the results of multiple agents' computations (to inter-dependent subproblems) may need to be aggregated in order to solve a high-level goal, and where an agent may be contributing concurrently to the solution of multiple goals. The TÆMS approach is domain-independent and is based on explicitly modeling what is known and uncertain about agent goals, hierarchical task structures, alternative actions, sets of resources, and external agent capabilities. We believe that TÆMS is at the correct level of detail—not too abstract as in game-theoretic [30, 50, 31] or

queueing-theory [40] models, and not wedded to a detailed, specific internal agent architecture or domain problem [8, 39, 24, 25].

The implication of the real-time requirement is that the agents should be able to choose among alternative methods for achieving a task producing results of varying quality based on the available time and other resource constraints. TÆMS is able to specify that there are multiple ways to accomplish a goal and that these alternatives make trade-offs regarding the execution time and other types of resource usage (see Figure 4).

The possibility of incomplete local information will, by necessity, lead to agents working in a satisficing mode in which problem solving is structured to operate effectively with missing information. For this reason, the representation includes what we will call "soft" coordination relationships that define the implications, in terms of result quality, computation duration, and other costs (e.g. resource usage), 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 in terms 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.

We will extend the TÆMS framework in several ways, including:

- *Resource Models* — Currently, TÆMS is used to model computational environments without regard to physical resources. We will extend TÆMS to be able to explicitly model physical resources based on ideas developed in [13]. From the point of view of task environment specification, the effect of a resource, with respect to computational tasks, is to change the duration, cost, and/or quality of a task, and so we represent this as a set of non-local effects (NLEs) just as we model subtask interrelationships.

- *Representation for Organizational Design* — The TÆMS modeling framework allows the following type of domain-independent reasoning to be implemented: predictions about future agent activity (their timing properties, resource requirements, and the importance and quality of their output), how activities relate to each other not only in terms of precedence but more complex relationships which effect their timing and output characteristics, how results are aggregated into higher level output, and how the quality of that output relates to the quality of the input. These types of reasoning are crucial to developing dynamic coordination strategies and constructing agent organizational structures.

Although developed for the representational needs of short-term, dynamic coordination, we believe that the ability of TÆMS to statistically model alternative agent activities and physical resources will allow us to easily extend the framework to meet the needs of representing complex organizational structures and long-term characteristics of the environment.

Developing Key Technologies in the context of TÆMS

We have already developed coordination algorithms based on TÆMS. Generalized Partial Global Planning (GPGP) is a family of distributed coordination algorithms developed in

our Laboratory [21]. GPGP algorithms coordinate heterogeneous agents to achieve efficient computational results under time and quality constraints. GPGP agents have a database of beliefs, a local scheduler, and approaches to coordinating their actions with those of other agents. Agents schedule their own subtasks based on limited understanding of an overall task, while coordinating with other agents. GPGP algorithms are able to achieve highly coordinated processing without centralized control.

TÆMS provides a strong framework within which learning algorithms can be developed. First, it identifies a set of important parameters needed for effective coordination. Accurate estimates of these parameters will be a central task for the learning algorithms. Second, TÆMS task structures provide guidance about what features to consider while learning multivariate models. Agents and methods linked within TÆMS are natural candidates for providing features for learning.

The technologies we propose build on our previous work in learning. We have developed learning algorithms that account for the statistical effects of extensive search [35, 36], and these algorithms consistently outperform conventional learning algorithms. We have also developed algorithms for learning situation-specific coordination rules [55]. This work was in the context of a sophisticated multi-agent system for network diagnosis. We have also recently extended GPGP so that situation-specific control strategies could be learned on-line [48].

Finally, TÆMS assists greatly in developing diagnosis techniques. TÆMS task structures represent the pedigree of results, allowing deviations in high-level results to be traced to specific methods that yielded unexpected or faulty low-level results. This provides the capability of localizing and isolating intrusions quickly, a definite aid to survivability. We will build on our previous work in error detection and diagnosis where we developed algorithms for detecting inappropriate activities that degrade system performance [55]. We will pursue other diagnosis techniques as well, particularly the use of learned models, but TÆMS provides a foundation on which to add learned knowledge. We also feel that the explicit representation of agent activities, the organization of agents, and the state of the network allows us to integrate other approaches to survivability, developed elsewhere, with our approach.

E. Conflict of Interest Statement

The only DARPA connection that Victor Lesser of the University of Massachusetts and Keith Decker (as an independent contractor) currently have is through a subcontract from Boeing Helicopter which is being supported by DARPA under the RaDEO program (contract number 70NANB6H0074). This subcontract involves applying the TÆMS/GPGP framework to the problem of coordinating a group of engineers at workstations performing concurrent engineering tasks.

The only DARPA connection that David Jensen of the University of Massachusetts currently has is through support as a research scientist on a subcontract through Sterling Software, Inc./DARPA, F30602-95-C-0257 (Evaluation Methods for Complex, Intelligent, Information-Driven Systems) and as a research scientist on a DARPA/Rome Laboratory Planning Initiative, F30602-95-1-0021 (A Substrate and Tools for Experimental Testbeds).

Section III. Detailed Proposal Information

A. Statement of Work

We propose to conduct a medium-scale experimental and prototyping effort centered on techniques for coordination and learning for survivability. Our effort will be conducted at two locations: the Distributed Artificial Intelligence Laboratory at University of Massachusetts Department of Computer Science (the primary contractor) and the other at the University of Delaware Department of Computer and Information Sciences (a subcontractor).

Year 1:

- *Extend existing simulation testbed to model survivability issues and complex resource constraints.* This task involves extending the existing TÆMS testbed to simulate models of complex resource constraints and survivability issues such as failures and intrusions. Currently, the simulator focuses solely on computational resources. *Exit criteria:* delivery of the extended testbed for use in development and evaluation of the other technologies (coordination, learning, detection, and diagnosis). Also, demonstration of sample scenarios involving system attacks or system resource failures.

- *Extend distributed coordination approach to support both procedural and declarative mechanisms.* The existing approach, GPGP (Generalized Partial Global Planning) allows for the combination of several different coordination mechanisms in response to different environmental characteristics. We will extend this approach to begin to develop new classes of mechanisms specifically aimed at survivable systems. Some of these mechanisms will be general, but we also want the agents to be able to learn situation-specific coordination mechanisms. To do this, we will have to alter the approach somewhat to allow the use of *declarative* mechanisms—simply put, mechanisms that can be created or altered on the fly. *Exit criteria:* demonstrate declarative mechanisms for using declarative organizational knowledge. This task will partially be carried out by the UD subcontractor.

- *Develop representations for learning, detection, and diagnosis.* The representation of univariate models must extend easily to multivariate models, and the multivariate models must support causal conjectures useful for diagnosis. Likely candidates include rule-based and graphical models. *Exit criteria:* specification document and prototype implementation.

- *Begin to develop algorithms for univariate and multivariate learning.* We will experiment with univariate models in the form of both empirically and analytically-derived distributions. For multivariate learning, we will initially concentrate on fixed feature sets and later extend the methods to select their own feature sets.

- *Begin to develop algorithms for detection and diagnosis of behavior during normal operation (e.g., use of shared resources).* This task will initially use hand-constructed models to characterize the expected behavior of agents, methods, resources, and tasks. These models will be used to detect statistically significant deviations from that behavior. We will use learned models when the diagnosis and learning components are integrated.

Year 2:

- *Extend distributed coordination algorithms for task sharing, self-interested agents and organizational design.* This involves the development and testing of new coordination mechanisms for transferring tasks between agents when appropriate, building and interacting with agents that maximize their own utility rather than their view of the system's utility, and for creating appropriate organizational structures (such as matchmaking or brokering services for finding out other agent's capabilities [10]). *Exit criteria:* Demonstration of these capabilities and technical papers describing and analyzing them. This task will partially be carried out by the UD subcontractor.

- *Complete development of algorithms for univariate and multivariate learning. Exit criteria:* demonstration of task characteristic learning using simulated data, technical papers describing and analyzing the new algorithms.

- *Introduce controlled variation and exploration into coordination algorithms to learn characteristics of rarely-used methods.* One well-known problem with learning based on current actions is that an agent continue taking actions it believes are optimal, rather than exploring new actions whose effects may be superior. We will provide agents with the ability to statistically estimate the relative value of exploration vs. exploitation.

- *Develop algorithms for detection and diagnosis of behavior during abnormal operation. Exit criteria:* demonstration of successful intrusion detection is several different scenarios, technical papers describing and analyzing the new algorithms.

- *Produce integrated prototype shell with coordination, learning, detection, and diagnosis.* This task requires the production of a standalone agent shell that integrates the technologies that have been developed so far. Wrapping legacy code is not a requirement at this point. *Exit criteria:* initial deployment of integrated prototype.

- *Begin evaluation of the initial integrated prototype.*

- *Extend and improve algorithms and prototypes.*

Year 3:

- *Enhance simulation testbed.* This task requires that the testbed be altered to provide additional realism and allow a greater range of agents, tasks, resources, and intrusions. The directions of this effort will be learned from other program participants, who will have reported some of their results by this time. Detailed documentation should be completed. *Exit criteria:* Delivery of the final testbed and documentation.

- *Complete extension and evaluation of distributed coordination algorithms for organizational design. Exit criteria:* Deliver final algorithms, reference implementations, and analyses. This task will partially be carried out by the UD subcontractor.

