A Distributed Simulation System for Team Decisionmaking

Alan A. Song and David L. Kleinman
Department of Electrical and Systems Engineering
The University of Connecticut
Storrs, CT 06269-3157

Abstract

This paper gives an overview of a unique distributed, real-time simulation system for studying team decisionmaking and coordination - the DDD (Distributed, Dynamic, Decisionmaking) paradigm. The DDD paradigm captures the essential elements in real-world decisionmaking problems and integrates them into a controlled, computer-mediated, laboratory setting. The DDD simulation system is implemented on a network of UNIX workstations with real-time control, on-line data acquisition, interactive graphical display, and a simulated inter-human communication network. With a highly reconfigurable user interface and a flexible scenario generator, DDD has been used in many team decisionmaking experiments with different problem context, including military command and control, job scheduling, and medical diagnosis.

1 Introduction

In large scale systems that involve humans, machines, computers, etc., the problem scope and complexity often requires that the decisionmaking function be distributed over several humans. Quite often such systems have a team of human decisionmakers who are geographically separated, but who must coordinate to share their information, resources and activities in order to attain common goals in what is generally a dynamic and uncertain task environment. Although problem contexts can be different among various systems (e.g., military command and control, electric power distribution, air traffic control), the essential elements of decisionmaking remain the same. In order to study problems such as those above on a scientific basis, we have developed a unique distributed simulation system, termed DDD (Distributed Dynamic Decisionmaking) paradigm, that abstracts and simulates the essential elements of real world decisionmaking problems.

Unlike existing large scale simulation systems such as SIMNET [18] which stresses high fidelity, specialized, full scale simulation, the DDD paradigm stresses the small team (typically with less than ten team members) with an abstracted, low fidelity task environment, and emphasizes the basic aspects of interaction and coordination that are central to "teamness". It simulates the real-world problems in such a manner as to be amenable to study in a controlled laboratory setting. The task environment in DDD is reconfigurable for different problem context. For example, in our previous research, the system has been configured as naval command and control, military situation assessment, medical diagnosis, job scheduling in manufacturing systems, etc. The DDD system can be used as a versatile tool for studying/training small teams in military or industry.

The DDD paradigm is built upon the body of knowledge we have accumulated during the last ten years in performing model-driven, basic experimental research [1] [2] [3]. As the backbone of our normative-descriptive research for team decisionmaking and coordination, the DDD paradigm has been used for more than fifteen team-in-the-loop experiments, and proved to be a very powerful empirical research and training tool [6]-[17]. The DDD paradigm is implemented on a network of UNIX workstations, with facilities providing real-time control and on-line data acquisition, an interactive display/interface media, and a computerized inter-human communications and information network within which delay and occasional failure can be manipulated. The simulation system can run on workstations connected by a local area network, or on remote workstations connected by Internet. Its X11/Motif based graphical interface is highly reconfigurable (viz., for different problem context, the look and feel can be very different). Currently, it can support up to seven-person hierarchical or parallel team (expandable if desired). The DDD simulation system has the flexibility to examine a variety of ways in which information processing and resource allocation prob-

0-8186-6440-1/94 $04.00

lems can be solved by a team of decisionmakers (DMs) under different organizational architectures and information structures.

This paper gives an overview of the DDD simulation system with an emphasis on newly developed features that are not included in our early report [3]. The remainder of this paper is organized into two sections and a conclusion. In section 2, the team decisionmaking environment is described, and the basic elements of the DDD paradigm including resources, tasks, information, and responsibility are discussed. Section 3 reviews the main features of the simulation system including system architecture, user interface, scenario generator, experimental variables, built-in distributed database, and training support tools. Finally, section 4 offers concluding remarks.

2 Basic Elements of DDD paradigm

The DDD paradigm is implemented as a computer-driven interactive game among several decisionmakers (DMs) who may be geographically separated (see Figure 1). In a real time simulation session, each DM sits at a workstation which is capable of displaying the tactical situation and sending/receiving information to/from the other players. Team decisionmaking is formulated as a process of allocating limited resources to a variety of tasks in a dynamic and uncertain environment. Thus, the essential elements of team decisionmaking are abstracted as: i) resources (e.g., machine tools, man powers, sensors, weapons, etc.), ii) tasks (jobs to process, e.g., parts, unidentified targets, enemy airplanes, etc.), iii) information (e.g., sensor measures, intelligent sources, reports, etc.), and iv) responsibility (i.e., who should do what, at what time). To achieve the team goal, co-acting DMs must process distributed information to: i) estimate/identify various task attributes, and ii) determine and schedule their resources to process specific tasks. The DMs are thus required to coordinate their information, actions, and resources in a timely and accurate manner. Below we describe in more detail the salient elements of the DDD paradigm.

