A Software Environment for the Design of Organizational Structures[*]

Yuriy Shlapak
Jie Luo
Georgiy M. Levchuk
Fang Tu
Krishna R. Pattipati

Dept. of Electrical and Systems Engineering
University of Connecticut
Storrs, CT 06269-3157
Tel./Fax: (860) 486-2890/5585
E-mail: krishna@engr.uconn.edu

Abstract

This paper presents a software environment for adaptive organizational design, with focus on synthesizing Joint Task Force (JTF) C2 architectures subject to organizational constraints (e.g., the availability of resources and/or DMs, the distribution of DMs' expertise, etc.). Currently, the design environment includes software modules for: (1) mission modeling to extract task dependency graphs; (2) mission planning to allocate resources to tasks; (3) hierarchical clustering algorithms for grouping resources into decision-maker (DM) nodes, and (4) building an organizational hierarchy. In addition, basic modules for dynamic adaptation of organizational strategies and structures in the face of changing mission environment and/or resources are being added. The organizational design environment presented in this paper enables an analyst to synthesize robust organizational structures and evaluate their performances. The software tool allows an analyst to decompose the process of organizational design into a sequence of stages and visualize the design process. The software also allows a user to input parameters and constraints in a natural way at various stages of the design process, making it possible to design organizational structures with desired attributes (e.g., speed of command, workload, team coordination).

1. **Introduction**

A growing number of problems, ranging from agile manufacturing systems to joint task force missions, require a coordinated effort of a large group of individuals controlling a variety of resources. In an uncertain mission environment under time pressure, the ability of an organization to coordinate information, resources, decisions and actions among its agents has a significant effect on organizational performance. Consequently, a proper balance among information acquisition, designation of decision hierarchy, resource allocation, and action coordination, in short, *a proper organizational design*, is critical to superior organizational

*This work was supported by the Office of Naval Research under contracts #N00014-93-1-0793, #N00014-98-1-0465and #N00014-00-1-0101

performance of C3I organizations. Previous research has demonstrated that, while there is no one universally optimal organizational design ([Carley & Lin, 1995], [Papastavrou and Athans 1992], [Pete *et al.* 1998], [Reibman and Nolte 1987], [Tang *et al.* 1993]), there is a strong relationship between the specific structure of a task environment (i.e., mission) and the concomitant optimal organizational architecture ([Pete *et al.* 1998], [Levchuk *et al.* 1998, 1999a]). Subsequently, the ability of organizations to exhibit superior performance depends on the actual mission parameters (e.g., the type of mission, the attributes of corresponding tasks, etc.) and on the organizational constraints (e.g., the operational resources available, the capacity of communication channels, the level of training/expertise of personnel, etc.). To utilize this structural dependency, appropriate models of missions and organizations, capturing the actual mission parameters and organizational constraints, must be formulated prior to the design phase [Levchuk *et al.* 1997, 1998, 1999a]. For large missions and organizations, this leads to ever growing demand for automated tools to assist an analyst in modeling missions and generating optimal organizational architectures [Levchuk *et al.* 1999b]. Ideally, such tools must be sufficiently versatile to handle a wide scope of missions and to optimize organizational structures for different (user-defined) criteria. In addition, having such an automated design environment would allow one to examine human decision-making and coordination processes under different conditions, to generate new performance measures and hypotheses, to perform a comparative analysis of different (not necessarily optimal) organizational structures, and to test the robustness of a given organizational design.

