The Use of Simulation Models in Model Driven Experimentation

Holly A. H. Handley, Zainab R. Zaidi, and Alexander H. Levis*
System Architectures Laboratory, C3I Center
George Mason University
Fairfax, VA 22030

Abstract

In model driven or model based experimentation, the model of the experiment is a key component of the closed loop model of the process. The model is created through interaction with the team designing the experimental organizations as well as the team creating the experimental environment. Starting with preliminary descriptions, the model evolves as more specific details are available and influences the final experimental design.

The methodology used to design the model reflects both the types of design information available and the underlying hypothesis of the experiment. Experiments validating fixed types of structures or processes lead to a model designed with a Structured Analysis Design Technique which leads to an explicit but rigid model design. Experiments investigating adaptation require a more flexible model which can be created using an Object Oriented design approach. This leads to a more flexible, object view of the experimental design. Either approach leads to an appropriate set of models from which an executable model can be derived. The executable model is used to carry out simulations

In order to analyze the dynamic behavior of the model, an input scenario must be created based on the actual inputs that will be used in the experimental setting. When the model is stimulated with the scenario, its behavior can be observed and its performance measured on different criteria. Because it is a computer simulation, input parameters can be varied, constraints can be relaxed, and other variables (possibly) affecting the hypotheses can be explored to see their effect on the model and by inference the experiment. These results can then be made available to the design teams to influence further iterations of the design. Indeed, the model allows the consideration of many excursions, a situation that is not possible when the experiments include teams of humans.

After the experiment is conducted, model validation is carried out by comparing the model results to the actual experimental results. This is done by driving the model with the original scenario, but including the actual decisions made by the human subjects, or decision makers.

1.0 Introduction

The development of the experimental model is the second step in the process of model driven experimentation. After the hypothesis to be investigated has been defined, the development of the model occurs concurrently with the development of the decision making organizations and

* This research was sponsored by the Office of Naval Research under grant no. N00014-93-1-0912.

the experimental situation. These three areas of development evolve together, with early iterations of the model providing feedback for improvement to the other two areas. As progress is made on the organizational and environmental designs, this information is incorporated into the developing model and preliminary simulations conducted. The results from the simulations are then passed back to the design teams for evaluation. Improvements can then made to the previous designs. When the experimental organizations and environments are finalized, the model development is complete; it can now be used to conduct the pre experimental simulations.

The design of the model is directly related to the experimental hypothesis; the hypothesis defines the purpose of the model. This in turn affects the methodology chosen to design the model. For example, if the hypothesis is comparing different fixed organizational structures, the model must be designed in a way to represent the different candidate designs. If the hypothesis is evaluating an organization responding to a changing environment, the model must be designed in a way to respond to a variety of inputs. In some cases in may be more important *what* the model does, in other cases it may be more important *how* the model does it. The model must be designed befitting the hypothesis.

An appropriate model can only be created if the model designer interacts with the designers of the other parts of the experiment as the model represents the entire experimental design – the organization, the environment, and the performance measures. The model consists of the organizational design responding to stimuli from the environmental design and must record or measure the organization's response. These three facets are often being designed by separate groups using different tools. For example, the organizational designer may provide information in the form of hierarchical organization diagrams, Gantt charts of resource allocations, and tabular data concerning task requirements. Likewise, the environment designer may talk in terms of task types, geographic restrictions, and minimum force requirements. The performance measures may be in terms of time limits, task accuracy, or casualties. The model designer must be able to integrate and implement this varied information into a cohesive model that accurately represents the complete experimental design.

The primary reason for modeling is to use the results from the pre experimental model simulations to improve the experiment before it is conducted, by isolating the conditions that will generate the most pithy results – those that have the most substance and are in a well defined region. Using a model eliminates the guesswork surrounding the experimental design; a model can withstand multiple iterations of the same procedures with minor variations. The model can be used to explore the extreme boundaries of operation as well as the incrementally small changes associated with fine-tuning the ranges and values of variables and parameters in the experimental design. The model's behavior should be observable and its performance measurable using the criteria defined in the experimental design. These criteria for evaluation of the behavior and performance of the modeled system should be defined in the design stage of the model in order to ensure that these variables are indeed observable or that the necessary data to generate these measures is available for capture.

2.0 Modeling Methodologies

System engineering provides two distinct approaches for designing models, the Structured Analysis Design Technique and the Object Oriented approach [Levis, 1999]. Both methodologies lead to executable models of systems, but they take different paths and emphasize different aspects of the system. Models that require sequential processes and fixed structures are best implemented through the Structured Analysis approach. Models that respond to multiply occurring, independent events are better suited to an Object Oriented approach. Both approaches will be described and examples given based on the actual models from the Adaptive Architectures for Command and Control (A2C2) research program. Both models had similar purposes, however the different models stressed different aspects of the experimental design.

2.1 *Structured Analysis Design Technique*

The pre experimental models for the second A2C2 experiment (A2C2-2) [Handley et al., 1997] employed the Structured Analysis Design Technique [Marca and McGowan, 1988]. This methodology was chosen because the hypothesis compared two distinct, fixed structure organizational designs, termed "Traditional Architecture" (TA) and "Non-traditional Architecture" (NTA). Both organizations would be required to complete a predetermined, fixed task sequence in order to achieve the requisite mission. The mission consisted of conducting simultaneous troop landings on a northern and southern beach. The southern group proceeds along the south road to secure the airport. The northern group secures a nearby hill, then proceeds along the north road to the seaport. Along the way they encounter mines, tanks, and ground, sea, and air assaults. Because this mission can be viewed as a fixed process, or sequence of tasks, and because two distinct fixed organizational structures were being evaluated, a structured approach was appropriate to design the pre experimental models for this experimental design.[1]

The structured analysis approach starts by identifying the functions that the modeled system must perform. A functional decomposition is created, which is a hierarchical description of the functions the system performs. To completely specify the model design under this methodology, a set of four models is created based on the functional decomposition: an activity model, a data model, a rule model and a dynamics model. Each one of these inter-related models contains a different aspect of the complete model design. The activity model describes the processing of inputs from the environment into outputs to the environment. The associated data model describes the relationship between the data at the different stages of processing. The rule model describes the conditions that must be satisfied for the activities to take place. The dynamics model describes the states of the system and the transitions between them.

Two pre experimental models were created for this experiment, one representing each of the two proposed organizational architectures. For each of the two organizations under evaluation, the information available from the organizational design team was the number of decision-makers, the platforms they controlled and the command hierarchy. Platforms represent the deployable resources of the organization, for example a tank or a helicopter, as well as a troop battalion.

[1] The first A2C2 experiment (A2C2-1) also used a structured analysis approach. However, because that modeling effort was not completed prior to the experiment being conducted, it is not discussed here.

Both the TA and the NTA had six decision-makers but arranged in different command hierarchies with a different distribution of platforms as shown in Figs. 1 and 2. The information from the environmental team was the mapping of decision maker responsibility to the sequence of tasks necessary to support the mission. This information was extracted from the written description of the experimental environment which provides a situational assessment, including enemy and friendly forces, and the concept of operations, including task assignments and coordination instructions. The experimental design allowed the transfer of platforms from one decision maker to another. This only occurred if a decision-maker had responsibility for a task in which he did not have the proper platform, then a transfer was initiated through the chain of command. Since this occurs infrequently, the request and transfer for a specific platform for a specific task was reflected in the functional decomposition as specific functions and was embedded in the activity model.

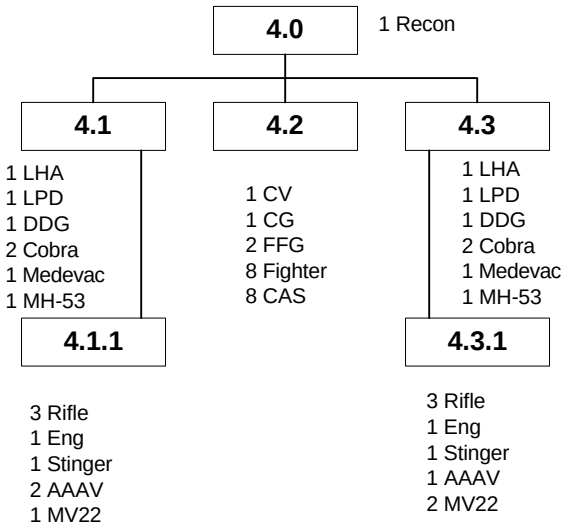

Figure 1: Traditional Architecture Organizational Structure

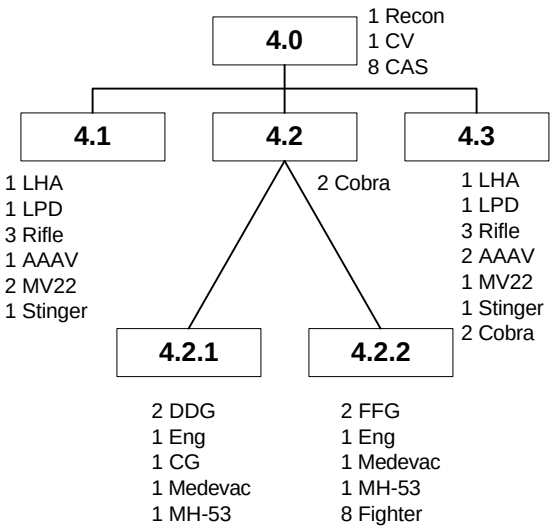

Figure 2: Nontraditional Architecture Organizational Structure

The functional decomposition for a model is based on the operational concept defined for the organization being modeled. The operational concept describes how the organization should carry out its mission; it may be that there are several possible ways to achieve a mission so one must be chosen and specified. For the two organizational designs under consideration, the operational concept was defined as the sequence of tasks indicated in the concept of operations as necessary to complete the mission. These mission tasks were decomposed into subtasks, which were in turn further decomposed. The decomposition continues until each element of the "tree" is distinct, there are no repetitions or duplications. An additional requirement for the lowest level of decomposition was that each function can be accomplished by a single decision-maker utilizing a single platform. There is some flexibility on the approach taken to perform the functional composition. In this case functions were decomposed first by geography and secondly in order to maintain sequential processing. Different functional decompositions are acceptable depending on the operational concept and the modeling objective. This functional decomposition consisted of 72 distinct functions. Fig. 3 shows the top-level decomposition and the A1 block, "Generate Orders", further decomposed to the second level decomposition. Each of the other three branches, "Attack Beaches and Hill", "Clear Roads and Advance", and "Secure Port and Airfield", are also further decomposed, but not shown here for clarity.

