A Task Process Pre-Experimental Model[1]

Track: Modeling and Simulation

Authors:

Holly A. H. Handley
hhandley@gmu.edu
C3I Center, MSN 4B5
George Mason University
Fairfax, VA 22030

Alexander H. Levis
alevis@gmu.edu
C3I Center, MSN 4B5
George Mason University
Fairfax, VA 22030

Point of Contact:

Holly A. H. Handley
10204 Rue Cannes
San Diego, CA 92131
858-695-8958
hhandley@msn.com

[1] This work was supported by the Office of Naval Research under grant no. N00014-00-1-0267

A Task Process Pre-Experimental Model

Holly A. H. Handley and Alexander H. Levis[2]
C3I Center, MSN 4B5
George Mason University
Fairfax, VA 22030

Abstract

The Adaptive Architectures for Command and Control (A2C2) program is a multidisciplinary program that employs a scientific basis for designing and analyzing adaptive and reconfigurable organizational structures at the Joint Task Force level. As part of its unique model-driven experimentation method, a pre-experimental model is created to support the formulation of hypotheses, the determination of key variables and parameter values, and the prediction of organizational performance. The pre-experimental model is used to explore the parameters of the experimental design in order to determine the appropriate region to conduct officer-in-the-loop experiments at the Naval Postgraduate School. A pre-experimental model based on the task process was created for an upcoming A2C2 subject experiment, which will examine the congruence between organizational structure and mission requirements. The pre-experimental model is a dynamic model created with Colored Petri nets, which can represent the changes in the task environment over time by implementing the stages of the tasks (i.e., detection, identification, attack, destroy, and disappear). The simulator used in the subject experiments, Distributed Dynamic Decision-Making (DDD), records timing information over the life of each task. Therefore, timing information regarding the tasks can be extracted from the output files of the trial experimental runs and included in the model before the final experimental simulations. In this way the model can be validated at the pre-experimental stage.

Introduction

The Adaptive Architectures for Command and Control (A2C2) program supports experimental research driven by models of organizations and adaptation in Joint Command and Control architectures. A basic premise of the program is that the organizational structure should be "matched" in some sense to the mission that the organization faces, and that the changes to the mission structure should in turn induce changes in the organization [Kemple et. al., 1997]. The types of changes that are most likely to induce adaptation and the types of organizations that are most likely to adapt are the focus of the series of experiments. The experiments involve a Joint Task Force in a six-node hierarchical organization conducting a multifaceted amphibious operation. In the first experiment, two organizations with different hierarchies were used to examine the relationship between the levels of hierarchy and the asset coordination requirements. In the second experiment, a traditional hierarchical architecture, designed by following standard doctrine, and a non-traditional architecture, derived from a model that

[2] This work was supported by the Office of Naval Research under grant no. N00014-00-1-0267.

minimized decision maker coordination, were used to determine if the novel distribution of resources in the non-traditional architecture facilitated the performance of the mission tasks [Entin et al., 1997]. The third and fourth experiments allowed the number of nodes in the architectures to vary, with four, five, and six node architectures, and also varied the communication nets between the nodes. For each of these experimental iterations, a pre-experimental model was created [Handley, et al., 1999].

Since experiments involving humans are difficult to design and control, a large number of trials are not feasible. A pre-experimental model can help determine which variables should be varied over what range and what variables should be measured. A pre-experimental model has been created to support the experimental design for an upcoming A2C2 subject experiment, which will examine the congruence between organizational architectures and mission requirements. In this experiment, two different scenarios will be designed, one termed "functional" and one termed "divisional". Two different architectures will also be created, again one termed "functional" and one termed "divisional". Each architecture will complete both scenarios; the hypothesis is that the architecture will better perform the scenario that matches its design, i.e., the functional architecture will perform the functional scenario better than the divisional one. The pre-experimental model will be used to confirm that the proposed subject experiments will actually produce constructive output, i.e., that the proposed scenarios are different enough to elicit measurable performance differences between the two proposed architectures.