Over the past few years, the Adaptive Architectures for Command and Control (A2C2) project has been developing a body of knowledge in current and future joint Command and Control (C2) and has been testing the resulting theories of adaptive C2 architectures through experimentation. The Adaptive Organizational Design (AOD) is one of the major components of A2C2. Given a C2 problem, the AOD focuses on the normative design of organizational structures by providing answers to three key architectural issues of an organization, i.e., task-DM structure ("who does what"), a Decision Maker (DM)-asset structure ("who controls what"), and an organizational structure ("who reports to whom"). When modeling a complex mission and designing the corresponding organization, the variety of mission dimensions (e.g., functional, geographical, terrain), together with the required depth of model granularity, determine the complexity of the design process. Our mission modeling and AOD methodology allow one to overcome the computational complexity by synthesizing an organizational structure via an iterative solution of a sequence of smaller and well-defined optimization problems [Levchuk *et al*. 1997]. The AOD methodology is implemented in an organizational design software environment to assist a user in representing complex missions and synthesizing the organizations. The component structure of our software environment allows an analyst to mix and match different optimization algorithms at different stages of the design process. Our mission modeling and a three-phase iterative organizational design process, first proposed in [Levchuk *et al.* 1997] and later enhanced in [Levchuk *et al.*, 1998], is graphically represented in Figure 1.

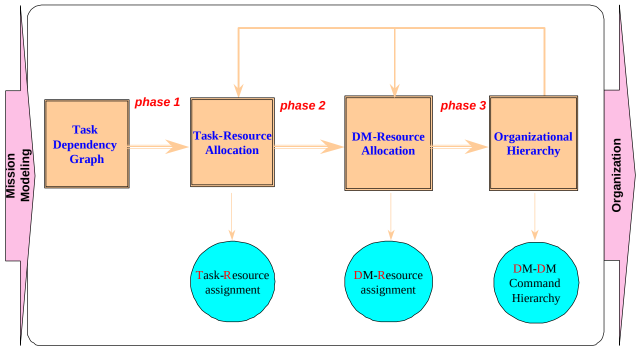


Figure 1. The 3-phase Organizational Design Process

The 3-phase design process of Figure 1 solves three distinct optimization sub-problems:

Phase I. Scheduling Phase
In this phase, an optimal *task-resource allocation* is established. It is defined in terms of a platform-to-task assignment matrix. The objective function (mission completion time **or** a combined objective function assembled from individual mission objectives such as the completion time, accuracy, workload, expended resources, external coordination, etc.) is minimized subject to assignment, resource availability, platform velocity and graph-related (such as precedence and synchronization) constraints.

Phase II. Clustering Phase
In this phase, an optimal *DM-resource allocation* is determined. It is referred to as DM-platform assignment matrix. The objective function (weighted sum of the maximum internal and external workloads **or** a combined objective function constructed from individual mission objectives such as the number of decision-makers, their expertise, available platforms and their resident resources, etc.) is minimized subject to assignment and DM workload constraints.

Phase III. Structural Optimization Phase
In this phase, an optimal *organizational hierarchy* is found. It is represented in the form of a directed tree with directed arcs specifying supported-supporting relations. The objective function (maximal hierarchy workload induced by direct (one-to-one) coordination and indirect coordination **or** a combined objective function gleaned from the identified mission objectives such as the number of communication links available for each DM, depth of organizational hierarchy, information flow, etc.) is minimized subject to the graph-related (information access and hierarchy structure) constraints.

On-line Adaptation Phase In case of an asset or a decision node failure, the application of a branch-and-bound method to the DM-task-platform allocation-preference matrix generates the next best assignments (the *new task-resource allocation strategy*). This method provides a quick and efficient search for adaptation options. The dynamic scheduling accounts for on-line changes without having to completely resolve the problem. If the newly obtained task-resource assignment matrix violates the organizational constraints, *Phases II* and *III* of the algorithm are used to generate the *new organizational structure*. In this case, *Phase II* is completed in an *evolutionary mode* (platform clusters are obtained by regrouping the old platform groups, rather than generating entirely new ones from scratch). Finally, if the process of generating a feasible organizational structure fails, the mission must be *aborted* (see [Levchuk *et al.* 1998] for details).

The rest of the paper is organized as follows. Section 2 presents an overview of the software environment. Section 3 shows the detail of each software module by applying our three-phase organizational design procedure to a simple mission example. Section 4 concludes with a summary and future extensions.

2. **Overview of Design Software Environment**

2.1 ***Design Principles***

In designing our software environment, we were guided by the following three design principles.