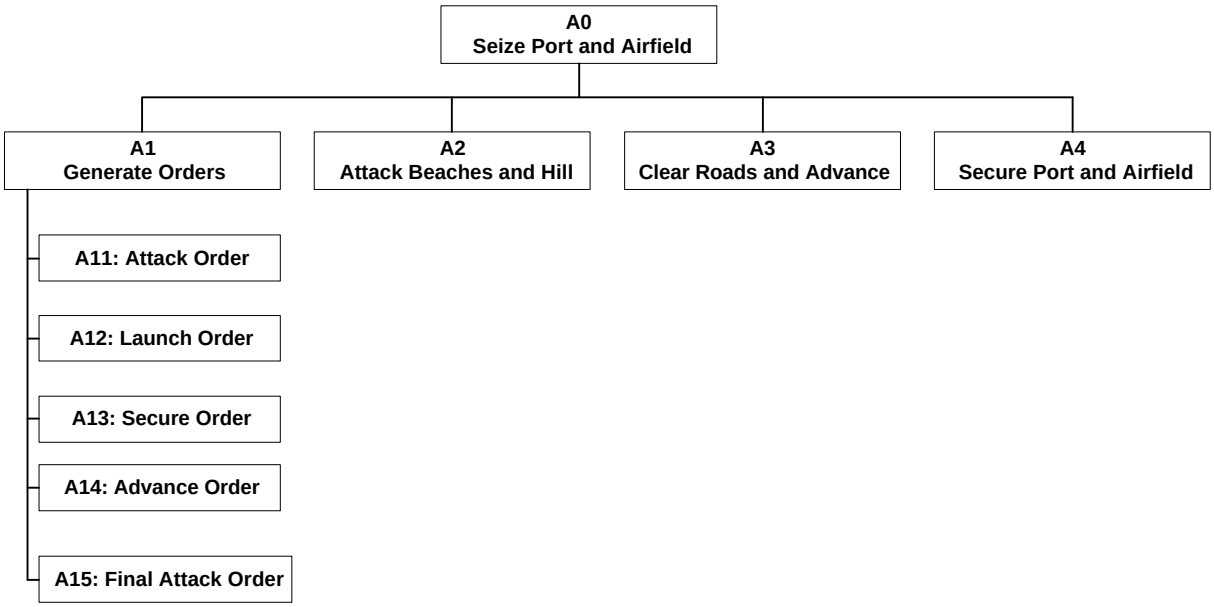

Figure 3: Functional Decomposition - A1 Branch

Once the functional decomposition is complete, the activity model takes the distinct functions and arranges them in a specific order, that when activated will achieve the defined mission. The activity model was created using IDEF0™. The blocks represent the functions, the input arrows on the left of the block are the data input to that function, the output arrows on the right of the block are the output data from the function. The input arrows on the top of the block are the control information for the function and the input arrows on the bottom of the block are the mechanisms necessary to perform the function. In this case, orders and enemy threats are represented as inputs, completed tasks are outputs, which become control information for future functions, and the decision-makers and the platforms are modeled as mechanisms. The offensive actions necessary to complete the mission are represented as the sequence of activities. The resulting activity model had 20 pages. The context page, shown in Fig. 4, indicates the model's objective, "Seize Port and Airfield of Sfax". "Sfax" is the hypothetical city in which this engagement is taking place. The first level decomposition is shown in Fig. 5, which shows the three main sequential activities: "Attack Beaches and Hill", "Clear Roads and Advance" and finally "Secure Port and Airfield". The "Generate Orders" box represents the coordination for each major activity. Note that these are also the top level functions of the functional decomposition: there is a direct relationship between the activity model and the functional decomposition.

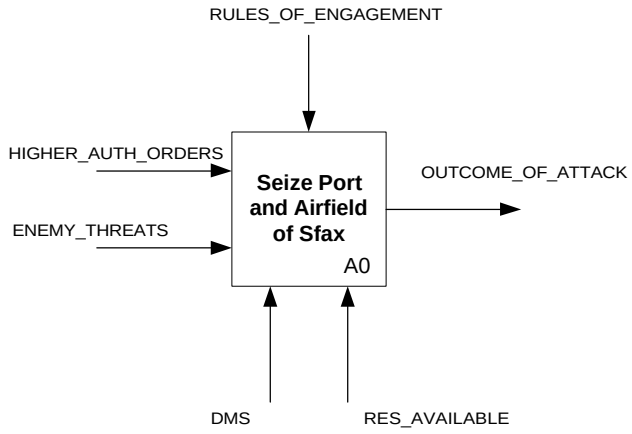

Figure 4: Activity Model – Top Level Context Diagram

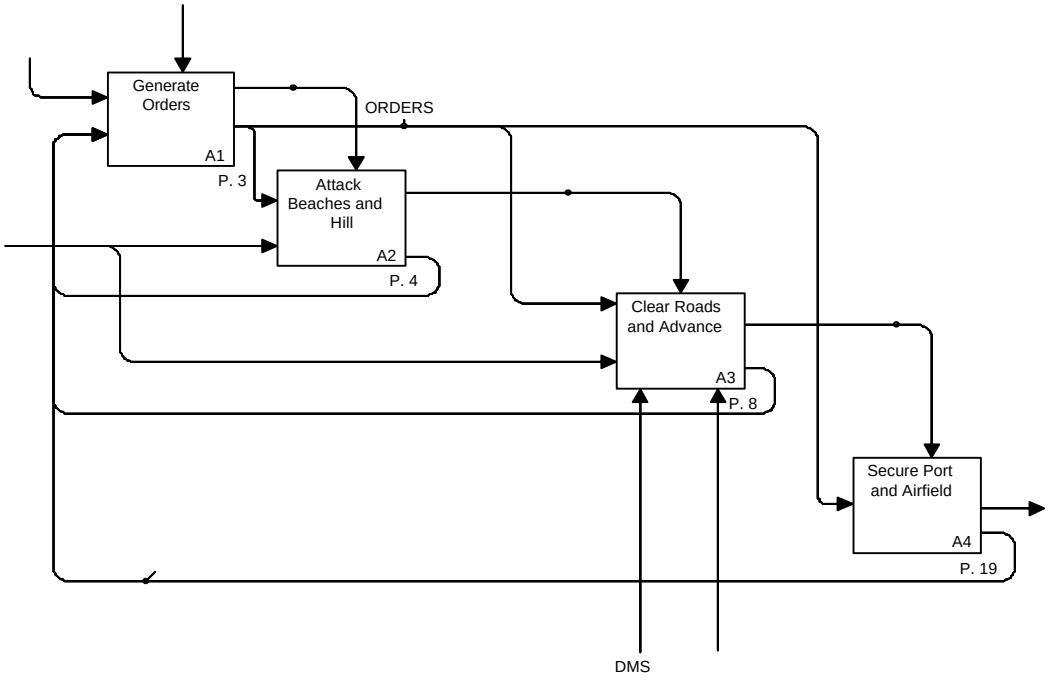

Figure 5: Activity Model – A0 Page of IDEF0 Diagram

The data model identifies the relationships between the information elements identified in the activity model, the attributes they may have, and the allowable values. Each data flow between the functions identified in the activity model becomes an entity in the data model, represented as a box. The attributes of the data are listed in the box. The relationships are indicated on the association lines between the boxes. The one page data model was created using IDEF1X™ and is shown in Fig. 6. The rule model defines the logical operation of each leaf function in the activity model; they may be activation, interaction, or coordination rules. Fig. 7 shows the block of rules for the "Generate Orders" activity, which was implemented using structured English. The fourth model, the dynamics model captures the expected behavior of the system. It is

described using a state transition diagram. Since the operational concept is based on a task sequence of two distinct paths, the state of the system corresponds to the current place in the sequence of events, with states transitioned on the giving of orders as shown in Fig. 8.