- *Complete extension and evaluation of learning, detection, and diagnosis algorithms for organizational design. Exit criteria:* Deliver final algorithms, reference implementations, and analyses.

- *Create interface specifications for legacy subsystems.* Exit criteria: demonstration of legacy wrappers for a few common situations: a non-interruptible basic tool and a program with a few controllable parameters.
- *Extend integrated prototype with final coordination, learning, detection, and diagnosis algorithms.* Leads to the final task.
- *Evaluate and document integrated prototype. Exit criteria:* Deliver final complete integrated prototype and all documentation.

B. Expected Results

The expected results of our research includes: 1) a prototype of an integrated architecture, 2) algorithms and prototypes for coordination, learning, and detection and diagnosis; 3) a testbed for modeling and simulation; and 4) extensive evaluation.

Integrated Architecture Prototype

Figure 1 shows a diagram of the agent architecture that we will produce. It is based on an architecture that we have previously developed for studying agent coordination in the GPGP framework. The architecture is divided into two sections. The upper section contains components that agents possess in order to successfully execute their short-term coordination behaviors. The lower section contains the additional components that agents will need to alter their long-term behavior. These latter components are involved in organizational design.

The components in the upper section of the figure are part of the typical agent architectures we already build. The most important part is the problem solving/task assessor components, which is where a legacy system would reside, or the code that does domain work in a newly coded agent. The problem-solving component could be an existing program or database. The set of tasks that the problem-solver can potentially accomplish are generated either from external requests (perhaps from other agents) or by the legacy system inside the problem solver itself. The components that surround the problem-solver help make decisions about what particular tasks the problem-solver should actually do, what methods should be used, in what order activities should be attempted, and precisely when these actions should occur. It also makes decisions about to whom to transmit problem solving outputs and when to request that another agent do tasks that cannot be done locally.

The components in the lower section of the figure — the organizational design agent components (organizational designer, learning, and detection and diagnosis module) and their associated data structures — will be new additions to our agent architecture, and will be one of the major efforts of the proposed research (see below). The organizational designer component is responsible for providing the organizational knowledge to be used at the operational level given information about system goals and the available network resources. This component will also enable agents to negotiate to coordinate organizational design activities. The detection and diagnostic module provides information to the organizational designer about any changes in computational needs or network resources that are detected by the execution monitor.

We already have operational versions of the components in the upper section of figure 1. However, work will also have to be done to modify and extend some components (e.g., the design-to-resource scheduler, GPGP coordination module, task assessor) so that they can operate effectively with the new organizational design components. The coordination module will be extended to exploit organizational knowledge, reason about physical resources, consider fault-tolerance and security issues, negotiate using more complex protocols, perform load balancing, and work with rough commitments. The design-to-resource scheduler, which currently creates schedules taking into account, quality, cost, and duration of executable methods, will also be extended to be able to reason about physical resources, fault tolerance, and security issues. The task assessor will have to be modified to use organizational knowledge to restrict the task alternatives available to the scheduler.

Figure 1: Diagram representing the agent architecture for our coordinating agents.

We will provide prototypes, specifications, and analytical models for the existing, extended, and new components of our agent architecture. We will provide a prototype that demonstrates how these components of the architecture can be integrated and that allows for empirical evaluation. However, we will also provide an abstract specification of the components and their interactions. This specification will indicate which features of our prototype are incidental and which features are essential. Finally, we will produce an analytical model to support the abstract specification and provide a means to reason about the effect of specification changes.

The system architecture will be designed to incorporate technologies designed by other contractors on this and future projects. Other approaches to detecting, diagnosing, and responding to attacks will be able to provide information to agents and thus affect the organizational design. Variability and diversity in components will be exploited by the architecture in a variety of ways, including how agents select methods and resources and how agents learn about the characteristics of those subsystems and components. In this way, our work will provide a platform for exploiting technologies beyond the prototypes we develop.

In addition to the system architecture, we will define wrappers for legacy systems at multiple levels of functionality. At one extreme, some legacy systems can only serve as a single method executable by an agent. For example, a legacy system might return the results of a single calculation or type of database search. At the other extreme, legacy systems with multiple, interacting functions could be organized into one or more TÆMS task hierarchies and deeply integrated into our agent architecture. In this latter case, legacy systems would differ little from methods specifically designed for use within a survivable system. We will specify both the high and low ends, as well as several intermediate levels, and detail the requirements, costs, and benefits of each level of integration.

Algorithms and Reference Implementations

We will provide algorithms and reference implementations for the three technologies we deem essential for survivability — coordination, learning, and detection and diagnosis. These technologies will be integrated into our agent architecture (above), but are also useful individually. We will deliver algorithm specifications and reference implementations. As we will do with the integrated architecture, we will also produce analytical models of the behavior of the algorithms. These analytical models will support the algorithm specifications and allow inferences about the effect of altering the algorithms' basic design.

Specifically, we will develop:

- *Coordination algorithms* — Techniques that support both procedural and declarative mechanisms, task sharing, self-interest, and organizational design. Approaches to exploring and exploiting variability and redundancy in available methods and agents.
- *Learning algorithms* — Techniques for statistically modeling the value of individual parameters (univariate learning) and for explicitly modeling the joint distributions of several parameters/characteristics (multivariate learning).
- *Detection and diagnosis algorithms* — Detection techniques that use learned models and that use analytic techniques for checking output. Diagnosis techniques that use both existing TÆMS task structures and learned models to pinpoint the cause of deviations from normal behavior.

Each of these technologies are both independently useful for survivability and mutually supportive. Each technology can be used individually to enhance survivability. For example, effective coordination does not require learning and diagnosis, nor does accurate learning require coordination. However, each technology supports the others. For example, accurate learning will greatly aid the coordination algorithms. These characteristics reduce the risks of pursuing all three categories simultaneously (because progress in any one category will be beneficial), but greatly increase the potential payoff if research in more than one category is successful.

Testbed for Modeling and Simulation

We will extend our existing testbed to encompass the new technologies developed under this proposal and the new challenges posed by survivability. Specifically, the enhanced simulator will include facilities for:

- *Compromised agents, methods, and resources* — To properly simulate intrusions, the simulator will allow random and non-random patterns of degradation and failure for individual agents, methods used by one or more agents, and shared resources (e.g., communication links).
- *Additional resource types* — Our existing simulator contains only one type of resource — a method that can be used by one or more agents. Actual systems use a variety of other types of resources (shared communication links, physical devices, and user attention). These must be appropriately simulated in order to evaluate survivability.

We will coordinate with other contractors to insure that our simulations are faithful. This includes the types and characteristics of tasks, the constraints of shared resources, and the challenges posed by particular types of intrusions. In the latter case, it will be particularly important to model a wide range of potential intrusions. Similarly, we will coordinate with other contractors regarding the characteristics of legacy systems. These will include their complexity, the ability to model their characteristics, and the degree to which their functions and control parameters can be represented as TÆMS task hierarchies.

Evaluation

We will evaluate both individual components and integrated systems. The individual components will be evaluated first, because this limits the complexity of evaluation during the initial phases of evaluation and redesign. Later, however, the integrated system will be evaluated as a whole in order to evaluate overall performance and important interactions among system components.

We will conduct several types of studies to evaluate the characteristics of various system components, including:

- *Lesion studies* Remove a component of interest from a larger system, or substitute a less capable component, to observe the effect on performance.
- *Multi-factor studies* Systematically vary two or more aspects of a system simultaneously in order to discover important interactions among those aspects.

C. Detailed Technical Rationale

Organizational design is a difficult task because any non-trivial system can be organized in an extremely large number of ways. Any effective approach to organizational design must therefore be *knowledge-based*. It must direct the search for high-performance organizations using knowledge of the relative capabilities of agents, performance of methods, interactions among tasks, and characteristics of shared resources.

Organizational design in the context of survivability faces additional challenges. First, design must be decentralized. Centralized systems are both vulnerable and brittle — they are relatively easy to attack and fail catastrophically when attacks are successful. Rather than being centralized, design capabilities should be distributed throughout the system. Second, the relevant knowledge for organizational design is not necessarily stable. The capabilities of agents, performance of methods, interactions among tasks, and characteristics of shared resources can be altered by intrusions or fluctuating demands. Third, organizational redesign must be rapid and effective. Slow response gives attackers an unnecessary advantage.

These challenges indicate the importance of the key technologies we have identified. *Distributed coordination* ensures that the capabilities for organizational design are spread throughout the system and not easily disabled. *Learning* ensures that the knowledge necessary for effective design is constantly updated. *Diagnosis* provides rapid organizational design by using learned knowledge to limit the search space of potential designs.

Organizational Design

An organizational design specifies roles for agents — limiting agents' available activities. This prunes the problem-solving search space of agents based on the characteristics of the likely tasks, resources, and available methods. We refer to this imposed organizational design as the agents' "long-term" behavior.

Within a given organizational design, agents still coordinate with each other to decide which methods to execute to achieve tasks based on the dynamics of the current problem-solving situation. These more dynamic actions are the agents' "short-term" behavior. Short-term coordination behavior is especially important for survivability where tasks and available resources may change rapidly.

Note that the "long-term" behaviors imposed by an organizational design are not necessarily fixed. Agents may change the organizational structure of the system as the nature of the tasks or available resources changes. This requires detection and diagnosis to determine when the tasks or the environment have changed in a fundamental way (as opposed to small fluctuations).