During the development of the pre-experimental model, data was available from a trial experiment that had been conducted with preliminary versions of the two proposed architectures. The output data from the trial experiments was used to validate the behavior of the pre-experimental model by comparing the model results to the trial results. The model does not attempt to replicate the experiment in total, but rather it elicits specific behaviors and performance issues related to the experimental design. In order to complete the pre-experimental modeling, the model will be reconfigured for the two final versions of the architectures under study and executed under the different proposed DDD scenarios in order to test the hypothesis of congruence.

Computational Modeling

Computational or computer modeling can contribute significantly to research on organizational theory by modeling the organization as a system, including the decision makers in that system. A model can simulate both the behavior of the decision makers and the behavior of the organization as a whole. In the A2C2 program, pre-experimental modeling has served as an initial testing and research design avenue for the architectures under consideration for use in the subject experiments. Pre-experimental modeling can also be used to evaluate the developing experimental design. Computer simulations can be viewed as experiments done in silica [Ilgen and Hulin, 2000]. Experimental designs rely on the control over one set of variables to draw inferences about the effects on other variables, which are allowed to vary freely. Simulations can be concentrated in regions of the experimental space to maximize that information. Modeling can indicate what

empirical data to expect based on the model operating within configurations of organizational parameters and environmental constraints. The output of the computational model can then be compared with empirical traces of behaviors of similar systems in the real world, such as the output of subject experiments, to validate the behavior of the model.

For each subject experiment, a military scenario is developed that describes the mission to be accomplished and what the final objective is. The mission is often, but not always, described with a task graph that illustrates the sequence in which the major tasks, called mission tasks, must be performed. The term task refers to an entity within the scenario that must be done; there are both defensive and offensive tasks. Mission tasks are often large in scale, such as an amphibious beach assault or an infantry attack on an airport. These mission tasks may be broken into smaller subtasks that need to be completed in parallel or in sequence by one or more decision makers. Also present in the scenario are tasks that occur independent of the mission. These tasks are often called "mosquito" tasks, as they serve to distract from the mission, such as a land mine or a SCUD missile. Each independent task has a single occurrence, can usually be handled by one decision maker using one resource, and has a finite lifetime. Tasks are accomplished or attacked with resources; resources are the assets available to the decision makers to process tasks. Each task has specific attributes that must be matched to an appropriate resource. If an independent task appears and is not attacked with a resource, then it will either launch a new task, as in the case of a task such as a missile launcher, or collide with its intended target and cause damage, as in the case of the task of the launched missile.

Previous models for the A2C2 program used an object oriented approach with three entities, decision makers, tasks, and resources and modeled the relationships between them [Handley et al., 1999]. The complete assignment of task to decision maker to resource was pre-determined. The architectures were evaluated based on the completion time of the five objective tasks; tasks were performed in a precedence order, as pre determined by a task graph, and the end points of the graph were the objective tasks. Latency built through the course of the scenario due to delays associated with the wait for busy resources and the travel time of platforms to locations. The pre-experimental modeling evaluated compressed and expanded scenario tempos to identify the range where the architectures under consideration would show the most variability.

The current modeling approach, however, is based on task processing (the task lifetime) and has several more degrees of freedom than the previous approach [Handley et al., 1999]. Tasks are mapped to decision makers as they appear in the scenario based on heuristics compatible with the type of architecture. The tasks are all considered independent tasks, and each task is evaluated separately. The decision maker can match any resource that he controls to the task based on task attribute requirements; the decision maker, task, resource assignment is not pre-defined as in previous models. This allows much more variability in the model, that more closely replicates the variability seen in the team trials.

Pre-experimental Model Design

An architecture can be characterized by the people, resources, and tasks and the relationships between them [Carley and Ren, 2001]. This can be shown in tabular form as shown in Figure 1. In order to model the experimental architectures, each relation must be evaluated and implemented.