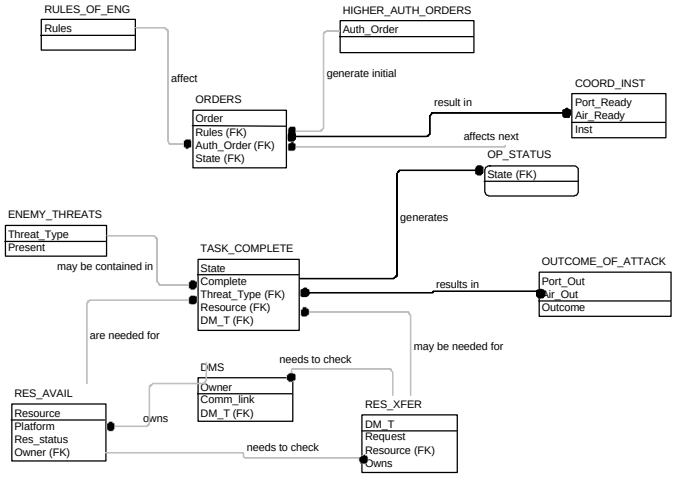

Figure 6: Data Model – IDEF1X Diagram

[A1] Generate Orders

[A11] Generate Attack Order:: IF Auth_Order = Attack AND Rules=Plan
THEN Order.A1=Clear_Beaches AND Complete.A11=Yes
[A12] Generate Launch Order:: IF State.A214=Bmines_Cleared AND Rules=Plan
THEN Order.A1=Launch_Marines
[A13] Generate Secure Order:: IF State.A225=Marines_Launched AND Rules=Plan
THEN Order.A1=Secure_Beach_Hill
[A14] Generate Advance Order:: IF State.A235=Beaches_Hill_Secure AND Rules=Plan
THEN Order.A1=Advance_Roads
[A15] Generate Final Attack Order:: IF State.A315=North_Road_Cleared
OR State.A325=South_Road_Cleared AND Rules=Plan
THEN Order.A1=Coord_Attack

Figure 7: Rule Model – Generate Orders Functions

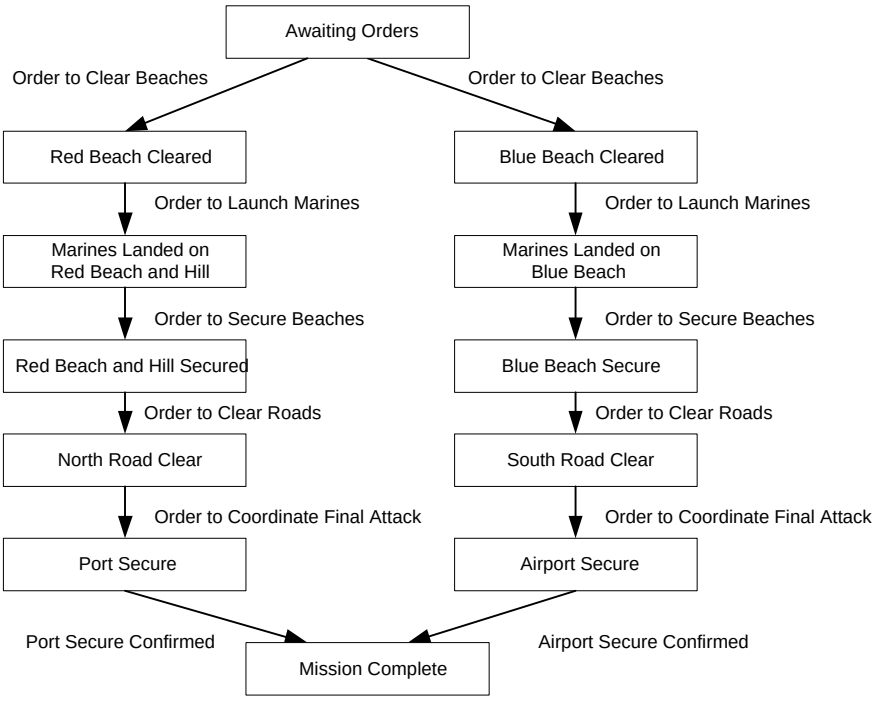

Figure 8: Dynamic Model – State Transition Diagram

A system dictionary was created which accompanies the set of four models. Each individual model generates a dictionary of elements and definitions for that model. The system dictionary integrates all the model dictionaries to provide the sole reference for definitions and cross-reference of all elements of the system. While the system dictionary provides a consistency and completeness check for the set of models, model concordance insures that all models are in a representative state of agreement. Model concordance is achieved through a process of creating hypertext links between all objects in the static models. [Levis et al., 1999] This was completed in order to achieve a cohesive set of models.

Up to this point in the model design, the different organizational structures have not played a role in the design. Now two distinct models were created, by assigning a decision-maker and platform, where appropriate, to each leaf function of the functional decomposition. If the decision-maker owns the platform he needs to accomplish his assigned task, none of the additional resource transfer function activities were retained for that mission task. If he did not own that resource, the number of transfer resource functions retained depended on how many steps through the chain of command were necessary to complete the transfer. For example, in the mission task “Remove wounded from North Road” in the TA model, two steps were necessary to transfer the Medevac platform from the owner decision-maker to the responsible decision-maker. In the NTA model four steps were necessary. In most cases, no resource transfer was necessary at all. By customizing the activity models appropriately for each architecture, the static model designs were now complete.

Some limitations of this model and approach should be noted. By choosing a structured analysis approach, the functional decomposition becomes the foundation of the model. However, in the case where similar tasks were being performed in different branches of the tree, the functional decomposition becomes clumsy, in the sense that more and more adjectives need to be added to distinguish the functions. For example the geographic qualifiers "north road" or "south road" were added to the function "clear road mines" to distinguish them. Secondly, because only two instances occurred where the transfer of resources was necessary, embedding them within the activity model was not difficult. However, if transferring resources were a much more common occurrence, it would complicate the model significantly to embed those in both the functional decomposition and the process model. Thirdly, in this model, the tasks are grouped geographically. For example all the north road functions appear on one page. This supports the sequential processing of tasks, however, the decision-makers responsible for the tasks are interwoven, making it difficult to gauge the activity level of the individual decision-makers. Fourth, noticeably missing from this model, is the random, enemy threats that occur independently of the task sequence of the mission, for example, the submarine strikes at sea. With this type of approach, it was not feasible to include them in the model. Lastly, because only two variants of the architecture were under consideration, it was not difficult to create two separate models. But for a larger number of candidate architectures, or for more design iterations, this would prove difficult.

2.2 *Object Oriented Approach*

The third experiment of the A2C2 program (A2C2-3) expanded on the results of the second experiment and considered a hypothesis that evaluated different organizational structures with both a varying number of decision makers and number of platforms. The number of decision-makers was no longer restricted to six, there could be less, and the number of platforms could be either "full" (36 platforms) or "reduced" (20 platforms.) Moreover, at the time of the hypothesis definition, the number of variant architectures or design iterations was not fixed. With the possibility of many variants and iterations, it was not feasible to anticipate creating a new set of models with each design change. Therefore, one of the modeling objectives became to create a re-configurable model, that is a model that could be easily modified to incorporate new organizational designs with different numbers of decision makers and platform parameters. While the mission and operational concept remained the same for this experiment as in the previous ones, with minor variations such as including a bridge task, much more information was made available from the environmental design team concerning the number and nature of enemy threats. Most of these threats would occur randomly and independently of the task sequence representing the mission objectives. Based on this information, the structured approach used in A2C2-2 was no longer appropriate. An Object Oriented approach was taken to design the pre experimental model for A2C2-3[2], creating a flexible and re-configurable model [Handley et al., 1998].

In order to create the model using an object oriented approach, the structure of the system was defined first instead of the process, or task sequence. By examining the basic characteristics of the system, three abstract entities or object classes were defined: Decision Maker (DM),

[2] The fourth A2C2 experiment, (A2C2-4), was an extension of the third experiment. It used the same re-configurable model designed for A2C2-3 with minor modifications.

Platform, and Task. These are general descriptions that represent a whole class of objects. The description of these object classes is shown in Fig. 9. Each object is given an identifier based on the type of object it is. For example the DM objects are DM0, DM1, etc. Each object also has attributes or characteristics specific to that object class. For example the DM object class has attributes Skills, Owned Platforms, and Assigned Tasks. Giving the attributes values defines the different instances of a particular object class. For example to instantiate the Platform instance whose ID is SAT, the fixed attributes would be set as follows: resources = 6 IDES, Owner platform = CV, Trigger Eliminated = No, and Velocity = 0.7. The instances have initial conditions that fix the variable attributes until they are changed in order to accomplish a particular task. For example the Platform instances are given an initial Current_Location, indicated by the environmental design, and the Distance_to_Travel is set to zero until the platform is assigned to a target. Then the target location is used to calculate the distance the platform must travel from its current location to the target's location. The method or operation that the instance implements is also identified. The Platform object class has one method, Travel_to_Task.

	Decision Maker Object Class	**Task Object Class**	**Platform Object Class**
ID	DM_ID	TASK_ID	PLT_ID
Attributes: Fixed	Skills	Value	Resources
	Owned Platform	Time	Superplatform
	Assigned Tasks	Type	Trigger
		Location	Velocity
		Resource Required	
		Skill Required	
Attributes: Variable			Current_Location
			Distance_to_Travel
Methods	Launch Platforms	Execute Task	Travel_to_Task
	Enable Tasks		

Figure 9: Object Class Definitions

Conceptually, the model operates as follows: when a task arrives, the DM mapped to that task recognizes it. The DM then launches the appropriate platform by checking what platform is applied to that task. When a platform is launched, it begins to travel to the location of the task. Travel time to the task is included in the model based on the individual platform's speed and its previous location. Once the platform has arrived at the task location, the DM enables the platform and the task is executed for the corresponding duration. Once the duration time has expired, the platform is then available to be assigned to another task. In the case of coordinated tasks, multiple platforms are required to complete a task. The DM responsible for the task identifies the other DMs who own the necessary platforms to complete the task. He sends out messages over communication channels to the other DMs to check the availability of the other platforms. Once the platforms are all available and present at the task location, the responsible DM sends out the message to enable the platforms, and the task is then executed.

As described above, the interaction of these object classes is how the system operates to complete the mission objectives. Following the Object Modeling Technique of Rumbaugh, [1991], three views of the system are necessary in order to complete the model design. The

object view describes the relationships between the object classes. A functional view must be defined for each method in the object class definitions; it shows data transformation through the use of data flow diagrams. The dynamic view depicts the behavioral aspects of the system through the use of state transition diagrams, similar to the dynamics model of the structured analysis approach.

The object view shown in Fig. 10 depicts the three interacting object classes, DM, Platform, and Task, and their relationships. The Scenario Driver is also shown which provides the type, timing, and ordering of the tasks to which the organization must respond. An object definition file describes in detail all the allowable instances of each object class. For each method defined in the object view, a functional view must be created. For example, Fig. 11 shows the functional view for the method Execute_Task. It shows the objects that provide data, how the data are transformed by the method, and the output data to other objects. Finally, the event trace shown in Fig. 12 depicts the behavioral aspects of the model. It shows two instances of a DM interacting to launch two platforms to accomplish a task.