In order for the organizational design capabilities of our approaches to supporting the computational needs of survivability, several basic issues regarding representation need to be addressed:

- *How to model the characteristics of the tasks* necessary for supporting the computational and informational needs of survivability. These characteristics include how frequently specific classes of tasks or sequences of tasks occur, which agents or classes of agents are likely to be responsible for certain tasks, and the relative importance and utilities of tasks.

- *How to model alternative methods* for achieving tasks so that the system can reason about the tradeoffs involved with each. Tradeoffs may involve varying levels of cost, quality,

duration, resource usage, and security risks. Tasks may be interrelated to the degree that how one task is executed may affect the possible alternatives available to do another task and the characteristics of the alternatives.

- *How to model computational and network resources*—both long term capabilities and short term state. These resources define the environment in which the system operates.

We believe that an extension of the TÆMS (Task Analysis, Environment Modeling, and Simulation) modeling framework which we have developed for short-term agent coordination will also be able serve the above mentioned representational needs for long-term coordination.

Agents

In order to support both long-term and short-term coordination behavior, agents must have an architecture that supports negotiation and coordinated effort among heterogeneous agents. We propose to use an agent architecture that has worked well with the GPGP approach to agent coordination [16, 21]. GPGP assumes that agents have:

- a belief database containing their current beliefs about the structure of tasks in the current problem-solving episode.
- a domain-dependent way of detecting the presence of coordination relationships (sub-problem interactions) between local tasks and those of other agents.
- a local scheduler that decides what method execution actions take place and when. The local scheduler can take into account constraints placed on local behavior by coordination relationships between local actions and non-local tasks.

The agent-based framework of TÆMS is quite general. We usually experiment with agents that are originally designed within TÆMS, but such explicit design is not required. Instead, agents can consist of a legacy element with a wrapper that provides the belief database, local scheduler, and coordination modules. In principle, the core functions of an agent could be provided by any system element whose performance characteristics can be modeled (a requirement for making reliable commitments with other agents).

Distributed Coordination

While agents are part of an organizational design, they still may make decisions regarding more dynamic coordination with other agents. An organizational design only prunes the possible problem-solving search space but does not specify exactly how a problem is to be solved. From our experience, we feel that permitting agents some level of flexibility in activities is critical to the adaptability of the system with regard to the current situation. Without this level of flexibility, organizational redesign (which is computationally expensive) would have to be performed much more frequently.

Extensive work has been done in the area of coordination in multi-agent systems. The next section provides an overview of our existing work in domain independent coordination

mechanisms for agents in multi-agent systems. This work will be extended as part of the proposed research.

GPGP (Generalized Partial Global Planning) [16, 21, 13] is a extendible family of coordination mechanisms. It specifies an agent's coordination behavior by telling how and when to communicate and construct non-local views of the problem-solving situation, how and when to communicate the partial results of problem solving, and how and when to make and break *commitments* to other agents about when partial results will be made available.

The role of the coordination mechanisms is to provide information that allows the local scheduler to construct better schedules. This information can be in the form of modifications to portions of the subjective task structure of the episode or in the form of local and non-local commitments to tasks in the task structure.

If inconsistent constraints are introduced, the local scheduler will return at least one violated constraint in all its schedules. Since the local scheduler typically satisfices instead of optimizes (i.e., it does not search exhaustively), it may do this even if constraints are not inconsistent. The current set of GPGP coordination mechanisms are for cooperative teams of agents – they assume that agents do not intentionally lie and that agents believe what they are told. An important issue in our research for survivable systems will be how the coordination mechanisms can handle the situation in which other agents will intentionally or unintentionally give false information about their activities. A possible approach to this problem that we will explore is to introduce more self-interested coordination modes into GPGP. This will be based in part on our own work on negotiation among self-interested agents [51]. In previous work, GPGP and TÆMS have been used to investigate various agent organizational structures or "coordination modes" [48]. In this work, the idea of "rough commitments" was used to model organizational structure in a distributed data processing context. Whereas the commitments discussed above relate to short-term coordination behavior between agents, rough commitments relate to the long-term behavior of agents. For example, a rough commitment between two agents may say that partial results will be communicated from one to the other on a specified periodic basis, and this commitment will be in effect as long as the problem-solving environment does not change drastically.

The advantage of rough commitments is that the two agents involved do not have to negotiate about when a partial result is to be communicated each time a result is available. Rough commitments take advantage of a regularity in the task environment to reduce the overhead of unnecessary negotiation. Extensions to the rough commitment concept will be part of the work to add organizational design considerations into the GPGP coordination module.

To summarize, distributed coordination using GPGP can provide substantial survival advantages for large scale information systems. Coordination provides the underlying mechanism to flexibly organize computational resources. Coordination can also help manage variability and redundancy of system elements – two other key approaches to survivability. Redundancy and heterogeneity pose serious challenges to maintaining efficient operation of an information system. If left uncoordinated, several redundant elements may all attempt to provide a particular result, introducing unnecessary computational burden. Similarly, poor coordination might make use of suboptimal combinations of heterogeneous elements. Effective coordination can allocate tasks to the agents best suited to particular tasks. Finally, a multi-agent, multi-coordination mechanism approach allows us to easily integrate system components designed

by others (such as sophisticated intrusion detection monitors) and model their inclusion as alternative organizational structure choices.

Learning

The TÆMS framework specifies parameters that characterize individual agents, tasks, resources, and methods. For example, a method can be characterized by its duration (the amount of time required to execute it), its quality (a scalar value or vector indicating desirability, e.g., the precision, belief, or completeness of its result), and its resource usage patterns. These parameters may be point estimates, distributions, or functions that depend on one or more inputs. For example, quality may be a function of the timing and choice of an agent's actions ('local effects') and possibly other previous or future task executions ('nonlocal effects').

Coordination algorithms such as those in the GPGP family depend on accurate parameter estimates to characterize agents, tasks, resources, and methods. If estimates of these parameters are inaccurate, agents may select inappropriate methods, they may depend on results from other agents that are unreliable, they may inappropriately commit to accomplishing tasks by certain deadlines, and they may saturate resources.

In the context of survivability, static parameter estimates will quickly become obsolete. The performance of some agents will degrade because of direct attack, because they take on new tasks shifted from other agents, and because they depend on important resources (e.g., network bandwidth, storage, etc.) whose availability will change.

As a result, agents designed for survivability need dynamic parameter estimates. Such dynamic estimates can be produced in one of two ways. First, they can be estimated directly by what we call *univariate learning*. This approach makes estimates directly from measurements of the parameter in question. For example, the duration of a method can be estimated by executing the method several times and using the measured duration of the method to estimate a distribution.

Accurate learning of parameter values has obvious advantages. As parameters change, the models will track those changes, resulting in better coordination than methods using static parameter values. It will also provide the means to detect important changes in agents, resources, and tasks. By statistically testing new values against the existing distributions, we will effectively balance the relative probabilities of false positives (detecting a change when none exists) and false negatives (missing a true change). This use of parametric models is covered in more detail below.

Learning parameter values is important, but such learning is limited in scope. If a situation changes, then a new parameter model will need to be learned. An alternative is to learn *multivariate models* that can characterize different contexts, and then learn parameter models for each context. For example, consider a communications link where traffic is affected by an air base's alert status. A simple parameter model would have to relearn the traffic load on the communications link each time the alert status changed. In contrast, a more complex model could represent the relationship between alert status and load. Once such a model was learned, the model's prediction of load would change as soon as the base's alert status changed. Our learning algorithms will develop such models based on operating experience.

In many cases, it may be important to produce more than just point estimates of important parameters. For example, the mean duration of a method is little help in predicting the

likelihood that a new run of the method will finish within an allotted time. Instead, an estimate of the distribution of duration is needed. In the case of some parameters, distributions of low-level parameters (e.g., the duration of a subtask's method) can be combined to produce distributions for higher-level parameters (e.g., the duration of an entire task).

Detection and Diagnosis

One of the best ways to enhance survivability is to detect and diagnose intrusions by using learned models. These types of reasoning are similar to those used by doctors and public health officials in human societies. Univariate learning establishes baseline health figures (e.g., normal white-blood-cell counts or the frequency of emergency room admissions). These baselines are used to detect abnormalities that indicate the early stages of disease in a patient or the outbreak of new diseases in a population. Multivariate learning produces models that help diagnose diseases (e.g., fever is nearly always a sign of infection). In the same ways that doctors and public health services develop and use medical knowledge, computational organizations can learn and reason with models for survivability.

Inferences based on these models need not be perfect in order to be useful. First, such inferences can constrain the search space of potential organizational designs that agents have to explore in more detail. This increases the efficiency and effectiveness of search. Second, such inferences can inform the efforts of human managers to detect and contain intrusions. Providing specific, focused evidence about a potential intrusion allows human managers to respond more quickly and effectively.

Example. An example of the type of diagnostic reasoning that we propose to implement in a domain-independent way is exemplified by our work with the LODES system [55]. The LODES network diagnosis system observes message traffic on the network in order to detect and analyze situations which indicate a hardware or software problem in the network. The LODES system is a multi-agent system in which there is a LODES diagnosis agent on each network segment.