1. Software must be highly flexible in order to meet the great diversity of modern C2 missions.
2. The software must guide an analyst to describe the mission in a natural way so that the analyst can gain an understanding of the mission model and provide the required data.
3. The structure of the software must be modular with well-defined interfaces so that a multi-disciplinary team of algorithm and software developers can design and implement the component software with minimal coordination.

2.2 ***Structure Overview***

The overall structure of software environment, as currently implemented, is shown in Figure 2. Our modeling and design environment includes the following 7 key organizational components (Figure 2):

(1) *Asset/Resource Description*;
(2) *DM Structure Profiler;*
(3) *Mission Modeling;*
(4) *Performance Criteria/Measures;*
(5) *Schedule Generation;*
(6) *Resource Allocation;* and
(7) *Hierarchy Construction.*

The first three components are referred to as modeling components and the remaining are termed the optimization components. These are briefly described below.

2.2.1 ***Modeling Components***

The first three components of our design environment (*Asset/Resource Description, DM Structure Profiler,* and *Mission Modeling*) are devised to assist an analyst in developing mission models (of various complexity) and organizational constraints. These serve as inputs to our design process.

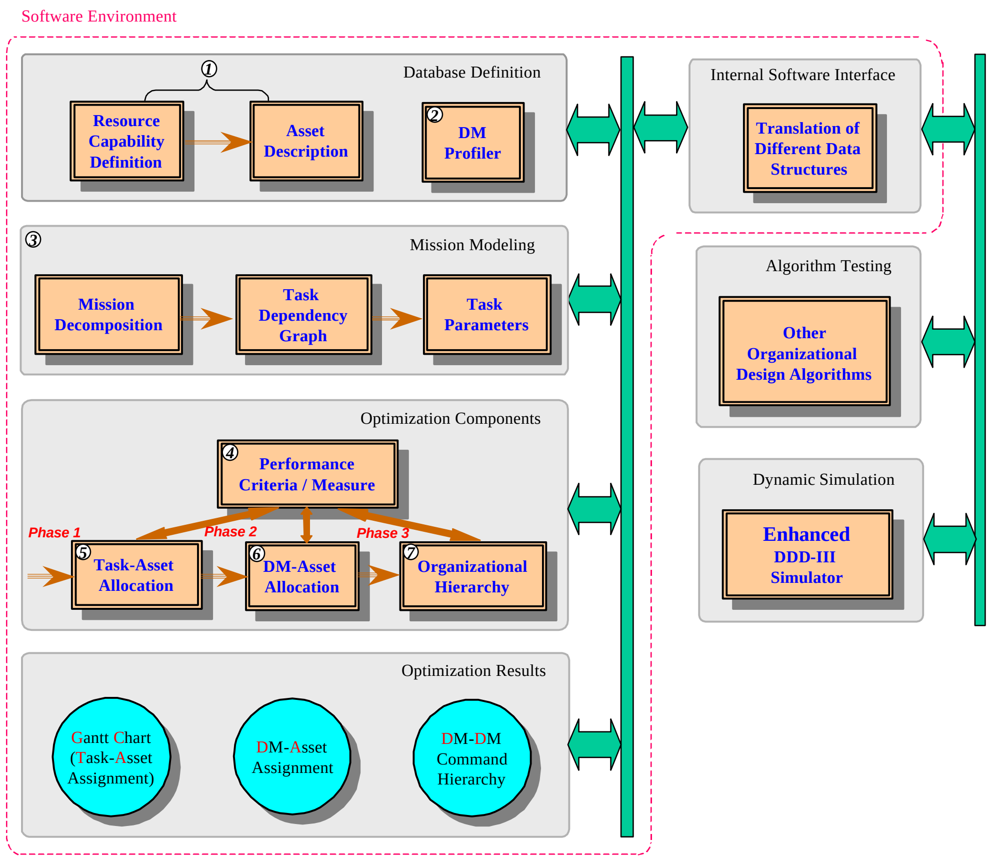


Figure 2. Structure and Interfaces of the Software Environment

Asset/Resource Description component enables the analyst to define the resource capability and information requirements types used to characterize the Organizational Assets. The asset data can be gleaned from subject matter experts or from a pre-defined database. *DM Structure Profiler* component lets a user specify design constraints such as workload and expertise constraints, and organizational hierarchy constraints (the number of communication links available for each DM, depth of organizational hierarchy, information flow, etc.). An analyst can build a database of assets as well as DMs for similar missions, since this data usually does not vary significantly from one mission to the next of a similar type.