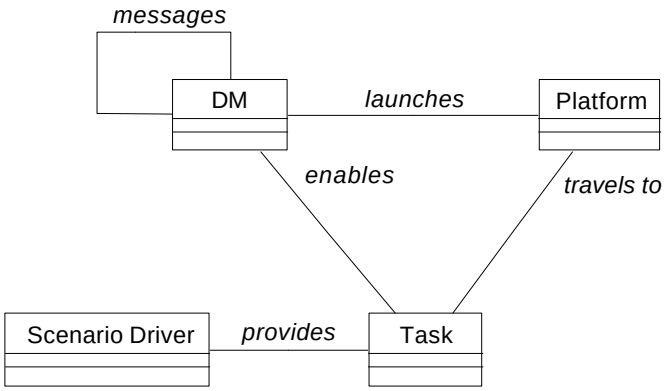

Figure 10: Object View

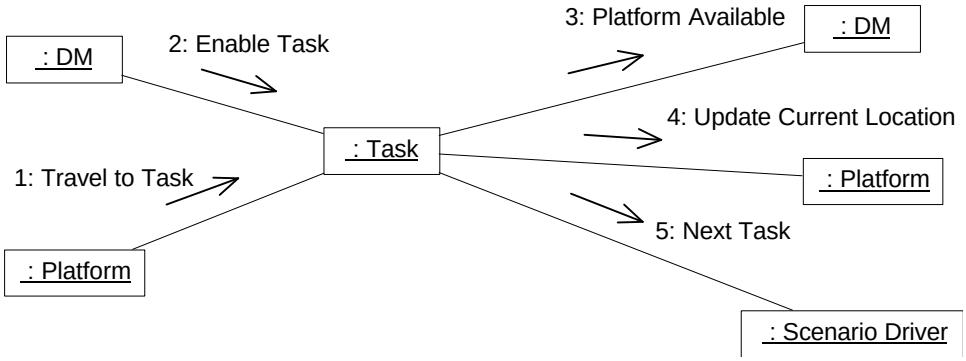

Figure 11: Functional View – Execute Task Method

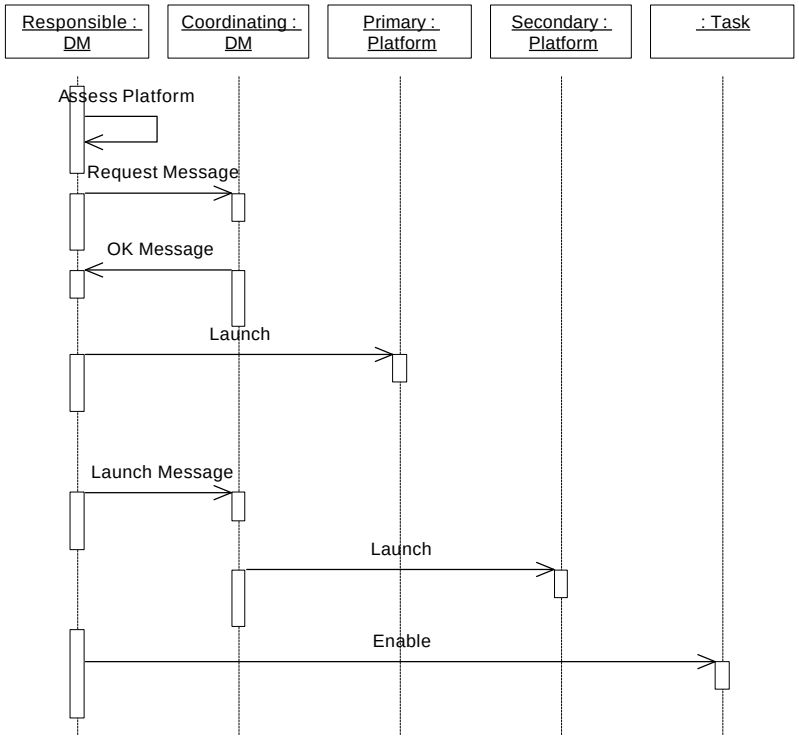

Figure 12: Dynamic View – Coordinating Decision Makers

Up to this point in the design, the variants of the different organizational architectures have not been considered. After several design iterations, three candidate organizational structures were identified for the experiment. The decision maker hierarchies with the major platform distributions are shown in Figs. 13 - 15. Recall that this model was designed to be re-configurable, meaning one model was designed and initialized to a different organization prior to simulation. Changing the mapping, the relationship between the DM, the platforms he owns, and the tasks he is responsible for, customizes the model. A sample of this mapping information is shown in Fig. 16. Once the mapping information is recorded, the static design of the model is complete.

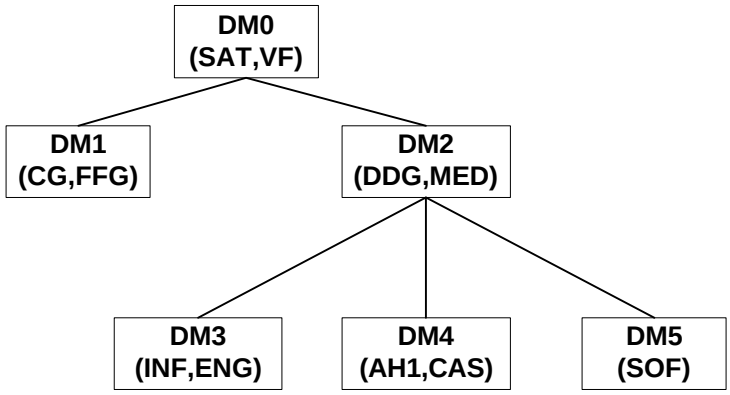

Fig. 13: Organizational Architecture A0-6

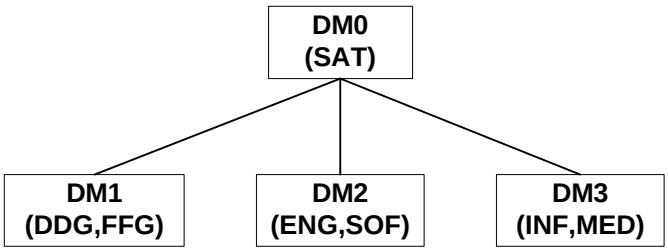

Figure 14: Organizational Architecture A1-4

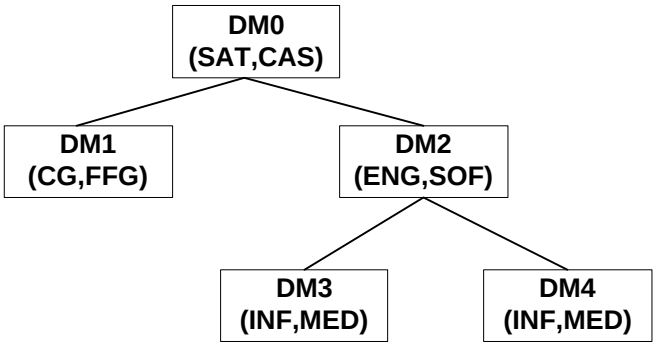

Figure 15: Organizational Architecture A2-5

UConn ID	DDD ID	PLATFM	A0-6	A1-4	A2-5
731	[T10-PZ5] (D:AC:Air-Sea)	DDG-4	DM2	DM1	DM1
731	[T17-PZ5] (D:SC:Submarine)	FFG-3	DM1	DM1	DM1
731	[T7-PZ5] (D:GC:Silkworm)	CAS-13	DM0	DM0	DM0
741	[T15-PZ5] (D:GC:Dummy Silk)	SAT-8	DM0	DM0	DM0
731	[T16-PZ5] (D:SC:Patrol Boat)	DDG-4	DM2	DM1	DM1
741	[T19-PZ5] (D:SC:Neutral Patrol)	SAT-8	DM0	DM0	DM0
741	[T18-PZ5] (D:SC:Neutral Sea)	SAT-8	DM0	DM0	DM0
741	[T21-PZ5] (D:AC:Neutral Air)	SAT-8	DM0	DM0	DM0

Figure 16: Sample Task – Platform – DM Mapping for North Fleet Tasks

Some limitations of this model and approach should also be noted. Because the mission tasks, as well as the threat tasks, were included as part of the scenario driver, the two types of tasks are indistinguishable and this model does not prioritize one type of task over the other, regardless of the consequences. Secondly, the model was designed to complete all tasks received, regardless of how long they are delayed before they are executed. Tasks can be delayed when the required platform is busy with another task. Likewise, the model does not stop processing tasks when the mission tasks are complete, and delayed tasks may still be active long after it is appropriate. It's worth noting that the resource transfer function, which was an issue in A2C2-2, [Handley et al.,

1997] did not occur in A2C2-3. Each decision-maker was only assigned tasks for which he had the appropriate resources. Ironically, the object-oriented model could have incorporated that function quite easily.

3.0 Creating an Executable Model

Once the design of the model has been completed, an executable model is created. An executable model can be created regardless of the design approach used. The model design at this point is static, it is a description of the model and how it should behave. In order to execute the model, it must be created in a dynamic environment. Colored Petri nets provide a modeling and simulation environment in which an executable model can be created. The executable model combines all the information in the various static models or views into one model that can illustrate dynamic behavior: the flow of data, the conditions under which functions are performed, and the order in which they are performed. Behavior analysis and performance evaluation can then be carried out using scenarios consistent with the operational concept.

Colored Petri nets [Jensen, 1992] are a generalization of ordinary Petri Nets [Murata, 1989]. Ordinary Petri nets are bipartite directed graphs. There are two sets of nodes: places denoted by a circle node and transitions modeled by a bar node. The arcs or connectors that connect these nodes are directed and fixed. They can only connect a place to a transition or a transition to a place. A Petri net also contains tokens. Tokens are depicted graphically by indistinguishable dots and reside in places. A marking of a Petri Net is a mapping that assigns a non negative integer, representing the number of tokens, to each place. A transition is enabled by a marking, if and only if all of its input places contain at least one token, since each input arc represents a single connection between the place an the transition. An enabled transition can fire. When the firing takes place, a new marking is obtained by removing a token from each input place and adding a token to each output place. The dynamical behavior of the system is embedded in the changing of the markings.