Each agent is responsible for monitoring traffic on its segment and diagnosing any problems that are recognized. LODES agents start out with the assumption that there is no need to explicitly coordinate with other LODES agents on different segments of the network. The only interaction among agents is an occasional request for a diagnostic task that can be performed by only the LODES agent that is located on a specific network segment, and the sharing of information about the network configuration and the final results of diagnosis. The basic assumption behind this lack of coordination is that there are sufficient communication and computational resources available on the network to sustain a certain level of non-coherent behavior (e.g., two agents diagnosing the same problem). This assumption is appropriate for diagnostic activities in most network environments and was the basis for how the system was originally implemented. However, it is not a valid assumption, for example, in network environments where diagnostic activities generate substantial message traffic, and where there are multiple agents performing redundant diagnosis and the cost of communication is significant.

Consider the following situation in the network environment as shown in Figure 2, where L1 to L7 are LODES agents, Net1 to Net7 are network segments, and Net5 and Net6 are connected with a narrow-bandwidth line. Suppose a host, HA on Net1, sends a broadcast to all hosts on Net7, but most of the hosts cannot understand that protocol. Upon receiving the

broadcast, the hosts on Net7 that cannot understand the protocol simultaneously send back error packets in order to inform HA that they have discarded its broadcast packet (this use of an inappropriate broadcast by HA is called the first problem). In this situation, agents L1 ... L7 will be concurrently diagnosing this problem since the return route of these error packets will traverse network segments monitored by each of these agents.

In diagnosing the cause of these error packets, each LODES agent may independently send diagnostic test packets through the network to assess which one of a variety of causes is actually responsible for the error messages. This redundant diagnostic message traffic may in turn lead to another problematic situation if the network environment has network segments which are implemented as narrow-bandwidth lines, or if a tariff is associated with each message (e.g., in this case, the narrow band link between Net5 and net 6 gets congested). This is a problematic situation since it indicates the potential overloading of an expensive or scarce resource. This secondary problem, which is directly caused by LODES agents' activities, can be attributed to the lack of effective coordination among agents.

Figure 2: Network Environment for the Example Problem

One of the interesting aspects of our diagnosis application is that, in this example problem, the LODES agents, in their monitoring of the network, will detect the secondary problem. In this case, agents L5 and L6 will decide that the secondary problem is not tolerable because they know the existence of the narrow-bandwidth line. The existence of this problem and the fact that it was caused by the LODES agents themselves will be the trigger for the learning component to be invoked. Other mechanisms for invoking the learning component are possible.

In order to detect such problems, LODES agents record a trace of their inference processes and communication actions. In this example problem, a detection component identifies the task(s) in which the test packets were sent by analyzing the trace of the first diagnostic problem solving activity. This identifies that the observed test packets were sent as the result of an execution method named "Get-RTT-between-Both-Ends".

Next, agents create a comprehensive view about the situation in which the plan "Get-RTT-between-Both-Ends" was selected. Agents select this method to estimate the network load and bandwidth by gathering statistics on the round-trip time (RTT) of a number of test packets sent into the network. This method, which is not the optimal way to gather this statistic, produces only an approximate value for RTT. A better plan would have been to use the Simple Network Management Protocol (SNMP), which is designed for acquiring network management data, but this was not possible in the example since the adjacent network routers did not implement

it. An agent, in order to understand what other tasks agents are executing, must know that local routers do not use the SNMP, and that all agents along the route from Net 7 to Net 1 will be performing the same diagnoses. Agents can then understand that the identical task was selected in other agents. This view of which tasks were to be executed in other agents was not present and thus responsible for the error; this is one of the important parts of the enhanced subjective view necessary for better coordination (see Figure 3 (a) and (b)).

Figure 3: Views of LODES Problem Solving.

Finally, if agents have the control rule that can derive appropriate coordinated actions from the comprehensive view, information about the resource type and usage by other agents, and to acquire the fact that the identical and sharable task is scheduled in all agents are marked as the mainstream. This important part of the comprehensive view is described in Figure 3. The tasks for requesting these information are put into the action-part of the new control rules that are then added to each agent.

In this way, the agents have adapted their coordination strategy to respond to unanticipated characteristics of the environment. We feel that this is exactly the type of diagnostic reasoning and organizational adaptation that is needed for survivable systems. We are proposing to do this reasoning and adaptation in a domain-independent way using the extended TÆMS representation. The extended representation includes the resource consumption attributes of activities, a description of network resources, and the declarative representation of coordination policies that we propose to develop for GPGP.

D. Detailed Technical Approach

Much of our approach concerns how we will develop key technologies coordination, learning, and detection and diagnosis. Below, we describe specific directions for our work in each of these technologies, as well as important challenges that we will address. The development of these technologies will all be done in the context of the agent architecture outlined in Figure 1 on page 14.

Representation: TÆMS

TÆMS represents agent activity in terms of task structures at multiple levels of abstraction, each with a set of goal criteria. The goal of the agent or agents is to maximize the value of the evaluation function specific to each task group. A task group consists of a set of tasks related to one another by a subtask relationship that forms an acyclic graph. Figure 4 shows an example task structure from the information gathering domain. Tasks at the leaves of the tree represent executable methods, which are the actual instantiated computations or actions the agent will execute (in the figure, methods are shown as boxes). Each method has quality, cost, and duration probability distributions associated with it that statistically model the outcome of executing the method. Note that the methods can be "anytime" methods that produce better results as they are allowed more computation time. The circles higher up in the tree represent various subtasks involved in the task group, and indicate precisely how quality will accrue depending on what methods are executed and when. The arrows between tasks and/or methods indicate other task interrelationships, in this case quantitative in character, where the execution of some method will have a positive or negative effect on the quality or duration (or other resource requirements) of another method. These effects of task interrelationship are called *non-local effects* (NLEs).

Figure 4: Example TÆMS task structure from the information gathering domain.

Much of the work on TÆMS proposed here involves extending the representation to multiple resource types. Physical resources, be they people, equipment, or computational resources such as files, are very important to model in many domains [59, 3, 49, 53, 56, 43]. From the point of view of task environment specification, the only effect of resources, with

respect to computational tasks, is to change the duration or quality of other tasks, and so we represent this as a set of non-local effects (NLE's). Because the NLEs are linked (i.e., not just between two nodes) we add a *resource* node to the task structure and associate any state information needed with it. From the point of view of the discrete, state-based mathematics behind TÆMS, a *resource* looks like a task—but instead of a quality vector, a nominal state vector is used, and there is no associated duration or subtask relationships. Instead, methods are related to the resource via one kind of NLE (e.g. uses, replenishes, consumes, reads, writes), and another NLE runs from the resource to the methods (e.g. exclusiveaccess, limitedbandwidth, consumable). The method-to-resource NLE's change the state of the resource, and the resource-to-method NLE's affect duration and max quality as usual.

We also want to use TÆMS to represent an organizational structure. This includes representing the frequency with which certain tasks occur, and at which agents. We need to represent capabilities (long term commitments) of the agents, and about what the agents need to communicate for particular tasks (see for example [12]).

Coordination: GPGP

Figure 5: An Overview of Generalized Partial Global Planning

This section will provide a brief overview of the GPGP approach. Figure 5 shows a simple two-agent example that we will use. Each agent has as part of its architecture a belief database, local scheduler, and coordination module. The local scheduler uses the information in the belief database to schedule method execution actions for the agent in an attempt to maximize its performance. We add to this a coordination module that is in charge of communication actions, information gathering actions, and making and breaking commitments to complete tasks in the

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

Here is a short example intended only to give the reader a feel for the overall approach. In Figure 5, both agents have executed an initial information gathering action, and have their initial views of the task structure-everything in the agents' belief database except for the shaded tasks (Tasks 2, 5, D and E), and the relationships touching the shaded tasks. One of the coordination mechanisms (Mech. 1, update non-local views) performs an information gathering action to determine which tasks may be related to tasks at other agents ("detect coordination relationships"). These tasks are then exchanged between the agents, resulting in the belief databases shown in the figure (including the shaded tasks). Other mechanisms react to the task structure. One mechanism (Mech. 5, handle soft predecessors) notices that Task 2 at Agent Y facilitates Task 5 at Agent X. In order that Agent X might schedule to take advantage of this, Agent Y's mechanism makes a local intermediate deadline commitment to complete its Task 2 by time 7 with minimum quality 45 (an omniscient observer may infer that Y intends to execute Method B, but that local information is not a part of the commitment). A commitment is made in two stages: first it is made locally to see if it is possible as far as the agent's local scheduler is concerned, and then it is made non-locally and communicated to the other agents that are involved. Note that the deadline on the non-local version of this commitment is later (time 8) to take into account the communication delay (here, 1 time unit). Similarly, Agent X has a mechanism (Mech. 3, handle simple redundancy) that notices that either agent X or Y could do Task 4. Agent X does eventually commit to this task and communicates this commitment to Agent Y.

In both cases the agents' local schedulers use the information about the task structure they have in their belief database, and the local and non-local commitments, to construct schedules. The local scheduler may return multiple schedules which permits the coordination module to make the decision about which is the best schedule based on criteria not available to the local scheduler. Each schedule is evaluated along the dimensions of the performance criteria (such as total final quality and termination time) and for what (if any) local commitments are violated. If a commitment is violated, the local scheduler may suggest an alternative (for instance, relaxing a quality or intermediate deadline constraint). The coordination module chooses a schedule from this set, and handles the retraction of any violated commitments.