	People	Resources	Tasks
People Relation	Social Networks *Who knows whom?* [Decision makers process tasks independently.]	Capabilities Network *Who has what resource?* Different architectures induce different resource capabilities.	Assignment Network *Who does what?* Tasks mapped by target or attribute depending on architecture.
Resources Relation		Substitution Network *What resources can be substituted for which?* Decision makers choose any resource that he controls with the correct attributes for the task.	Requirements Network *What resources are needed to do what task?* Resource matched to task based on task attributes.
Tasks Relation			Precedence Network *Which tasks must be done before which?* [Tasks are independent]

Figure 1: Architecture Network Parameters

Since the model was designed based on the experimental design and results of a preliminary trial experiment, some of the constraints currently implemented will be different in the final experimental design; these are shown in brackets in Figure 1. In the trial experiment, all tasks were processed independently by a single decision maker; this will change in the final experimental design to allow coordinated processing, so the social network parameter shown in Fig. 1 is in brackets. In the trial experiment, the mission was to "establish air and sea superiority in the theater and clear the area of hostiles that could pose a threat to the introduction of subsequent ground forces." All tasks were independent tasks with no precedence network. Several tasks "spawned" subsequent tasks, but there was no formal mission graph for the scenario. However, a formal task graph with precedence is being developed for the final experimental scenarios, so the tasks relation is also shown in brackets.

The experimental design consists of two different architectures, a functional architecture and a divisional architecture. The functional architecture was created by assigning resources to decision makers based on their function, such as Anti Aircraft, Strike, etc. The divisional architecture was created by assigning resources to decision makers based on the platform on which the resource resides; a single decision maker was responsible for all the resources on a single platform, such as a Destroyer. The two architectures induce two different capabilities networks; therefore the capabilities network is an independent variable in the experiment. These different architectures induce different mapping strategies that result in different assignment networks. The requirements network is a function of the tasks; an appropriate resource is matched to the task based on the attributes of the task. A decision maker may control several appropriate resources for a task; he may choose any of the resource of the correct attribute that he controls; no

other type of resource sharing between decision makers or substitution network among resource types exists.

The task process model follows the "life" of a task, as shown in the block diagram in Figure 2. The first stage, *Appear*, occurs when a task is first present in the environment. This is controlled by the input scenario which has an appear time for each task. In the case of a mobile task, such as a missile, it will continue along a trajectory; in the case of a fixed task, such as a missile launcher, it will be fixed at one location. As soon as the task is noticed, either by a decision maker or a sensor, it is *Detected*. This stage initiates the processing of a task by a decision maker. The task is then *Identified*; this indicates the decision maker knows what type of task it is and what type of resource can be used to process the task. When an appropriate resource is launched and travels to collide with the task, the task is *Attacked*. When the resource has succeeded in finishing the task, the task is *Destroyed*. When the task has disappeared from the simulator screen, it has *Disappeared*.

Each stage of the task may induce a delay in the model. Information on the task delays can be deduced from information in the input and output files of the Distributed Dynamic Decision-Making (DDD) simulator. The DDD simulator is a software platform that creates the environment for the subject experiments. It captures the decision maker's actions and task data through the course of the scenario. This information is made available after the experiment in log files, which can be sorted by decision maker, resource or task identifiers to find timing information. The time the task appears in the scenario is preset in the scenario input file. A task must be identified by a decision maker and a resource chosen; the launch and travel parameters of that resource can also be found in the DDD setup files. When a task is attacked or attacks, it has a pre determined processing time before it disappears.