In Colored Petri nets (CPN), the tokens are no longer indistinguishable; they now carry attributes or colors. Tokens of a specific color can only reside in places that have the same color set associated with them. The requirements to fire a transition are now specified through arc inscriptions; each input arc inscription specifies the number and type of tokens that need to be in the place for the transition to be enabled. Likewise, output arc inscriptions indicate what tokens will be generated in an output place when the transition fires. Code segments can also be associated with a transition; these are blocks of code that execute when the transition fires, representing the functionality of the transition. A global declaration node of the Colored Petri net contains definitions of all color sets and variables with their domains for the model.

Although there is not an automated process for creating an executable model using CPN from the static designs, some general guidance is available and there are associations between the different static models and the components of the executable model. For designs created with the structured analysis approach, the CPN most often follows the activity model. Each activity is converted to a transition and each arrow becomes a place with the color set of the data label. The information in the data model is used to specify the color sets and their respective domains, while the rules in the rule model result in arc inscriptions. The top-level page of the A2C2-2

model is shown in Fig. 17. The transitions represent the main mission tasks and the places represent the data. In order for the task transition to fire, there must be a token in each of the input data places. After the transition fires, these tokens are removed, and tokens are produced in the output data places. This model is hierarchical, similar to the activity model. Sub pages, such as shown in Fig. 18, show the complete sequence of functions for each mission task. Figs. 19 and 20 show the effect of the different organizational structures requiring more embedded functions for the resource transfer in the Remove_Wounded_from_North_Road task.

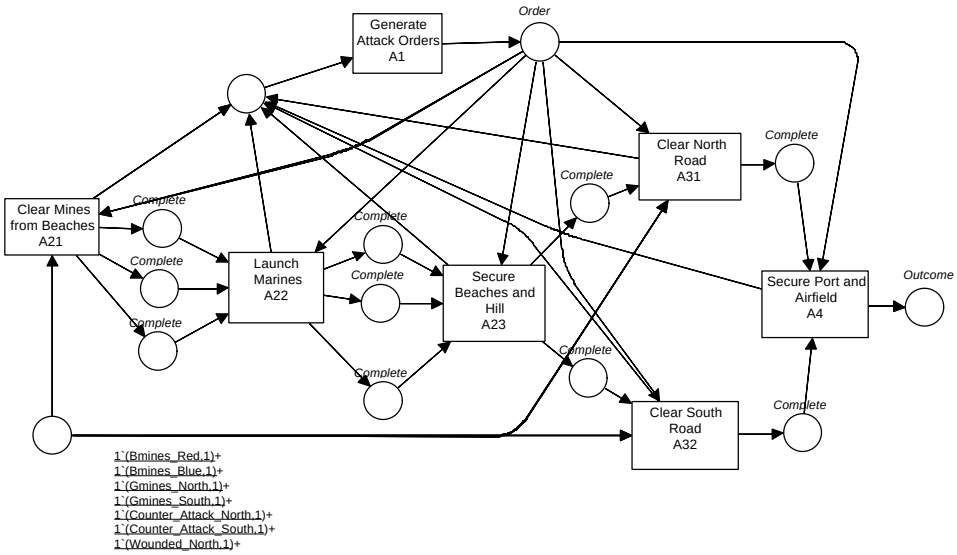

Figure 17: Top Level Executable Model – A2C2-2

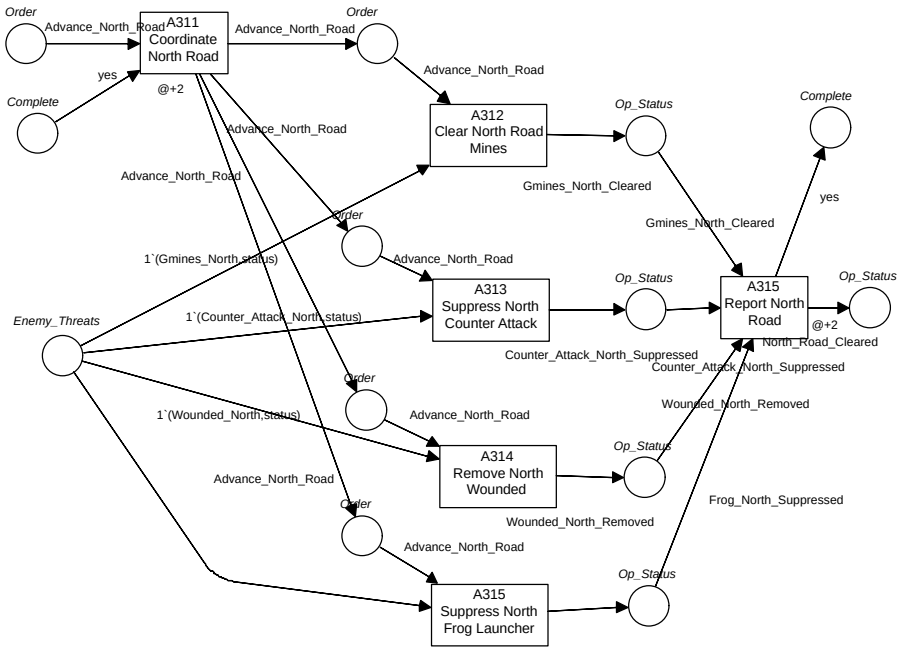

Figure 18: Sub Page – Clear North Road

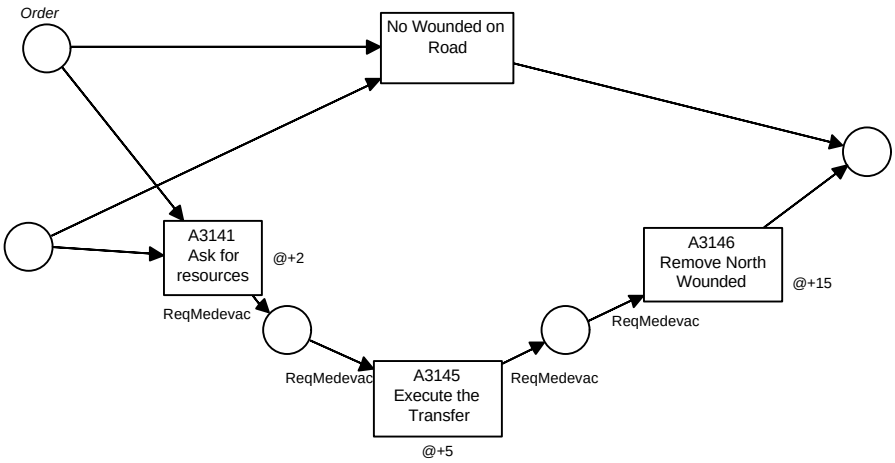

Figure 19: Sub Page – Remove Wounded from North Road, TA Variant

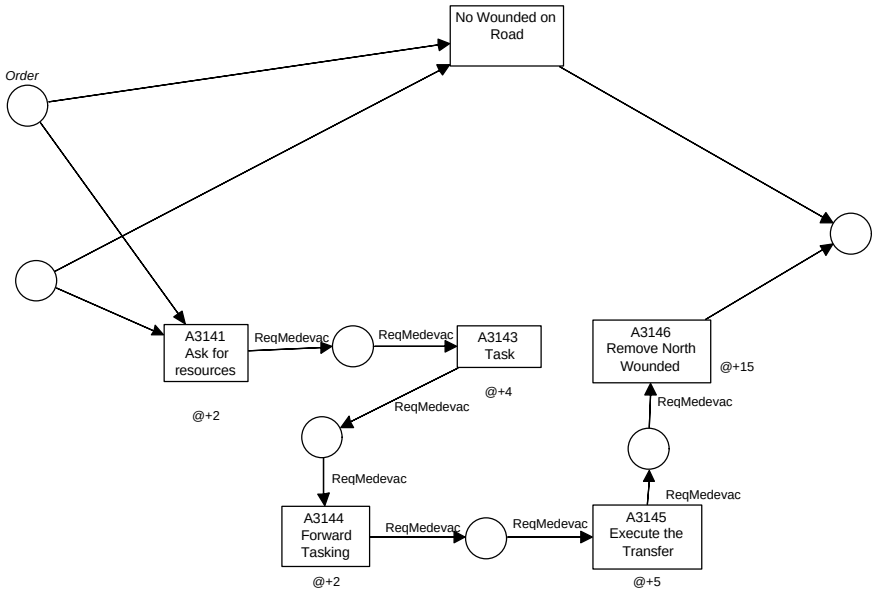

Figure 20: Sub Page – Remove Wounded from North Road, NTA Variant

For object oriented designs using OMT, the CPN most often follows the object view. The executable model for A2C2-3 contains a top-level page representing the object view and the scenario driver as shown in Fig. 21. In this case each object class has become a transition and the associations have become places. There are sub pages for the scenario driver, the task, the platform, and the organization object classes. The organization, Fig. 22, has a further sub-page, the decision-maker, Fig. 23. Although the organization page contains instances of six decision-makers, organizations with fewer than six can be created by placing “dummy” markings on the unnecessary DMs. The DM objects were enumerated in this case, as opposed to using an object class, in order to provide the customization necessary to reconfigure the organization. Each DM object on this page has a distinct DM Data place. The initial marking on this place customizes the organization by indicating each decision maker’s platforms and tasks for this configuration.