The role of the coordination mechanisms is to provide information to the local scheduler that allows the local scheduler to construct better schedules. This information can be in the form of modifications to portions of the subjective task structure of the episode or in the form of local and non-local commitments to tasks in the task structure. The five mechanisms that are described in the example form a basic set that provides similar functionality to the original partial global planning algorithm as shown in [26]. Mechanism 1 exchanges useful private views of task structures; Mechanism 2 communicates results; Mechanism 3 handles redundant methods; Mechanisms 4 and 5 handle hard and soft coordination relationships. We have already built more mechanisms, including ones to update utilities across agents, or to balance the load better between agents. The mechanisms are independent in the sense that they can

be used in any combination. If inconsistent constraints are introduced, the local scheduler will return at least one violated constraint in all its schedules. Since the local scheduler typically satisfices instead of optimizes (i.e., it does not search exhaustively), it may do this even if constraints are not inconsistent. The current set of GPGP coordination mechanisms are for cooperative teams of agents-they assume that agents do not intentionally lie and that agents believe what they are told. A formal definition of this framework is specified in [13].

In this work we propose to add several new mechanisms, including hierarchical task decomposition, (both top-down and bottom-up [28], self-interest, and mechanisms that deal with limited resource availability. Mechanism choice will be expanded—mechanisms are not chosen on just a system-wide, or even agent-by-agent basis, but on a task-by-task basis within an agent. Mechanisms will be designed to exploit the organizational knowledge about which agent to interact with in order to achieve certain goals. Organizations can also change which agent does scheduling for what other agents. Finally, we need to add the ability to create and interpret declarative mechanisms on the fly, so that we can dynamically learn new situation-specific coordination algorithms.

Learning

Coordination algorithms provide a system with the ability to respond to changing circumstances, but it cannot allow the system to *change how it responds* to particular situations or to develop novel approaches for dealing with previously unknown situations. Because a system designed for survivability should be able to change how it responds, we propose to provide algorithms for learning two kinds of models: univariate models (models that describe single *parameters*) and multivariate models (models of *relationships* among several variables).

Univariate Learning As noted earlier, distributed coordination requires accurate estimates of many parameters. Our approach will model these parameters statistically, using both empirical and theoretical distributions. An empirical distribution uses a set of actual data values as a model. Computer-intensive statistical techniques [44, 27] can be used to determine whether newer and older values are drawn from the same distribution. When the number of data points is small, such a modeling approach can be far more accurate than using a theoretical distribution. As more data points accumulate, we will model parameters with theoretical distributions, because they provide faster statistical computations.

Multivariate Learning Our approach to learning multivariate relationships is a two-step process. First, the agent must determine the set of variables, or *features*, that will be examined for potential relationships. For example, an agent attempting to model load on a communications link might select features concerning its own behavior (e.g., types of tasks, selected methods), the behavior of other agents, characteristics of resources (e.g., load on other links), and characteristics of tasks (e.g., time and quality constraints). Because the number of possible features can be extremely large, agents will generally select a subset of all possible features. Where possible, feature selection will be knowledge-based — using the agent's own knowledge and that of other agents.

The second step of learning multivariate relationships is to search for useful models. These models will distinguish between contexts which should be characterized by distinct parameter

values. Searching for useful models has been the focus of substantial research in machine learning. We will use some recent developments in systematic search [45, 58] that allow efficient search of extremely large spaces of models.

Multivariate learning must be iterative. After an initial model is found, it is necessary to continue examining potential new features and to continue searching for useful models. Systems designed for survivability must continue to adapt, to improve their models in stable situations and to learn new models when those situations change.

Detection and Diagnosis

We will produce two general methods for reasoning with induced models: detection and diagnosis. *Detection* identifies when situations have changed and new learning is required. For example, an agent might learn a multivariate relationship between two given variables (the size of a database and the complexity of a query) and a third variable (the time required to search that database). However, this relationship might be learned during a time of relatively low load on the database server. As the server's load changes, it is important for the agent to detect that its predictions are no longer accurate, and to learning a more complex model, probably by including some measure of server load in its model.

Diagnosis pinpoints probable causes of changes in situations. For example, the database search mentioned above might be one part of a much more complex task. If the time to accomplish the overall task increases significantly, it is important to pinpoint the cause of this increase, both to focus learning and to identify potentially compromised portions of the system. Diagnosis is often quite difficult. Distributed systems can have complex task structures where many agents contribute to a particular result. Those agents, in turn, may make use of several methods and shared resources to accomplish their tasks.

Both detection and diagnosis make use of models. These models will generally be induced, but they can also be provided *a priori* by human operators. This latter approach may be particularly useful before the system has accumulated substantial operating experience, but all models, regardless of their origin, will be altered and elaborated by the learning algorithms.

For the purposes of detection, the models define expectations, what statisticians refer to as a *null hypothesis*. Significant deviations from those expectations indicate that a situation has changed and that additional learning is required. For the purposes of diagnosis, the models provide knowledge about which variables are related and in what ways. This knowledge may not always be complete, but it will constrain the search space for learning, allowing more rapid and accurate learning than would otherwise be possible.

Our previous work on a model-based detection and diagnosis in the LODES system (see page 21) demonstrates this technology in domain-dependent context. We will generalize this work to be domain-independent and integrate detection and diagnosis with learning so that learned models, as well as TÆMS task structures and *a priori* knowledge, can be exploited to identify and pinpoint intrusions.

Detection We will pursue two complementary approaches to detection: consistency checking and statistical hypothesis testing. Consistency checking verifies that results fall within theoretically justified boundaries. For example, records returned from a database search should match the original query. Consistency checking can also involve the use of predetermined test

suites or the cross-checking of results from multiple agents, each of which should return the same result.

Statistical hypothesis testing can be applied to any parameter value that should remain constant over time or that should change in a known manner. For example, the duration or quality of a method should remain constant, or should be related to a finite number of other features (e.g., the agent's current processing load). Based on univariate or multivariate models, we will develop appropriate statistical test for agents to determine whether parameter values have significantly changed.

The methods have complementary strengths. Consistency checking is a powerful method, but it is often not possible to theoretically determine the correctness of a result. In addition, some parameters are inherently statistical, requiring the machinery of statistical hypothesis testing to detect significant changes.

Our prior work on empirically constructing statistical reference distributions [35, 34] and on adjusting analytic statistical distributions [36] have explored approaches to testing for significant deviations in learned models. In these cases, the intent was strictly to determine whether a learned model should be extended, but the inferences could easily be applied to enhancing survivability.

Diagnosis We will also pursue two complementary approaches to diagnosis: analyzing the task structure and analyzing induced models. The TÆMS task structure provides a way to trace the pedigree of a result and decompose it into constituent components. By analyzing these components it is possible to determine which agents and methods are responsible for changes in behavior. For example, an information gathering task might involve several subtasks: selecting which information servers to search, searching those servers, and aggregating the results. If the overall results are poorly suited to the original query, any one of the subtasks could be responsible. The TÆMS task structure explicitly represents these subtasks and supports reasoning about their interdependence.

Learned models can also support diagnosis. Multivariate models indicate which variables are related and in what ways. This information only indicates correlation, and not necessarily causation. However, the relationships expressed in the models can be used to greatly constrain the search space of possible features and relationships, and thus speed learning.

E. Previous Accomplishments and Work

Previous Work

We have been working on the problem of distributed coordination of intelligent agents for over twenty years and are recognized as the pre-eminent group in the world in this area. Our work, which has been highly experimental in character, has involved building complex applications to evaluate the effectiveness of alternative distributed control strategies and discovering new computational phenomena. Most notably has been our work on the distributed vehicle monitoring testbed (DVMT) for studying distributed interpretation systems [39] and its follow-on work involving DRESUN [1, 2]. We have also built applications in the area of distributed scheduling of manufacturing work cells [43], cooperating expert systems for crisis management [41], distributed planning in the context of Multi-fireboss Phoenix [42], cooperating experts for engineering design [38], and a distributed constraint satisfaction system for resource allocation in a long-haul communications network [7]. We have also recently extended our research to issues involved in coordinating activities of self-interested agents for use in the electronic commerce[52]. Out of these systems we have gained a deep understanding of issues and techniques necessary for achieving effective real-time coordination. This understanding is reflected in our recent developments of the TÆMS modeling [19] analysis and simulation framework that allows us to explore coordination issues without having to resort to complex applications that are not only time consuming to build but also allow only a limited range of experimentation, and the GPGP family of generic real-time coordination algorithms [16, 21] which will be one of the bases for proposed work.

Recently, we have also been building real standalone agent that operate on the Internet, and that implement some of these ideas, in a system called WARREN for multi-agent financial portfolio management. Many agents work together in a dynamically organized team in order to efficiently and robustly retrieve changing stock prices, news, and fundamental data from various locations on the Internet, and to analyze that data in various contexts in order to provide an up-to-date financial picture. Although built in the general TÆMS architectural style, WARREN contains only a simple scheduler and concentrates instead on inter-agent communication and case-based plan retrieval [57]. The WARREN system is *open*, meaning agents can come and go at any time. A WARREN organization consists of a portfolio interface agent for each user, two task agents for fundamental stock analysis and price-news graphing, a news information agent for Dow-Jones and Clarinet news, several different stock ticker information agents, and two EDGAR information agents assigned to the SEC's electronic archives for quarterly and annual reports. Organizational information agents include a "matchmaker" or yellow-pages agent that helps an information requester find the appropriate information server in the dynamic, open system. This is accomplished using a descriptive capability advertisement language[11] One important requirement was robustness, so that when any WARREN agent leaves the system (or crashes) the remaining agents reorganize so as to carry on as effectively as possible. Our experimental evaluations have included studies of both cloning behaviors [9], and matchmaking vs. brokering behaviors [10]. More information and demos can be found on the WWW at `http://www.cs.cmu.edu/~softagents/warren/warren.html`.