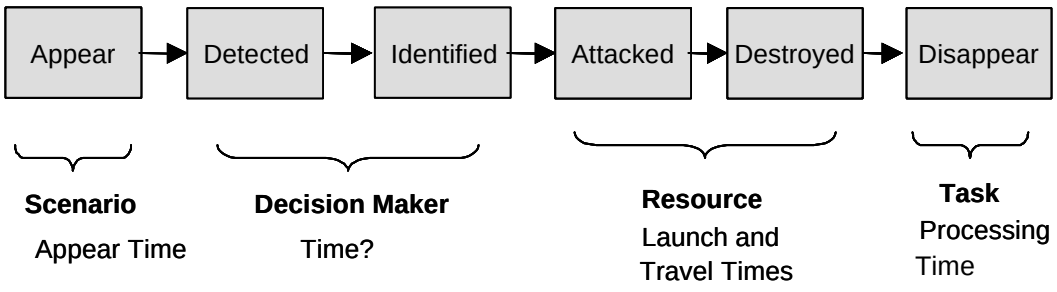

Figure 2: Task Process Model Block Diagram

However some delays are not apparent; these are the delays that concern the decision maker – how long he takes to identify the task and how long he takes to select a weapon. This information needs to be inferred from the output files of the DDD in order to determine how long the trial decision makers spent on these stages of the task. An attempt was made to extricate this information from the trial log files by task class, but the data was inconclusive. Particularly troublesome was that the DDD did not record when a task was first acted upon, i.e., noticed or identified by a decision maker. Hence

the delays concerning the decision maker's actions are incorporated as a variable in the model. Pre-experimental modeling runs will be conducted in order to determine the appropriate range of decision maker delays to configure the model.

The model was created using Colored Petri nets, a graphical modeling language and a useful and powerful modeling tool to expose critical time dependencies, task concurrencies, and behavior that is event driven. They can represent the external interactions of the decision makers as well as any internal algorithms the decision maker must perform. Colored Petri nets have both graphical and mathematical properties. For a complete description of modeling A2C2 architectures see Handley et al. [1999]. A portion of the current model, the detection and identification stages, is shown in Figure 3.

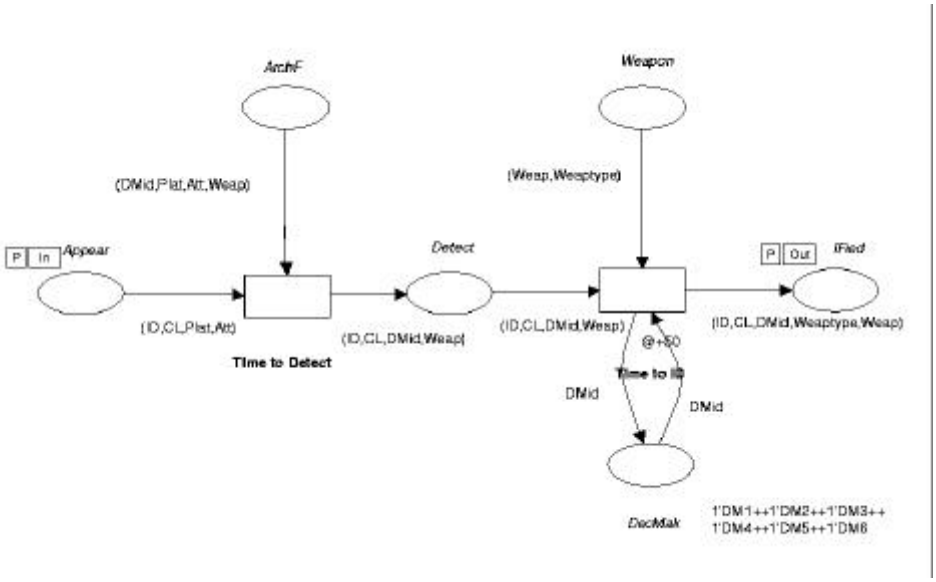

Figure 3: Task Detect and Identify

When a task "appears" in the model, it is matched to a decision maker in two ways. In the divisional model, the task is pulled by the decision maker that commands the platform targeted by the task. For example, if it is a missile aimed at the DDG, the decision maker that commands the DDG will process the task. In the functional model, the task is pulled by the decision maker that commands the resources that are necessary to counter that task. The decision maker will then assign a resource to that task. Each decision maker is assigned a set of resources and he will assign an available resource with the correct attributes to match the task. The available resource is launched and travels to the task; each resource has a specific launch time and an endurance value which combines range and velocity. When the resource meets the task, the task processing time is invoked and then the task disappears.