2.1 Resources

Resources are basic elements of the system. A resource can carry other resources called sub-resource, for example, a destroyer can carry some helicopters, and the helicopters can carry some sonobuoys, etc. In this way the resources can be nested down to any desired level of detail.

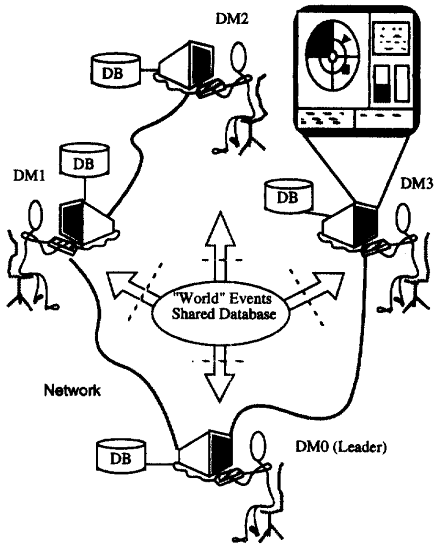

Figure 1: The Distributed Dynamic Decisionmaking Environment

The resources are divided into several classes depending on the design parameters of the experiment. All resources of a given class will have the same features with respect to capacities (i.e., sensor range, weapon strength, etc.). The only difference among resources in a given class is the number of sub-resources each carries.

The sub-resources are located on board their parent resource. A sub-resource does not become an independent resource until it is launched from the parent resource. The DM can launch one or more sub-resources that will become available after a certain launch time delay. The sub-resources can only stay away from their parent for a limited time period. An industrial example of resource/sub-resource could be the manager (resource) that hires temporary employees (sub-resources).

The strength of a resource can be described as a generalized vector which stands for strength in different aspects. Each resource has its effective range. For example, a sensor resource can have three ranges: a detection zone, a measurement zone, and a classification zone.

Each resource is controlled by the DMs who own it(a resource can be owned by multiple DMs), and the control of any resource may be transferred during the simulation from one DM to another with an attendant transfer time delay.

2.2 Tasks

A team is presented with multiple tasks having different deadlines, processing times, attributes, and priorities. During the real time experiment, tasks appear, move/maneuver and disappear according to a scenario that is under the control of the experiment designer.

The tasks are also divided into classes. For example, we can have AA, AB, AN which may correspond to different air targets such as a backfire bomber, a bird, or a civilian airplane. The hostility of each task class, i.e., whether they are threats or neutrals, can be defined by the experiment designer.

Each task has an attribute vector a with elements that characterize it quantitatively. For example, the attributes can include strength, evasiveness, vulnerability, etc. These attributes are random from task to task, but have a probability distribution (mean and standard deviation) that is unique to task class. The resources r required to successfully process a task is a mapping of the attributes of that task and will generally depend on task class.

Tasks can be processed in one or more operations, each operation can be assigned to different DMs. Two types of processing are possible: sequential and parallel. The sequential processing requires two or more DMs to process in sequence, the next operation cannot be started before the current one is finished, for example, a part in a manufacturing line may need molding, painting, and assembling. The parallel processing requires two or more DMs (or resources) to process at the same time, all required operations must be synchronized to complete the processing, for example, to diagnose a disease, all blood test, X-ray, and urine test must be finished before the final decision can be made.

The DDD also includes complex tasks such as active tasks and dynamically attributed tasks. An active task can change its trajectory according to the current situation and the treatment it received. A dynamically attributed task can change its attributes as a function of time and/or location of the task (for a simple task, the trajectory and true attributes are set by the scenario generator and remain unchanged during the real-time session). These complex tasks provide facilities to investigate team decisionmaking and coordination issues in more complex and reactive task environment.

2.3 Responsibility Structure

The overlap in task processing responsibilities of the team members can be adjusted based on the experimental condition. Responsibility can be preassigned in a variety of ways, e.g., by task class or by geographical location. Under the conditions of no overlap we have a disjoint team requiring no coordination in task processing. As overlap is increased, conflicts in the overlapping areas of joint responsibility will occur which will need to be resolved through coordination. A new feature of the DDD is the ability to modify online task responsibility on a task-by-task basis. Thus, the responsibility for individual task prosecution can be (re)assigned dynamically by the team leader.