We have also built multiagent systems that integrate coordination and learning. Our work on network monitoring [54] involved analyzing traces of inferences made during previous coordination episodes. The analysis identified situations where agents used inappropriate

coordination strategies. The agents then used the analysis to create situation-specific coordination plans that improved performance. Our work on concurrent engineering design [47] involved agents that learned preferred organizational roles based on the past results of past problem solving episodes. The adaptive system produced better results than its non-adaptive predecessor.

Our previous work on learning has focused on developing statistical significance tests for machine learning techniques. As noted above, machine learning algorithms often suffer from overfitting. Techniques such as randomization testing [35] and bonferroni adjustment [36] successfully avoid overfitting, allowing the development of models with far better performance than conventional techniques.

In addition, we have developed methods for learning sequences of events in computer networks [46, 37]. Accurately predicting these sequences eases the task of managing large networks. The rules can be used to filter the raw data stream usually seen by network managers. Rather than seeing all events, managers can now view composite events and more easily identify faults in the network.

Current and Pending Support

Victor R. Lesser:

Current Support:

- Principal Investigator, NSF, "Control Issues in Asynchronous Parallel Knowledge-Based AI Programs," $154,546, 6/1/94–5/31/97.
- Principal Investigator, Boeing Helicopter, "Manufacturing Automation and Design Engineering," $75,000, 8/1/96–7/31/97.
- Principal Investigator, ONR, "Distributed Real-Time Situation Assessment," $663,272, 7/1/95–6/30/98.
- Principal Investigator; NSF, "Towards an Architecture and Theory for Agent Coordination," $415,238, 9/15/95–8/31/98.
- Co-Principal Investigator with S. Zilberstein, NSF, "Intelligent Information Gathering Using Decision Models", $322,910, 8/1/96–7/31/99.
- Co-Principal Investigator, NSF–State/Industry, "SIUCRC on Intelligent Information Retrieval," $7,200,000, 9/1/92–8/31/00.

Pending Support:

- Principal Investigator (subcontractor; Carnegie Group, Inc. prime), DARPA, $75,000. Pending.
- Principal Investigator (subcontractor; Texas A&M prime), DoD AFOSR, MURI "Intelligent Agent Assisted Wireless Battlespace Communications and Networks,"$1,000,000, five years. Pending.

David Jensen:

Current Support:

- Research Scientist, 75% calendar: Evaluation Methods for Complex, Intelligent, Information-Driven Systems. Subcontract through Sterling Software, Inc./DARPA, F30602-95-C-0257.
- Research Scientist, 25% calendar: A Substrate and Tools for Experimental Testbeds. DARPA/Rome Laboratory Planning Initiative, F30602-95-1-0021.

Keith Decker:

Current Support:

- Independent contractor to University of Massachusetts, $10,000, 8/1/96–7/31/97.

Pending Support:

- Principal Investigator (subcontract; Boeing Helicopter and University of Maryland prime), DARPA, "High-Performance Integration Testbed for Enabling Knowledge Sharing Technologies," $145,524, three years. Pending.
- Principal Investigator, University of Delaware Research Foundation, "A Distributed, Environment-Centered Agent Framework," $23,200, one year. Pending.

F. Comparison With Other Ongoing Research

Designing computational organizations for survivability using agents that learn, reason, and respond has unique advantages. First, our approach avoids the need for large knowledge engineering efforts or extensive forecasting about likely modes of attack. GPGP's domain-independent approach to distributed coordination makes it applicable across a wide range of computational tasks, rather than only a few specialized tasks. GPGP's modular coordination mechanisms provide a ready-made framework for learning new control strategies. Our incorporation of distributed learning techniques will allow agents to flexibly respond to different types of attacks.

Coordination, Learning, Detection, and Diagnosis

Our work in *distributed coordination* is among the most advanced available. To our knowledge, it subsumes nearly every other major approach to distributed coordination. In the case where our approach does not subsume the work, it is often complementary. For example, one recent development in distributed coordination that our work does not subsume is the use of plan recognition techniques [32]. Plan recognition makes inferences about agents' plans based on observing their actions, rather than relying on direct communication among agents. This reduces interagent communication and makes it more difficult for agents to be deceptive. Plan recognition is essentially a learning technique — an agent models the internal state of another agent by observing actions — and it would complement our proposed learning and diagnosis techniques.

Other approaches to coordination, such as economic and ecological approaches, have produced a variety of interesting theoretical results. Many of these techniques could be introduced to GPGP via different coordination modules. However, it is unclear to us how useful these techniques would be because they make tremendously simplifying assumptions about agent activities, the interaction between agents, and interactions among agents and resources. In contrast, GPGP already includes a variety of realistic factors, and has served as the foundation for real applications [20, 22, 48].

Our previous work on *learning* is uniquely suited to enhancing survivability. Our initial work on domain-specific organizational design focuses on detecting, diagnosing, and remedying inappropriate coordination strategies [54]. To our knowledge, this is the most advanced work to date, implemented in a real multiagent system that shows sophisticated diagnosis and adaptation of coordination is possible. Our work on learning algorithms has focused on incorporating statistical approaches that improve accuracy and avoid overfitting, a flaw in many existing learning techniques [36, 6]. Overfitting introduces unnecessary structure into learned knowledge, making it poorly suited to use in reasoning for diagnosis and remediation.

Organizational Design

There is very little work on organization design for computational systems. Of the existing work, two primary lines of research are most prominent. First, Durfee and Montgomery [23] view organizational design as a distributed search process based on several dimensions of computational activity (e.g., what, who, when, why, and where). This represents the most sophisticated work that is directly appropriate to the concepts we discuss in this proposal, and

we will be studying this work closely for its use in our systems. The second body of work [29, 17, 18] primarily concerns load balancing in multiagent systems — creating organizations with the correct number of agents for each activity. We believe that these techniques for dynamic reorganization can easily be incorporated in the TÆMS/GPGP framework.

Integration of Other Technologies for Survivability

Underlying GPGP is a task representation that is uniquely suited to building survivable systems [19]. The framework, TÆMS (Task Analysis, Environmental Modeling, and Simulation), models problem-solving activities of an intelligent agent operating in environments where responses by specific deadlines may be required, where the information required for the optimal performance of a computational task may not be available, where the results of multiple agents' computations (for interdependent subproblems) may need to be aggregated in order to solve a high-level goal, and where an agent may be contributing concurrently to the solution of multiple goals.

Because the TÆMS/GPGP task structure is designed to explicitly represent alternative methods of accomplishing tasks, it naturally incorporates information that is vital for survivability. Existing components of GPGP already reason about constraints of time and quality, and they select the alternative best suited to achieve particular goals. In addition, GPGP is explicitly constructed for distributed coordination where each agent lacks complete information. In computer systems under attack, reasoning with such limited information will be vital.

TÆMS and GPGP can exploit multiple information sources and integrate multiple methods of accomplishing tasks. While we will provide some of the technologies for survivability, the general architecture we will construct can exploit technologies we do not create. For example, techniques for creating variability and diversity through alternative implementations can be directly used to provide alternative methods for GPGP agents. Likewise, detection and diagnosis components can not only be triggered by agents' local observations but also from decoy systems in the network that also detect and diagnose intrusions. Similarly, systems for computational immunology can be seen as searching for an appropriate organization to repel an intrusion. As such, they may provide alternative coordination modes for GPGP.

We feel that many ideas from biological immune systems, ecologies, and market economics can likewise be introduced into GPGP by changing the utility calculations used by an agent to make decision about how it interacts with other agents. TÆMS and GPGP provide a framework for this interaction, thus we feel that our approach can integrate with a variety of other approaches being developed to make systems more survivability

G. Facilities

University of Massachusetts

The Cooperative Distributed Problem Solving (CDPS) Laboratory at the University of Massachusetts, under the direction of Professor Victor R. Lesser, has been active in the research areas of Distributed AI and MultiAgent Systems for over 20 years. Numerous visitors from around the world have come to visit, some for extended periods of time. The laboratory currently consists of seven students and three post-doctoral research scientists. The research productivity of the laboratory has been very high with over 35 journal articles and highly refereed publications being produced over the last three years.

The laboratory has been responsible for the development of important DAI concepts such as Functionally/Accurate Cooperative Systems, DVMT (Distributed Vehicle Monitoring Testbed), Partial Global Planning, and Multistage Negotiation. A former student of the laboratory, Edmund Durfee, received a Presidential Young Investigators award for his work on Partial Global Planning. The laboratory is also known for its building of large experimental DAI and Multiagent systems.