Validating the Pre-experimental Model

Two versions of the model were created: one to represent the functional architecture and one to represent the divisional architecture. Each architecture had six decision makers with resources assigned based on the chart shown in Figure 4. In order to simulate the model, a scenario was needed with a list of tasks, their targets and the time the tasks first

appear in the simulation environment. A scenario of 118 tasks, with appear times from 1 to 2400 simulation units was created based on the input file to the DDD simulator used for the trial experiment. Using this scenario, baseline simulations of the functional and divisional architectures could be completed. The output of these baseline simulations was used to perform an analysis on the behavior of the model, specifically what decision maker performed which tasks and what tasks were "missed". These two parameters could be compared to the output log file of the trial experiment to indicate if the model was behaving similarly to the trial teams.

	Divisional	Functional
DM1	CVN	STRK
DM2	DDGA	ASUW
DM3	DDGB	SAR/ISR
DM4	CG	AAW
DM5	FFG	BMD
DM6	SSN	ASW

Figure 4: Experimental Architectures

The first parameter examined was the allocation of tasks to decision makers in order to determine if the modeled decision makers completed the same tasks as the trial decision makers. The results of this comparison are shown in Figure 5. In the divisional architecture, the SSN (Submarine) is troublesome and so is the ASW (Anti Submarine Warfare) in the functional architecture. There were two issues with the submarine: first it was the only platform that moved, and second it was not a target for any specific task. The submarine has since been removed from the final experimental design, and so the model task-assignment heuristics results are acceptable.

Divisional	Tasks	Different	%	Functional	Tasks	Different	%
CVN	13	3	23	STRK	26	1	4
DDGA	18	2	11	ASUW	17	2	12
DDGB	15	4	27	SAR/ISR	3	0	0
CG	18	4	22	AAW	25	0	0
FFG	13	1	8	BMD	5	2	40
SSN	10	8	80	ASW	7	5	71

Figure 5: Comparison of Model and Trial Task Assignment

The second parameter used to validate the model's behavior was the comparison of the tasks the model could not complete to the tasks the trial teams could not complete. The results are shown in the chart in Figure 6. Again, the percentages for task completion between the model and the trial results are comparable.

Divisional	Trial %	Model %	Functional	Trial %	Model %
CVN	80	88	STRK	71	88
DDGA	74	72	ASUW	86	72
DDGB	70	89	SAR/ISR	100	89
CG	71	84	AAW	86	84
FFG	81	74	BMD	81	74
SSN	70	80	ASW	78	80

Figure 6: Comparison of Model and Trial Task Completion

Once the model behavior was validated, the model performance could be examined. This is an important step to ensure that the model will be a reasonable predictor of the experimental performance, a key issue in the current experimental design. The model performance is evaluated by comparing the timeliness of the model's response versus the timeliness of the trial team's response to the tasks in the scenario. The completion time of the tasks, compared to the appear times of the tasks can be graphed for both the baseline model and the trial output. Figure 7 shows the baseline model output and Figure 8 shows the trial output. A straight line fit of the data yielded y = .96x + 90.41 for the data of Fig. 7 (the model) and y = .89 + 468.88 for the data of Fig. 8 (the trial). The slopes (.96 and .89) are similar but the variance is not. The coefficient of correlation (r^2) for the model is .99, indicating that the line is a good fit for the data. The r^2 value for the trial data is .82, reflecting the dispersion of the data points. Note that the baseline performance of both the modeled functional and divisional architectures was similar, because the differences that arise due to the mapping of the tasks to different decision makers are not included in the baseline simulation, only the delays due to the resource and the task. The baseline simulation allows the decision makers to complete multiple tasks in each interval, with a nominal delay. When the delays due to the decision maker are included in the simulation, the model output should approach the dispersion seen in the trial output, and differences in the functional and divisional architecture will be evident.