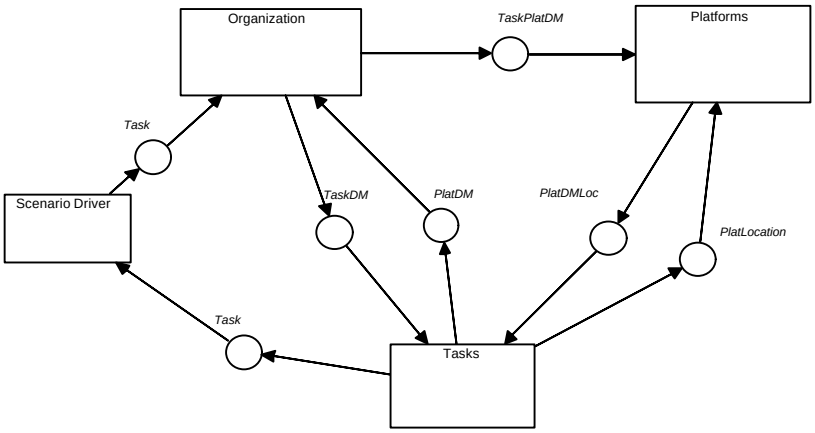

Figure 21: Top Level Executable Model – A2C2-3

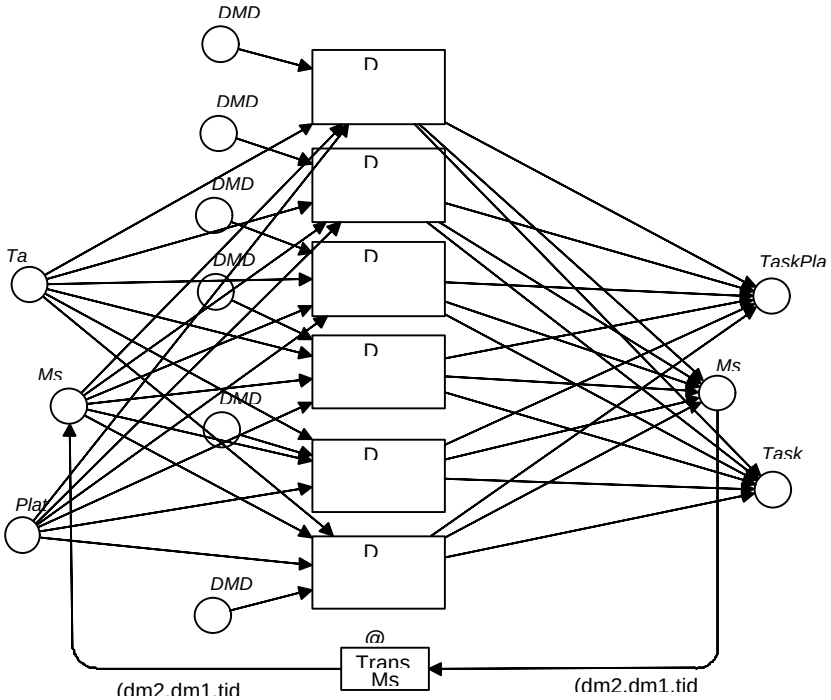

Figure 22: Sub Page – Organization

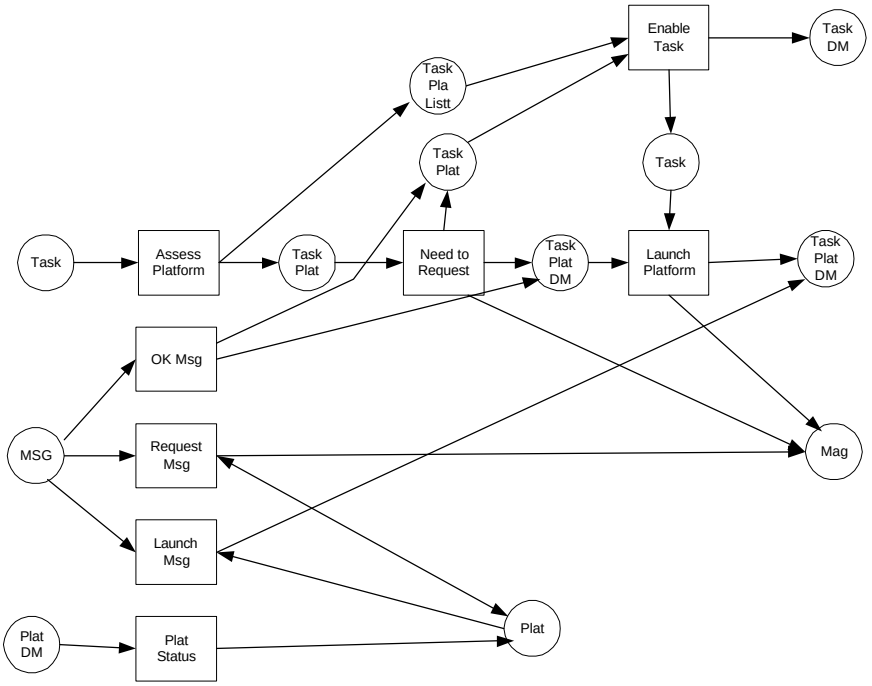

Figure 23: Sub Page – Decision Maker

4.0 Simulating the Model

Once the executable model has been completed, a set of stimuli for the model to must be created. This is termed a scenario or scenario driver. It represents the type, frequency and arrival time of the inputs the organization under study must respond to in the experimental environment. For the A2C2 series of experiments, the human subjects were arranged in the different organizational architectures and responded to a set of inputs from a simulated military environment. The DDD-III Simulator, located at the Naval Postgraduate School, is an interactive computer environment that can be configured to represent different missions. The monitor displays a map of the area of conflict, including the terrain, and also the friendly and enemy platforms involved in the mission. The screen also has a dialog box for messaging among the decision makers. The decision makers communicate and manipulate platforms to complete the mission, much like a video game. The input file which configures the simulator for a particular mission can be used to extract the information necessary to create a scenario driver for the executable model.

Creating a scenario driver for a model depends on the boundary that was established between the organization and its environment when the model was designed. In the A2C2-2 model, for example, much of the experimental environment was included within the model that was developed; the order, type and geographic location of tasks were embedded in the model itself. Consequently, the line between the organization and the environment was often obscure. On the other hand, the model developed for A2C2-3 incorporated no environmental information. The modeled system represented the organizational structures generated by the system design team. [Levchuk et al., 1999] It has the ability to complete tasks, but the tasks themselves are not captured within the framework of the model. The task information must be generated external to the model with a scenario driver.

Because so much of the environmental information for the A2C2-2 experiment was captured in the model, the scenario driver used to stimulate the model is very simple. The scenario driver initializes the state of the artillery and frog threats to be “yes” or “no” and then initiates the model execution by generating the initial attack order. All the rest of the defensive (enemy threats) and offensive actions are embedded in the model. However, for the model for A2C2-3 experiment, a complex input scenario was created that represented the occurrence of independent enemy threats and the mission task sequencing. Although the environmental design team did not release the exact input file for the simulator, a list of the types of threats, the number and geographic region of each threat occurrence was available as well as restrictions or relationships between them. For example, some tasks occur multiple times at different time intervals through out the scenario, some tasks must precede other tasks as depicted in the mission task graph, and some tasks trigger other tasks. Using a scenario designer with domain experience, an input sequence of 175 tasks was created based on this information and distributed over the time allowed to complete the mission. This scenario driver is shown in Fig. 24. Through the use of the scenario driver the expected simulator events were represented in the executable model simulation.

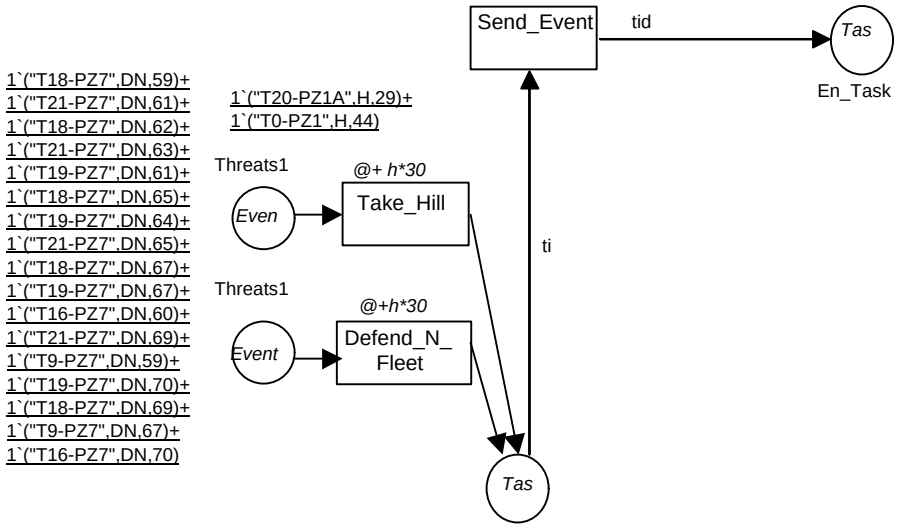

Figure 24: Scenario Driver - A2C2-3 – North Fleet Portion

Once the scenario driver has been created the model can be executed. The first simulations are logical in nature; they insure that the model is behaving correctly and as expected. These are followed by baseline simulations. Baseline simulations are simulations run with variable parameters set at default or design values, the values assumed during the design of the model, organizations and environment. The results of these simulations provide a basis for comparison to future simulations where the parameters are varied.

The executable model for A2C2-2 was evaluated for behavior but not for performance. This means that only baseline simulations were conducted and the results of the two different organizational structures were compared to each other but not ranked. The two variants were simulated with the simple scenario driver described above and the sequence of events that occurred was recorded and compared to a scripted key thread. Key threads are sequences of activities initiated by a certain event and traced through the modeled process to an organizational

response. The sequence of tasks is first identified in the activity modeled so that a key thread is generated and then traced in the executable model to do the comparison. These key threads are used to verify that the executable model correctly represents the activity model and that this behavior is what is desired in the experiment. If timing information is included in the model, which was added to the A2C2-2 model later, delays between the variants completing the scenario can be compared. A key thread used for A2C2-2 is shown in Fig. 25. The resulting traces from the executable models are shown in Figs. 26 and 27.