The laboratory is supported by the Computer Science Computer Facility of the Computer Science Department at the University of Massachusetts, which maintains over 350 computer systems for departmental research. Nearly all of these are connected to the Computer Science ethernet network which also supports over 100 asynchronous ports on terminal servers. Computer maintenance services are also provided by the Computer Science Computer Facility. It is comprised of two components: maintenance and support of equipment and software and central service fees. Central service fees include maintenance and support of communication networks, mass storage, file backup services, and consulting services.

University of Delaware

The Department of Computer and Information Sciences operates a joint research laboratory with the Department of Electrical Engineering. The facilities serve a total of roughly 40 faculty and 200 graduate students. The Department of Computer and Information Sciences (CIS) currently operates 72 Sun desktop workstations, four Mac or PC workstations, and one SGI Indigo graphics workstation for graduate student and faculty research purposes. The Department of Electrical Engineering (EE) presently operates 37 Sun, twelve SGI, three DEC alpha, one Mac, and one HP desktop workstations, plus 28 Sun 3's configured as Xterminals for research use by graduate students and faculty. The University's research computing facilities include an 8-processor Cray Research J90, and an 8-processor SGI Power Challenge.

H. Key Personnel

Victor Lesser is a Professor of Computer Science and Director of the Distributed Problem Solving Laboratory, University of Massachusetts, Amherst. He was General Chairman of the First International Conference on Multi-Agent Systems (ICMAS) held in June of 1995. He has also served on numerous AAAI and IJCAI program committees, most recently being an area chair for Communication and Cooperation at AAAI-93. He is currently on the editorial boards for IEEE Expert, Group Decision and Negotiation, International Journal on Intelligent and Cooperative Information Systems. Lesser is a founding fellow of the American Association of Artificial Intelligence.

Over the past 20 years, Professor Lesser's group has been one of the leading research groups in distributed AI/multiagent research in the world. His group has done work in most of the major areas of DAI research, including cooperative and self-interested systems, protocols for negotiation, multiagent learning from both statistical and explanation-based perspectives, and design of computational organizations. In performing this research, a number of large testbeds have been developed to evaluate ideas. These testbeds have involved applications in distributed interpretation, distributed network diagnosis, distributed scheduling, planning and resource allocation, concurrent engineering, and lately cooperative information gathering on the Internet. To indicate the currency of the group's work, three papers on multiagent systems from the group appeared in the recent AAAI96 and five papers appeared in the leading multiagent conference (ICMAS96).

Professor Lesser will devote at least 1 month (summer) of his time to this project, and will be actively involved in supervision of students and staff throughout the year. He currently receives support from the Office of Naval Research, the National Science Foundation, and Boeing Helicopter. Funding is pending from Carnegie Group, Inc. and Texas A&M.

David Jensen is a Research Scientist in the Department of Computer Science at the University of Massachusetts, Amherst. His research focuses on machine learning in interactive and agent-based systems. He received his Doctor of Science degree in 1992 from Washington University in St. Louis for design and implementation of techniques for statistical significance testing in machine learning algorithms. Prior to joining the University of Massachusetts, he was an analyst in the Office of Technology Assessment, an analytical support agency of the United States Congress. He currently serves on the program committees of The Third International Conference on Knowledge Discovery and Data Mining and The International Symposium on Intelligent Data Analysis. He is a member of the American Association for Artificial Intelligence.

Dr. Jensen's primary technical contributions to artificial intelligence involve understanding the statistical implications of machine learning algorithms. He has developed theoretical frameworks and practical tools to account for overfitting, a widespread pathology of machine learning algorithms. In addition, Dr. Jensen has spent the past year developing machine learning tools for managing computer networks. His work has been funded by Hughes Information Technology Systems, NASA's prime contractor on the development of the core system for the Earth Observing Satellite Data and Information Systems (EOSDIS). When completed, EOSDIS will be the largest civilian data management effort to date. Dr. Jensen's work focuses on finding predictive sequences of events in computer networks. Such sequences can be used to identify sets of related events, in order to better understand the causes and remedies for the

underlying faults that cause those events.

Dr. Jensen will devote 70% of his time to the project. He currently receives support from Sterling Software and the DARPA/Rome Labs Planning Initiative.

Keith Decker is an Assistant Professor in the Department of Computer and Information Sciences at the University of Delaware. His research interests include cooperative distributed problem solving, multi-agent systems, computational organization design, real-time AI, parallel and distributed planning and scheduling, and distributed information gathering. He received his BS in applied math from Carnegie Mellon University in 1984, his MS in computer science from Rensselaer Polytechnic Institute in 1987, and his Ph.D. in computer science from the University of Massachusetts at Amherst in 1995. He is a member of IEEE and the American Association for Artificial Intelligence.

Dr. Decker's contributions to artificial intelligence center around coordination in multi-agent systems. He has also previously been involved in the study of uncertain reasoning (with Dr. Piero Bonissone at GE Corporate R&D), and real-time and parallel control methods for blackboard systems with Dr. Lesser. Decker is the developer of the TÆMS framework and simulator for modeling and analyzing complex multi-agent problem-solving environments. TÆMS has been used by others to model and analyze problems in network diagnosis, concurrent engineering, computational resource scheduling, and distributed data processing. He is also the developer of the GPGP approach to coordination. Recently, Decker has worked at CMU with Dr. Katia Sycara to implement a real multi-agent information gathering system for the portfolio management domain. This system has strong robustness characteristics (such as the ability for agents to come and go at any time without disrupting problem solving, assuming that redundant resources are available).

Dr. Decker will devote at least two months (summer) of his time to this project per year, and will actively be involved in the supervision of a graduate student working full time on this project throughout the year. Support is pending from Boeing Helicopter and the University of Delaware Research Foundation.

I. Cost

Primary Budget

	YEAR 1 Total	YEAR 2 Total	YEAR 3 Total	PROJECT TOTAL
PERSONNEL/FRINGE				
1. Lesser, Summer Salary	19,500	20,280	21,091	60,871
2. Jensen, Cal.Yr. Salary	38,500	40,040	41,642	120,182
3. Clerical Assistant, 10% Cal.Yr.	3,436	3,574	3,717	10,727
4. Post-doc (no benefits)	20,000	20,800	21,632	62,432
5. Research Assistants (4@$16,500)	66,000	68,640	71,386	206,026
Sub-total, salaries & wages	147,436	153,334	159,468	460,238
Fringe (on 2 & 3)	11,589	13,128	13,653	38,370
Workers' Comp. (on 1-4)	570	601	625	1,796
Unemployment (on 1-4)	244	254	264	762
Universal Health (on 1-4)	33	34	35	102
FICA (on 2 & 4)	848	882	917	2,647
Health & Welfare (on 2 & 3)	291	303	315	909
Research Assistants' Health fees	5,720	5,949	6,187	17,856
Sub-total Fringe	19,295	21,151	21,996	62,442
A. TOTAL PERSONNEL/FRINGE	166,731	174,485	181,464	522,680
B. TRAVEL	10,000	10,000	10,000	30,000
C. BOOKS & RESEARCH SUPPLIES	900	900	900	2,700
D. ADMINISTRATIVE COSTS	1,050	1,050	1,050	3,150
E. COMPUTER SUPPORT COSTS	21,441	22,372	23,210	67,023
F. EQUIPMENT	20,000	20,000	0	40,000
G. Subcontract to UDEL (see below)	53,042	54,860	56,639	164,541
Total Direct Costs (A thru G)	273,164	283,667	273,263	830,094
Indirect Costs (53%MTDC*)	119,315	110,668	114,811	344,794
TOTAL COSTS	392,479	394,335	388,074	1,174,888

*MTDC=TDC-(Eqpt + (Subcontracts in excess of $25K each subcontract)

Contractor Budget

	Year 1	Year 2	Year 3	Total
A. PI summer salary	13,026	13,547	14,089	40,662
B. 3 Grad student stipend	16,323	16,976	17,655	50,954
Total salary & wages	29,349	30,523	31,744	91,616
C. Fringe benefits	3,778	3,929	4,015	11,722

Total sal+wages+bens	33,127	34,452	35,759	103,338
E. 1 Domestic travel	2,000	2,000	2,000	6,000
H. Total Direct Cost	35,127	36,452	37,759	109,338
I. Indirect Costs	17,915	18,408	18,880	55,203
J. Total Direct + In	53,042	54,860	56,639	164,541

Budget Justification

Personnel: Professor Victor Lesser will assume primary responsibility for project management and high-level technical ideas. He will devote, at the minimum, one month summer, and will be actively involved in supervision throughout the academic year. In addition, Professor Lesser will directly supervise two full-time graduate students: one will be responsible for the development of detection and diagnosis algorithms while the other will work on the development of algorithms for organizational design.

Dr. David Jensen, a research computer scientist in our department, will devote 70% of his time to the project, and will be responsible for the work on univariate and multivariate learning. He will be assigned one full-time graduate student to assist his efforts.

The half-time post-doc and one additional full-time graduate student will be responsible for the development of the TÆMS simulation testbed.

Professor Keith Decker, through a subcontract with the University of Delaware, will devote two months summer per year, and will be responsible together with one full-time graduate student for the development of the extensions to GPGP that have been proposed.

Though this is the primary breakdown of responsibilities, it is expected that everyone will be involved in issues of evaluation and integration of components into a fully working prototype. Through this organization of staff, we believe that we can cover the major objectives that are laid out in this proposal.