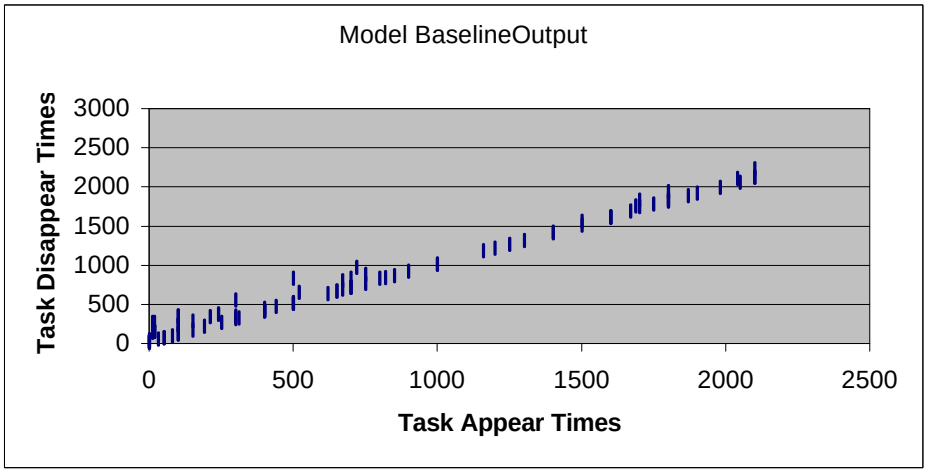

Figure 7: Baseline Model Output

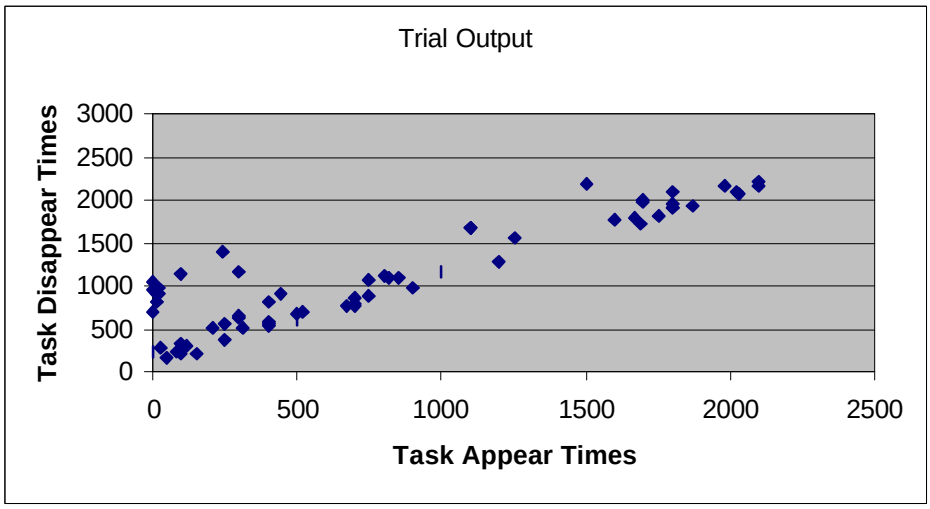

Figure 8: Trial Output

An attempt was made to calculate the decision maker delay values empirically from the data available in the trial output files. However, the correct data was not available to determine these values. Unfortunately the DDD files did not record the time the task was initially recognized by a decision maker in the trial runs; this time value could have been used to determine the decision maker's internal delay of assigning a resource to the task. This value will be recorded in the actual experiment output. It has also been determined that the complexity of the task may also affect the decision maker delay; therefore it will be calculated with the experimental output files by task class. This was not possible with the trial data, as most classes did not have enough tasks occurring to draw any statistically significant conclusions.

The decision maker's processing time can be affected in two ways. One is the number of tasks a decision maker can process in one time interval and the second is how much decision time is spent on each task. The correct combination of these two variables determines the point at which the decision maker is operating. In order to determine the missing decision maker delays, two model parameters can be varied: the number of tasks a decision maker can complete in one time interval, and the length of the time interval to process the task. As these parameters are varied, the model delays are expected to approach the trial delays. Many iterations of simulations were conducted, varying the number of tasks a decision maker can complete from 1 to 5 and the length of the interval from 1 to 100; the results of decision makers completing one task per interval, and a decision interval of 50 simulation units was the closest match to the trial results. The results of the divisional architecture simulated under these conditions are shown in Figure 9. Increasing the number of tasks per time period had only a limited affect on the performance, as it was only important when multiple tasks were present for a single decision maker. The time for task processing was a much more dominant factor in determining the delay of this stage, as this delay often induced delays in subsequent tasks. The output in Fig. 9 ($y = .91x + 273.99$) has an r^2 value of .89, much closer to the trial output in Fig. 8, and therefore the appropriate operating region has been determined for the decision makers for the pre-experimental model.