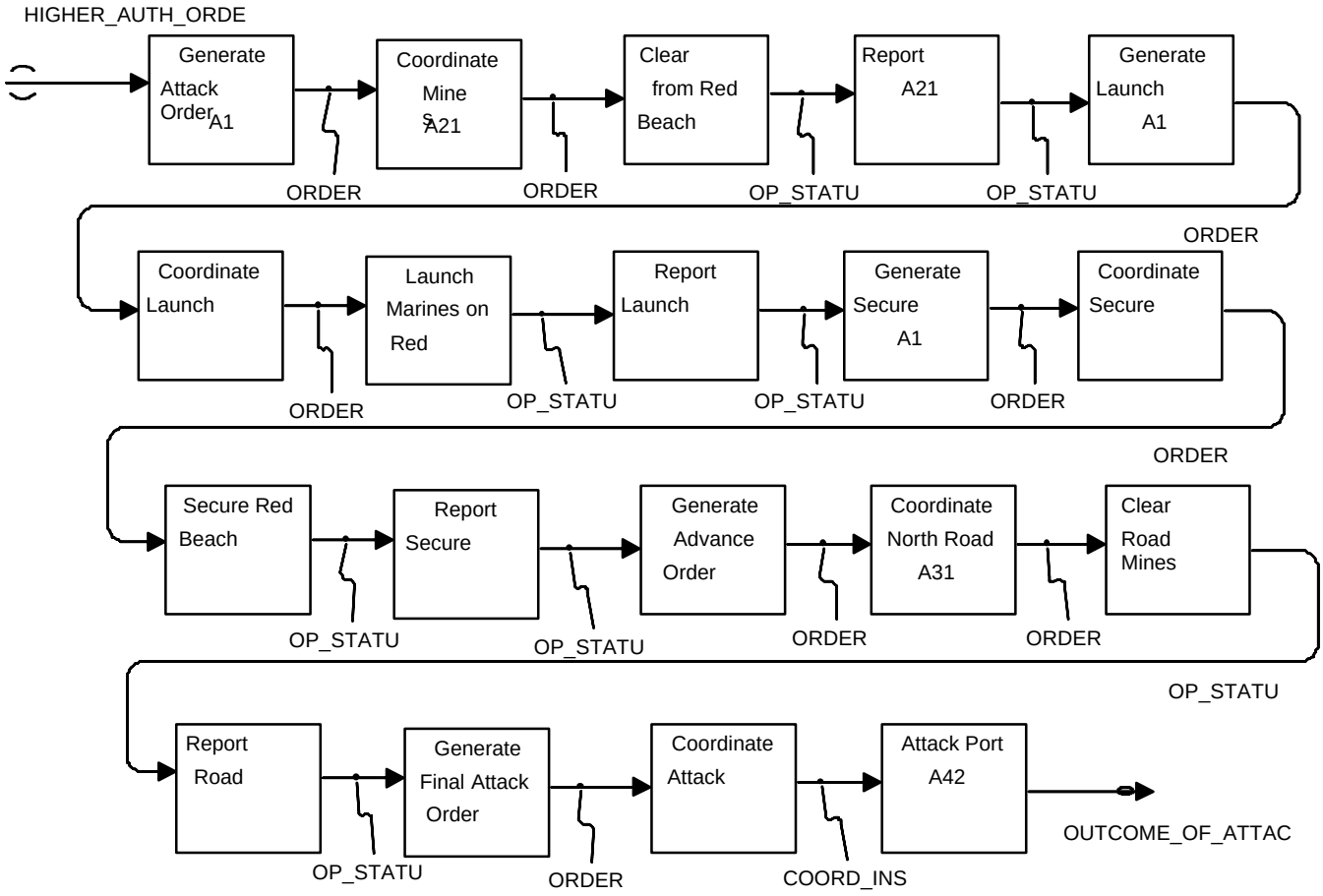

Figure 25: Key Thread – A2C2-2

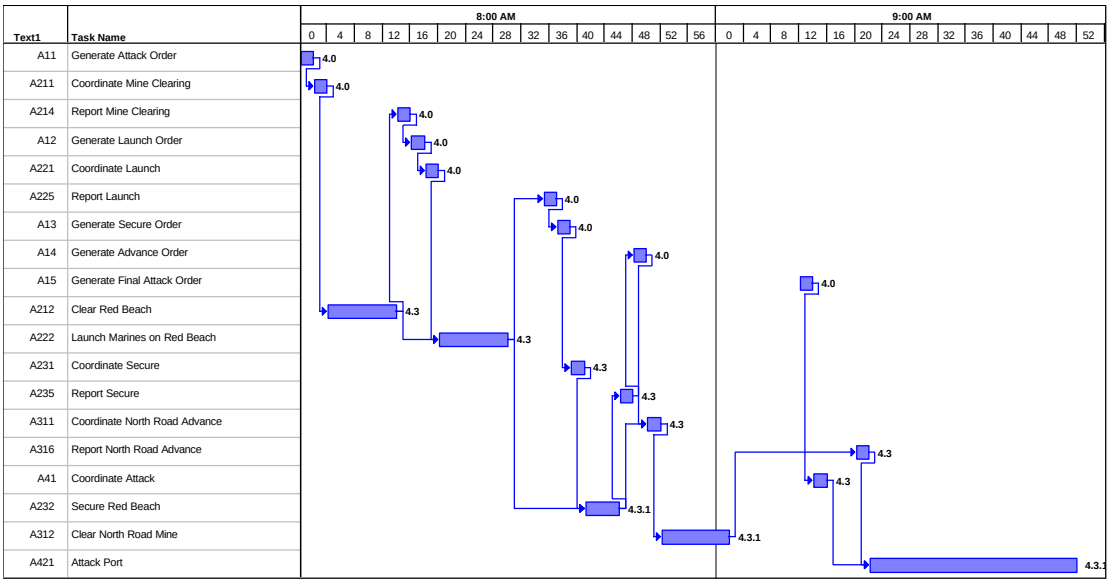

Figure 26: Results – TA

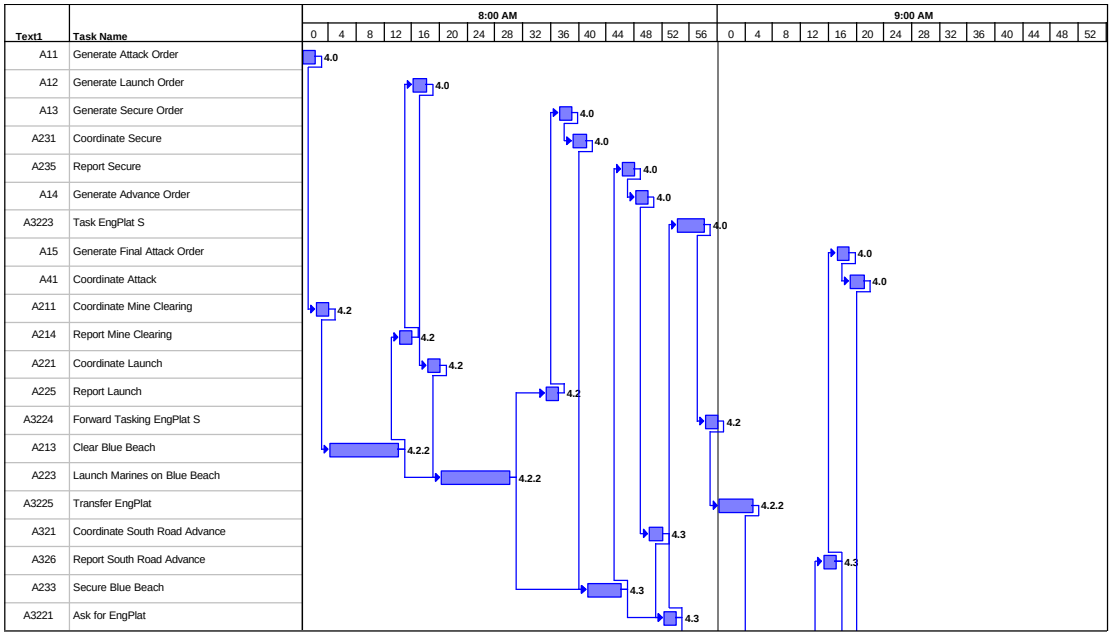

Figure 27: Results - NTA

The model for A2C2-3 included more realistic timing information based on the simulator input files. When this model was simulated for behavioral analysis, evaluations of each modeled organization were made based on overall processing time, processing of individual tasks, instances of resource contentions, and delays associated with decision maker coordination. This information, shown in Fig. 28 was displayed in a Gantt chart for review by the organizational designers. This feedback resulted in a second iteration of organizational design, dropping some previous organizations that did not behave as anticipated and adding new ones. When the variant organizations were finalized, the baseline simulations were conducted. The results from these baseline simulations are shown in Fig. 29. The primary performance measure for the pre experimental simulations became the completion time of the final mission tasks: securing the airport and securing the seaport.

Latency: All Tasks	**A0-6**	**A1-4**	**A2-5**
Average	27.67	79.24	83.76
Standard Deviation	45.44	130.03	143.92
MAX	313	792	793

Figure 28: Preliminary Results – A2C2-3

Mission Tasks Completion Time	**A0-6**	**A1-4**	**A2-5**
Hill (T0-PZ1)	464	514	514
North Beach (T4-PZ2)	407	447	447
South Beach (T4-PZ3)	419	417	417
Bridge (T27-PZ9)	546	660	574
Airport (T1-PZ4)	921	868	1090
Seaport (T2-PZ0)	870	835	909
Final	1116	1389	1390

Figure 29: Baseline Results – A2C2-3

The main contribution of pre experimental models is the ability to explore variations in either the organization or the environment and gauge the effect on the experimental design. This allows the “fine-tuning” of the experiment design before the subject experiment is commenced, and insures that the experiment is conducted in a manner that will lead to the most meaningful results. The parameters to be explored depend on the hypothesis itself and in the uncertainties within the design of the specific experiment.