Finally, due to the cooperative nature of this project (i.e., need to coordinate with the subcontractor), the need to consult with other contractors on applications, the large number of personnel, equipment buying, and substantial travel, we anticipate a level of administrative activity significantly beyond that experienced on other projects of this size. Therefore, funding for 10% of an Administrative Assistant's salary has been budgeted.

Travel The amount allocated for travel in the prime- and sub-contracts totals $12,000 which will cover: two travelers attending two contractors' meetings per year, at $500 per traveler, per trip ($2000/yr); one representative from University of Delaware traveling to the University of Massachusetts three times per year at $500/trip ($1500/yr); one representative from University of Massachusetts traveling to the University of Delaware three times per year at $500/trip ($1500/yr); three travelers to attend AAAI conference annually, at a cost of $1000 per traveler ($3000/yr); two to three travelers to attend the International Conference on Multi-Agent Systems (in Europe first year of contract, estimated at $1700 per traveler; in the United States in third year of contract, estimated at $1100 per traveler). We have had a high-acceptance rate at these conferences in recent years. In the event that more papers are accepted than anticipated,

it may be necessary to only partially fund some travelers' expenses. Any remaining travel funds are earmarked for travel to meetings with other contractors.

Supplies and Administrative Expenses Funds have been allocated for the purchase of technical books, manuals and research supplies specific to the needs of the project. The budget also includes $250 per year for express mail expenses, $500 per year for software upgrades, and $300 per year for long-distance phone and fax communications associated with the proposed research activities.

Computer Costs This service is provided by the Computer Science Computer Facility (on site at the University of Massachusetts). It is comprised of two components: maintenance and support of equipment and software, and central service fees. Central service fees include maintenance and support of communication networks, mass storage, file backup services, and consulting services.

Equipment Given the basic model that each researcher should have a dedicated workstation, six workstations are needed to support the research activities on this project. Currently, the lab is able to allocate four DEC/Alpha workstations to this effort.

In the first year we plan to purchase two high-powered workstations to reach a complement of one workstation per researcher and one PC-based workstation that will be used for document preparation. The moneys allocated in the second year will be used to either upgrade the existing four workstations, or replace two or three of them with new workstations. These four workstations are older DEC/Alphas that will no longer be appropriate by the second year of the project for running the large programs and simulations that will be required at that stage of the project.

Section IV. Additional Information

References

[1] Norman Carver, Zarko Cvetanovic, and Victor Lesser. Sophisticated cooperation in FA/C distributed problem solving systems. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 191–198, Anaheim, July 1991.

[2] Norman Carver and Victor Lesser. A new framework for sensor interpretation: Planning to resolve sources of uncertainty. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 724–731, August 1991.

[3] S. Cheng, J. Stankovic, and K. Ramamritham. Scheduling algorithms for hard real-time systems. In *Hard Real-Time Systems*. IEEE Press, 1988.

[4] Paul R. Cohen. How evaluation guides AI research. *AI Magazine*, 9(4):35–43, Winter 1988.

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

[6] Paul R. Cohen and David Jensen. Overfitting explained. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 115–122, 1997.

[7] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November 1991.

[8] Daniel D. Corkill and Victor R. Lesser. 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*, pages 748–755, Karlsruhe, Germany, August 1983.

[9] K. Decker, M. Williamson, and K. Sycara. Intelligent adaptive information agents. In *Proceedings of the AAAI-96 Workshop on Intelligent Adaptive Agents*, 1996. AAAI Press Tech Report WS-96-04.

[10] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering [poster]. In *Proceedings of the 2nd Intl. Conf. on Multi-Agent Systems*, 1996.

[11] K. Decker, M. Williamson, and K. Sycara. Modeling information agents: Advertisements, organizational roles, and dynamic behavior. In *Proceedings of the AAAI-96 Workshop on Agent Modeling*, 1996. AAAI Report WS-96-02.

[12] Keith Decker, Anandeep Pannu, Katia Sycara, and Mike Williamson. Designing behaviors for information agents. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

[13] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, 1995.

[14] Keith S. Decker. TÆMS: A framework for analysis and design of coordination mechanisms. In G. O'Hare and N. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 16. Wiley Inter-Science, 1995. Forthcoming.

[15] Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1996. Forthcoming.

[16] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[17] Keith S. Decker and Victor R. Lesser. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 360–366, Chambéry, France, August 1993.

[18] Keith S. Decker and Victor R. Lesser. A one-shot dynamic coordination algorithm for distributed sensor networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 210–216, Washington, July 1993.

[19] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):215–234, December 1993. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[20] Keith S. Decker and Victor R. Lesser. Coordination assistance for mixed human and computational agent systems. In *Proceedings of Concurrent Engineering 95*, pages 337–348, McLean, VA, 1995. Concurrent Technologies Corp. Also available as UMASS CS TR-95-31.

[21] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI Press. Longer version available as UMass CS-TR 94–14.

[22] K.S. Decker, V.R. Lesser, M.V. Nagendra Prasad, and T. Wagner. MACRON: an architecture for multi-agent cooperative information gathering. In *Proccedings of the CIKM-95 Workshop on Intelligent Information Agents*, Baltimore, MD, 1995.

[23] E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, November 1991.

[24] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August 1987.

[25] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

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

[27] Eugene S. Edgington. *Randomization Tests*. Marcel Dekker, 1995.

[28] Satoru Fujita and Victor R. Lesser. Centralized task distribution in the presence of uncertainty and time deadlines. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-96)*, 1996.

[29] Les Gasser and Toru Ishida. A dynamic organizational architecture for adaptive problem solving. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 185–190, Anaheim, July 1991.

[30] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, PA., August 1986.

[31] Piotr J. Gmytrasiewicz, Edmund H. Durfee, and David K. Wehe. A decision-theoretic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 62–68, Sydney, Australia, August 1991.

[32] Marcus J. Huber and Edmund H. Durfee. An initial assessment of plan-recognition-based coordination for multi-agent teams. In *Proceedings of the Second International Conference on Multi-Agent Systems*. AAAI Press, 1996.

[33] E. Hudlická and V. R. Lesser. Modeling and diagnosing problem-solving system behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):407–419, May/June 1987.

[34] David Jensen. Knowledge discovery through induction with randomization testing. In *Proceedings of the 1991 Knowledge Discovery in Databases Workshop*, 1991.

[35] David Jensen. *Induction with Randomization Testing: Decision-Oriented Analysis of Large Data Sets*. PhD thesis, Washington University, 1992.

[36] David Jensen. Adjusting for multiple testing in decision tree pruning. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 295–302, 1997.

[37] David Jensen, Tim Oates, and Paul R. Cohen. A prototype for ecs fault identification: Requirements and design. EKSL Technical Report #96-01, February 1996.

[38] S. Lander and V.R. Lesser. Negotiated search: Organizing cooperative search among heterogeneous expert agents. In *Proceedings of the Fifth International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, December 1992.

[39] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed. *AI Magazine*, 4(3):63–109, Fall 1983.

[40] Thomas W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33:1317–1332, 1987.

[41] T. Moehlman, V. Lesser, and B. Buteau. Decentralized negotiation: An approach to the distributed planning problem. *Group Decision and Negotiation*, 1(2):161–192, 1992.

[42] T. Moehlman and V.R. Lesser. Cooperative planning and decentralized negotiation in multi-fireboss phoenix. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*. Morgan Kaufmann, November 1990.

[43] D.E. Neiman, D.W. Hildum, V.R. Lesser, and T.W. Sandholm. Exploiting meta-level information in a distributed scheduling system. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, August 1994.

[44] Eric W. Noreen. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. John Wiley & Sons, Inc., New York, 1989.

[45] Tim Oates and Paul R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 346–354. Morgan Kaufmann Publishers, Inc., 1996.

[46] Tim Oates, David Jensen, and Paul R. Cohen. Technologies for fault management. EKSL Technical Report #95-09, December 1995.

[47] M. V. Nagendra Prasad, Victor R. Lesser, and Susan E. Lander. Learning organizational roles in a heterogeneous multi-agent system. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, 1996.

[48] M.V. Nagendra Prasad and V.R. Lesser. Learning situation-specific coordination in generalized partial global planning. In *AAAI Spring Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, Stanford, March 1996.

[49] Krithi Ramamritham, John A. Stankovic, and Perng-Fei Shiah. Efficient scheduling algorithms for real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(2):184–195, April 1990.

[50] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, August 1985.

[51] T. Sandholm and V. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 1995.

[52] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 256–262, Washington, July 1993.

[53] Chia Shen, Krithi Ramamritham, and John A. Stankovic. Resource reclaiming in multiprocessor real-time systems. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):382–398, April 1993.

[54] T. Sugawara and V. Lesser. Learning control rules for coordination. In *Multi Agent and Cooperative Computation '93*, pages 121–136, 1993.

[55] T. Sugawara and V. Lesser. On-line learning of coordination plans. In *Twelfth Annual Workshop on Distributed Artificial Intelligence*, 1993.

[56] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461, November/December 1991.

[57] Katia Sycara, Keith Decker, Anandeep Pannu, Mike Williamson, and Dajun Zeng. Distributed intelligent agents. *IEEE Expert*, 11:36–46, December 1996.

[58] G.I. Webb. Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[59] W. Zhao, K. Ramamritham, and J. A. Stankovic. Scheduling tasks with resource requirements in hard real-time systems. *IEEE Transactions on Software Engineering*, May 1987.