2.4 Information and Communication

Information and communication are two major aspects in team decisionmaking and coordination. Different structures of information/communication and their impacts on decisionmaking are important research issues. The DDD paradigm provides a variety of mechanisms to manipulate information and communication.

The information structure of the team can be manipulated easily via the DDD paradigm. This is implemented by establishing an information network within the simulation system. Every DMs can be assigned a level of "tie-in" to the information network depending on different task. A high level of "tie-in" means that the DM can get almost all measurements obtained by other DMs, and a low level of "tie-in" means that the DM can only rely on his own resources. Therefore, a centralized, a partially centralized or a decentralized information structure can be accomplished by setting different network "tie-in" levels by task type for different DMs. Furthermore, different roles in a hierarchical team may have different levels of information aggregation. For example, a team leader may have information on overall situation without details, in contrast, a subordinate may have information on detailed local situation within his responsibility.

Communication among DMs is the major way in which the team members can share their local information, and coordinate actions on resource transfer and task processing. In DDD paradigm, communication between different DMs is mainly carried out by electronic messages (Verbal exchanges based on multiperson communication/recording system can also be incorporated to the DDD paradigm, see [16]). In order to simplify the data analysis, all electronic message are preformatted. Our underlying model for the communication channel contains a variety of features that are important to human decisionmaking. To simulate the communication and data processing delay in real situations, a (random and/or fixed) time delay in message transfer was introduced. To simulate the limitation on communication capacity (or channel access), the number of communications (N) in a fixed time window (T) can be specified. Message loss and information scramble due to the network failure can also be simulated within the DDD. Finally, the communication network structure can be defined by a communication matrix, i.e., who can communicate with whom.

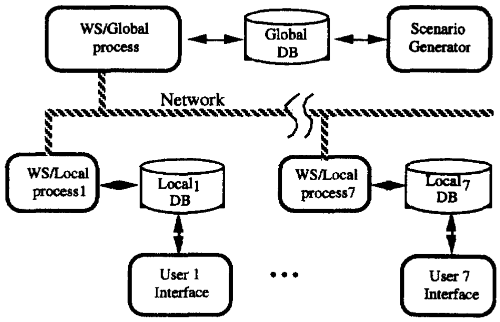

Figure 2: The Architecture of the DDD Paradigm

3 The Features of DDD Paradigm

3.1 System Architecture

The general architecture of the DDD environment is shown in Figure 2. The DDD paradigm runs on a network of UNIX workstations. In a real-time simulation session, eight (or more) processes run concurrently on different workstations, with all of the control and communication information traffic carried over network. In the figure, Global is a process that works as the "control center" for the environment by controlling the clock and timing, synchronizing the other processes, and sending out various control messages according to the experimental scenario. Each Local Process controls execution within a workstation (WS) and interacts with the User Interface and the Global Process. Each User Interface receives commands from a DM and displays the dynamic tactical situation. The Scenario Generator is used for assisting the experimental designer in developing various system parameters for a given experiment.

In the DDD environment, the global and local processes are implemented via a message-passing approach. Each action of the DM is composed of certain events transferred in the form of messages. For example, when a display object receives a "process" command issued by DM through a mouse/keyboard event, it sends a message "PROCESS EVENT" to a local database object that triggers the method "process" which in turn sends a message to the global process and then other local processes to update the state of all relevant objects. Thus, synchronization is achieved via the LIFO queueing and processing of messages.

3.2 Interactive Display

The user interface in DDD is very flexible. While all the facilities for decisionmaking are basically the same, the look-and-feel can be different according to different problem contexts of the scenario. Some examples of display at an individual node are shown in figure 3 and figure 4. Figure 3 is the screen of our basic paradigm [2] [3], three types of targets are shown on the screen: air, surface, and submarine; the resources are ships and airplanes, and sub-resources are helicopters. Figure 4 is a screen form our REST (Reward Structure) experiment [11], where triangles and circles represent targets assigned to different decisionmakers (combined triangle and circle means two DMs are responsible for the target). In these figures, the screen is divided into four major parts: the main display, the status panel, the communication panel and the prompt panel. The main display displays the objects that represent resources and targets. Different targets arrive and move according to a experimenter-defined scenario; targets must be processed within a limited time window. All commands related to the objects can be issued by pull-down menus, pop-up windows or double- click associated with the objects. The communication panel is composed of an incoming window which is used to display the messages from other players, and an outgoing window which is used to display the feedback information when a message is sent out. The status panel is used to display the current time, strength and the dynamics of resource transfer/utilization. The prompt panel is used to display prompts or error messages. All the display icons, menus, windows and messages can be modified or tailored according to different experimental designs.