The model for the A2C2-3 experiment was used extensively for pre experimental simulations. When the scenario driver was created for the model, it included the ability to vary the inter-arrival time, ordering and number of each type of task. In this way the effect of changes to the experimental environment could be explored. Although the ordering and number of particular tasks had only a minimal effect on the system, changing the inter-arrival time of the tasks did have an effect on performance. There were no time anchors in this scenario, meaning no task had to occur at a fixed time, however the tasks did have a fixed duration once they were initiated. A

factor, h, was included in the design of the input scenario that scaled the inter-arrival time of the tasks. As the value of h was increased, representing more time between tasks, the performance of the different organizations converged. As the value of h was decreased, indicating that tasks were coming faster, the performance among architectures showed differences. In some cases, organizations could not finish the mission at small values of h. Since the objective of this experiment was to illustrate differences between the architectures, a value of h was chosen for the experiment where the architectures could finish the mission, yet still show variation. These results are shown in Fig. 30.

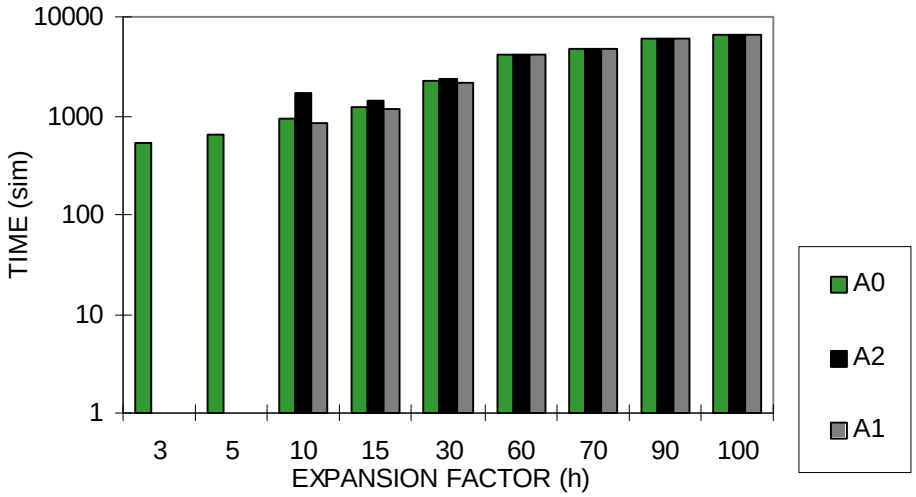

Figure 30: Varying Tempo Results

The other parameter that was explored with the A2C2-3 model was the number of communication resources. In this experiment all of the decision-makers were on one network; all decision-makers were allowed to communicate with each other, they were not restricted to the command hierarchy. Communication among decision makers increased when they needed to perform coordinated tasks. The number of communication resources represents the number of messages that can be exchanged between decision makers simultaneously. In organizations performing a greater number of coordinated tasks, the number of communication resources had an effect on performance, especially under increased tempo of operations. By varying the amount of communication resources the effect of communication resources on performance and the effect of coordination on performance could be explored. This data provided the feedback to the experimental design on the communication structure to be implemented in the experiment.

5.0 Model Validation

The pre experimental modeling effort is used to define the experimental design. It incorporates information from all the design groups involved in the experiment and provides feedback on the evolving designs as the simulations are conducted. When the model is finalized and the final pre experimental simulations conducted, the experimental design is concluded based on the results of these simulations indicating the appropriate settings and regions for experimentation. The experiment can then be conducted using the human subjects. After the subject experiment is

complete, the results can now be used to help evaluate and redefine the pre experimental model. Two issues need to be investigated: first the predictability of the model and second the accuracy of the modeling assumptions. The two are interrelated; if the assumptions used to design the model are inaccurate, the model is less likely to be predictive.

It often happens that due to unforeseeable circumstances the experiment is not conducted as quite expected. This may be due to equipment failures, personnel problems, or even the weather. In this case the pre experimental modeling results may not be predictive of the actual results. A review of the actual experimental situation can be used to modify the prior modeling assumptions and changes made to the model to reflect the experiment as it was actually conducted. The model can then be re-simulated with these changes to validate that it can correctly represent the actual experimental conditions and outcomes.
If further discrepancies between the validated model data and the experimental data remain this indicates there are more subtle differences between the model and the experiment. As these are resolved hidden variables are often uncovered that are unexpectedly affecting the experimental design. In a series of experiments, such as the A2C2 program, this validated post experimental model now becomes the starting point for the pre experimental model for the next experiment.

Before model validation could be conducted for the A2C2-3 experiment, some post processing of the subject experiment data was necessary. As it happened, some of the organizational architectures used in the experiment were not included in the pre experimental modeling; data from those trials was ignored. Secondly, since the model was designed to complete all tasks in the scenario, only data from teams that also completed all tasks in the experiment were used. Thirdly, due to technical difficulties, one team's data from all three of its trials was unreadable. This left eight of 28 result files available for data comparison. The post processing of this data resulted in a table of the mission tasks' completion times for use in the model validation process.

Once the experimental results were in a form similar to the model output data, a comparison of the two sets of data could be made for an initial assessment in the predictability of the pre experimental model. The results showed that the experiment was actually being conducted at a different tempo of operations than planned. That is, the simulator was providing inputs at a longer inter-arrival time then had been indicated by the pre experimental modeling. This slower tempo resulted in much less differentiation between the performance of the different organizational structures than anticipated. This was correctly predicted by the model when simulated at this slow tempo. However, some organizations did not complete the scenario in the allotted time, which was unexpected at the slow tempo. Further analysis of the results showed that these teams spent much more time on the threat tasks than on the mission tasks. This differentiating of the importance of tasks was not well represented in the model.

A discrepancy in the modeling assumptions and the actual experiment occurred in the ordering of tasks in the scenario driver. In the subject experiment, the DDD simulator used two slightly different scenarios, both of which were slightly different from the scenario used to simulate the model. These scenarios were made available after the experiment, so that two new scenario drivers could be created that more closely represented the scenarios used in the experiment. Also, a last minute change to the experimental design changed the communication structure of two of the organizations from a one-network configuration (all decision makers are able to

communicate with each other) to a two-network configuration, with one common decision-maker as the link between the networks. The model was revised to incorporate this restriction in communication for the specified organizations.

The models were then re-simulated with the revised scenario drivers, the revised communication scheme, and the slower tempo. The model results were now comparable to the subject experiment results. The results indicated that the revised scenario drivers, which represented a different ordering of the same tasks, had very little impact on the performance, however the tempo of the tasks, regardless of the task order has a much bigger impact. The communication structure results are not clear. Due to the nature of the model, the simulated communications between the decision makers were restricted to the terse, electronic messages available in the DDD simulator. However, in the actual experiment, subjects were allowed to communicate verbally, to broadcast, which was not represented in the model. The proper method of communication in the experimental design is still under debate.

6.0 Conclusion

The use of simulation models in model driven experimentation has been described. The model is a key component of the experimental design process as it represents the culmination of design information from all aspects of the experiment and feedback from the model simulations can be used to improve the complete experimental design. The modeling approach taken for a pre experimental model is dependent on the both the hypotheses of the experiment and the types of information available to the model designer. Both structured analysis and object oriented modeling approaches lead to executable models that can be used to provide feedback to the design teams regarding the dynamic aspects of the experimental design. Open communication between all the design teams is a necessary ingredient for a successful pre experimental modeling effort and the best indicator of a successful experiment. Post experiment model validation insures that the modeling assumptions are consistent with the subject experimental setting and that the model output is predictive of the experimental output. The validated model is now appropriate to use to begin the next cycle of pre experimental modeling for follow on experiments.

Experiments with teams of humans involve many variables and are hard to control fully, if the tasks are to be meaningful and appropriate for the domain of application. The detailed description of the modeling process and the findings illustrate the types of problems that are encountered and possible approaches for handling them.

7.0 References

[Handley et. al., 1997] Handley, Holly A. H. ,Didier M. Perdu, Insub Shin, and Alexander H. Levis. “Architectural Modeling of the A2C2 Second Experiment,” Technical Report GMU/C3I-183-R, March 1997.

[Handley et. al., 1998] Handley, Holly A. H., Zainab R. Zaidi, and Alexander H. Levis. "Pre Experimental Modeling of Adaptive Organizational Architectures," *Proceedings of the 1998 Command and Control Research and Technology Symposium*, June 29-July 1, 1998, Naval Postgraduate School, Monterey, California.

[Jensen, 1992] Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Springer-Verlag, Berlin, Germany, 1992.

[Levchuk et al., 1999] Yuri N. Levchuk, Krishna R. Pattipati, David L. Kleinman. "Analytic Model Driven Organizational Design and Experimentation in Adaptive Command and Control," *Systems Engineering*, Vol. 2,xxx

[Levis, 1999] Alexander H. Levis. "System Architectures," *Handbook on Systems Engineering and Management*, A.P. Sage and W.B. Rouse, Editors, Wiley, NY, 1999.

[Levis et al., 1999] Alexander H. Levis, Lee Wagenhals, Robert F. Phelps, "Architecting Information Systems," Report GMU/C3I-165-R, March 1999.

[Marca and McGowan, 1988] David A. Marca and Clement L. McGowan. *SADT: Structured Analysis and Design Technique*, McGraw-Hill Book Company, 1988.

[Murata, 1989] Tadao Murata. "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, Vol. 77, No. 4, pp. 541-579, 1989.

[Rumbaugh, 1991] James Rumbaugh. *Object-Oriented Modeling and Design*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1991.