Mission Modeling component facilitates an analyst to characterize the mission by decomposing it into tasks and specifying a *task dependency graph* together with other relevant task parameters. A *mission decomposition diagram* is built to represent a hierarchical structure among the mission tasks. Different decomposition techniques (e.g., functional decomposition, goal decomposition, terrain decomposition) generally result in different types of tasks representing the mission entities, and ultimately lead to different analytical models of the mission. The designer's choice of particular decomposition technique and model granularity must be consistent with the specific organizational design process and its supporting algorithms. Typically a *task dependency graph* represents the way in which a commander (or a planning cell) would plan to accomplish the mission, subject to his/her available assets.

2.2.2 *Optimization Components*

After the *Performance Criteria / Measures* component is used to define objective functions (i.e., speed of command, workload, coordination) for the design process, the last three components of our software environment (*Schedule Generation*, *Resource Allocation*, and *Hierarchy Construction*) allow an analyst to perform a step-by-step design of the organizational structure, while implementing, if desired, the user-defined design modifications at various stages of the design process to adjust the metrics of organizational performance (e.g., weights on objective function, workload distribution, etc.). These design optimization components present a step-by-step visualization of our organizational design process. Specifically, the *Schedule Generation* component produces the task-resource allocation schedule that corresponds to Phase-I of our organizational design algorithm. The *Resource Allocation* component (Phase II) defines DM functionality by grouping platforms and provides a balance between internal and external coordination. Finally, the *Hierarchy Construction* component (Phase III) derives organizational hierarchy to minimize the workload due to indirect external coordination induced by the hierarchy structure.

3. **Application to Organizational Design**

In this section, by applying our organizational design procedure to a simple mission operationalized in the Dynamic Distributed Decision-making (DDD-III) paradigm [Kleinman *et al*., 1996], we will illustrate the utility of our software tool and provide a brief description of the internal software modules.

3.1 *Example of a Simplified Mission*

A joint group of Navy, Marine and Air Force units, engineers and medical personnel are assigned to complete a military mission that includes capturing a seaport and airport to allow for the introduction of follow-on forces. There are two suitable landing beaches designated “Red Beach” in the north and “Blue Beach” in the south, with a road leading from the Red Beach to the seaport, and another road leading from the Blue Beach to the airport (see Figure 3). There is a small hill in the north, which is the commanding elevation in this small area. From intelligence sources, it is established that the beach is currently under the control of enemy. Near the beaches, enemy has set up a wide area of sea mines, which will prevent our ships from getting close to the beaches. Some SAM anti-aircraft missiles are found in both the northern and southern region of the beach. In addition, there is a bridge located in the northwest, which is important for the enemy ground forces to support the beach. This bridge needs to be destroyed before the arrival of enemy forces. The objective is to design an organizational structure, i.e., task-DM structure (“who does what”), a Decision Maker (DM)-asset structure (“who controls what”), and an organizational structure (“who reports to whom”), to successfully execute the mission.

3.2 *Organization Assets and DM Profile*

In the first part of the software, the Resource Capability Definition module and the Asset Description module enable an analyst to describe (or load from a database) the detailed parameters for all the assets used in the mission. This data includes the resource capability vector