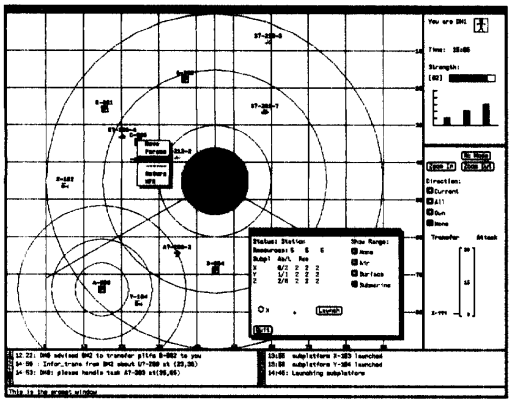

Figure 3: The Screen of the Basic DDD paradigm

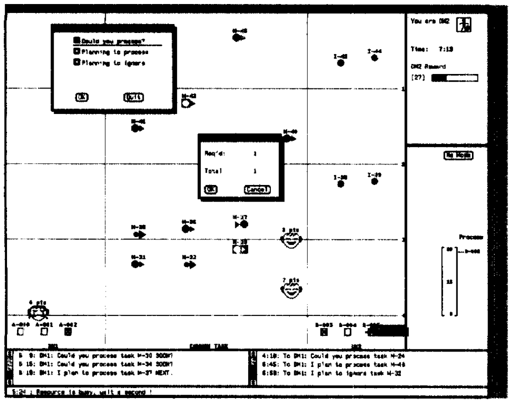

Figure 4: The Screen of the REST Experiment

3.3 Scenario Generator

A scenario generator has been developed to assist the user in setting up the experiment. It is used to configure the resources, to define the tasks, and to design the movements of tasks. The scenario generator is capable of representing a stochastic and imperfectly known environment. For example, unexpected or low probability events can be introduced, and false information and/or false threats can be employed to perturb the system. The intention is to represent a world that is difficult to predict, in which a hostile adversary introduces uncertainties into one's estimation of the current state of the system, thus making inferences about future states rather unpredictable. All task arrival times, task arrival positions, and task movements can be either automatically generated according to a certain random function, or a certain pattern, or specifically designed on a task by task basis.

Two interfaces are provided for the scenario generator. The first one is a flexible experiment description language, XS language, which can be used to define the "rules" of the DDD game, describe the resource and the task environment. Three types of items can be described via XS language, they are: 1) general items; 2) resource information; 3) task information. In general items one can set the overall features of the experiment, such as the numbers of DMs, simulation time and communication delay etc. In resource information one can describe the characteristics of the resources such as maximal velocity, strength, and ranges, etc. In task information one can define task attributes, the resource required to prosecute the task, the decision-makers who are able to see or process the task, etc.; one can also describe the task arrival times, initial positions, velocities, and the maneuvers of the tasks.

A graphical active database modeling tool for scenario generation has also been developed [5]. This tool has utilized data modeling techniques to correctly and precisely specify large amount diverse, intricate, and interdependent information including the structure of the decision team, the sharing of data, the interaction and exchange of data among DMs, and the data required by the different DMs. Furthermore, the structural information can be graphically specified by the experimenter, and changes in structural information automatically cascades to investigate changes of related information throughout the experimental scenario, resulting in time saving and consistent design.

3.4 Experimental Variables

The DDD paradigm is powerful enough to manipulate a variety of independent variables(IVs) that allow for the study and evaluation of different command and control configurations. Some of the major IVs are: i) internal variables (team structure, responsibility structure, information structure, and communication structure), and ii) external variables (tempo, uncertainty, resource quantity, information quality).

The number and type of dependent variables (DVs) this paradigm can handle is quite flexible. To date, over 100 performance, strategy, coordination, and workload measures have been collected and analyzed in various experiments.

All essential operations taken by the DMs are recorded in a log file. This file can be used to generate various dependant variables and statistics. Another important function of this file is that it can be used in

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**