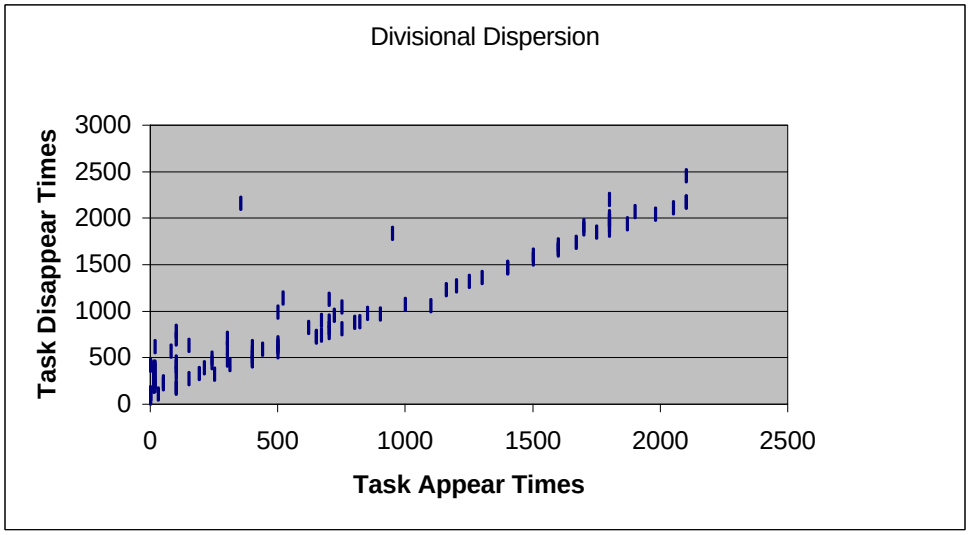

Figure 9: Divisional Model Output with Decision Maker Delays

An important aspect of the upcoming experiment is identifying congruence, or fit, between the scenario and the architectures. The scenario used for the trial experiment was not classified as favoring either a divisional or functional architecture. However, by using the model, the outputs can be compared to see which architecture has a better fit with the scenario. The trial output for the functional architecture, at the same restrictions as the divisional architecture, is shown in Figure 10. With the dynamic model, congruence can be evaluated over time, that is, when an architecture is incorrectly matched to a scenario, the performance decreases, either overall or in intervals, as decision makers become overworked and/or in-demand resources become delayed. A preliminary congruence analysis shows that the scenario favors the divisional architecture as the functional graph in Fig. 10 shows a high percentage of tasks delayed early on in the scenario. Architectures that are incongruent with the scenario will have a higher percentage of tasks with low timeliness scores. Fig. 10 ($y = .78x + 462.88$) also has an r^2 value of .72, well below the divisional data value, indicating the divisional data has a better fit.

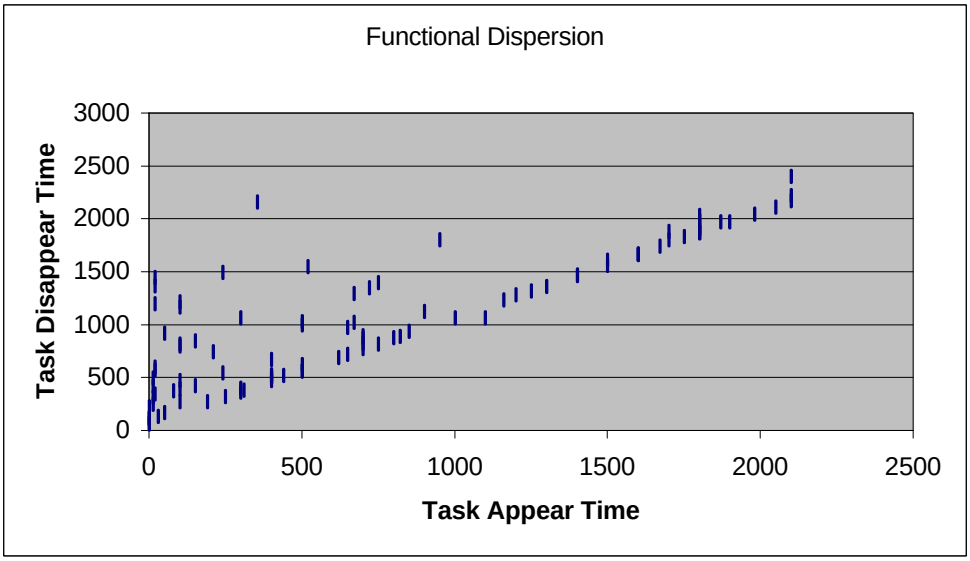

Figure 10: Functional Model Output with Decision Maker Delays

The model can be now used to confirm whether the proposed experiment will actually produce useful output. The current experimental design crosses functional and divisional architectures with functional and divisional scenarios. The performance of the non-congruent pairs, i.e. the functional architecture with the divisional scenario, will be compared to the performance of the congruent pairs, i.e. the functional architecture with the functional scenario. A concern of the experiment, a priori, is if the functional and divisional scenarios will produce discernable differences between the functional and divisional structures. The pre-experimental model will be used, when the architecture and scenario designs are finalized, to run both the congruent and the non-congruent pairs and determine the variation in the outputs.

Conclusions

This paper describes the development of a pre-experimental model for an upcoming subject experiment. It was designed to implement the time based task measures that are recorded in the DDD simulator, which provides the experimental environment. By using the output of a trial experiment, the behavior of the model could be validated: the way in which the model assigned tasks and the percentage of tasks that were completed were compared to the trial output. Because the time delays of the task stages controlled by decision makers were unavailable, iterations of simulations were run to find the correct operating region to configure these delays in the model. As congruence is the dominant issue in the upcoming experiment, the pre-experimental model was used to determine the congruence between the two trial architectures and the trial scenario; the scenario was determined to better match the divisional architecture.

The model is now ready to perform pre-experimental simulations on the final architectures and scenarios to be used in experiment, when they become available. One of the values of modeling is that it allows the modeler to incrementally incorporate

additional factors in a consistent manner, validating the model at each step. In this way, insights about design, performance and change build up incrementally. Of particular interest is the confirmation that the different architecture-scenario combinations will provide discernibly different results. Because the model has already been validated with the trial output data, the pre-experimental model can provide that answer.

References

[Carley and Ren, 2001]. Kathleen M. Carley and Yuqing Ren, "Tradeoffs Between Performance and Adaptability for C3I Architectures," *Proceedings of the 2001 Command and Control Research and Technology Symposium*, US Naval Academy, Annapolis, MD.

[Entin et al., 1997]. Elliot E. Entin, Caroline Kerrigan, and Daniel Serfaty, "Performance Under Non Traditional and Traditional Architectures, *Proceedings of the 1997 Command and Control Research and Technology Symposium*, National Defense University, Washington, DC.

[Handley et al., 1999] H. A. H. Handley, Z. R. Zaidi, and A. H. Levis, "The Use of Simulation Models in Model Driven Experiments," *Systems Engineering*, Andrew P. Sage, Editor, Vol. 2, No. 2.

[Ilgen and Hulin, 2000]. Daniel R. Ilgen and Charles L. Hulin, *Computational Modeling of Behavior in Organizations: The Third Scientific Discipline*, APA Books, Washington, DC.

[Kemple et al., 1997]. William G. Kemple, Jim Drake, David L. Kleinman, Elliot E. Entin, and Daniel Serfaty, "Experimental Evaluation of Alternative and Adaptive Architectures in Command and Control," *Proceedings of the 1997 Command and Control Research and Technology Symposium*, National Defense University, Washington, DC.