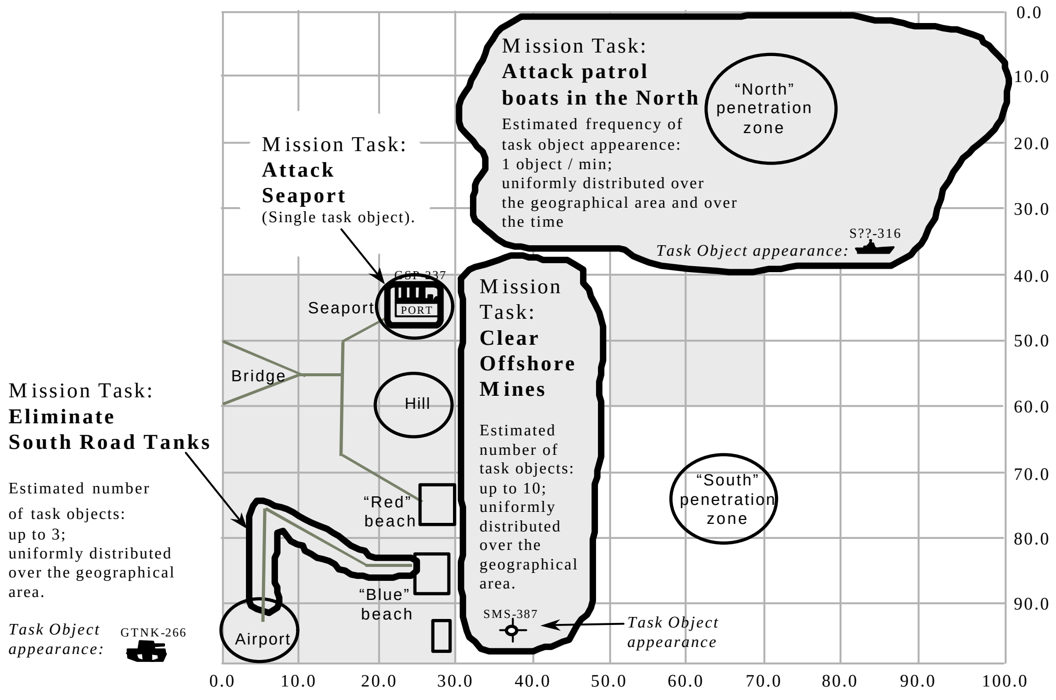
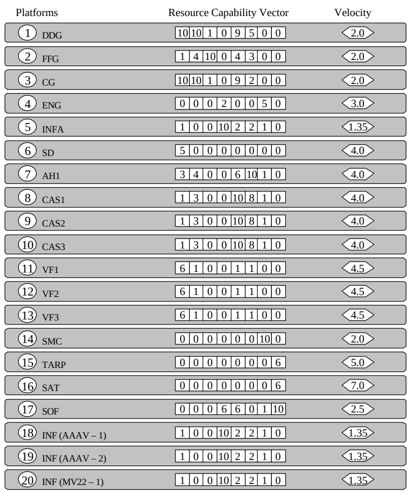

Mission Task:
Attack patrol boats in the North
Estimated frequency of task object appearence: 1 object / min; uniformly distributed over the geographical area and over the time
"North" penetration zone
S??-316
Task Object appearance:
Mission Task:
Attack Seaport
(Single task object).
GSP-237
PORT
Seaport
Bridge
Hill
Mission Task:
Clear Offshore Mines
Estimated number of task objects: up to 10; uniformly distributed over the geographical area.
SMS-387
Task Object appearance
"South" penetration zone
Mission Task:
Eliminate South Road Tanks
Estimated number of task objects: up to 3; uniformly distributed over the geographical area.
Task Object appearance:
GTNK-266
"Red" beach
"Blue" beach
Airport
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0

Figure 3. Scenario of the example mission

of each asset, speed, ranges and other attributes. Similarly, the profiles of available DMs will be input in the DM Profile module. The data for DMs includes the expertise in different functional areas, the limit of internal workload, the limit of external coordination, the communication limit the security level, etc. We call the first part of the software as database definition since these data items do not usually vary significantly from one mission to another of a similar type. In our simplified example, besides the resource capability vector, the only other parameter of an asset is the average speed, while we treat all DMs as equal. The following 8 resource requirements/capabilities are modeled in this simple example: AAW (Anti-Air Warfare), ASUW (Anti-Submarine Warfare), ASW (Anti-Sea Warfare), GASLT (Ground Assault), FIRE (Firing Squad), ARM (Armory), MINE (Mine Clearing), DES (Destroyer). The data for the assets and the screen for entering and displaying Asset Descriptions are shown in Table 1 and Figure 4, respectively.

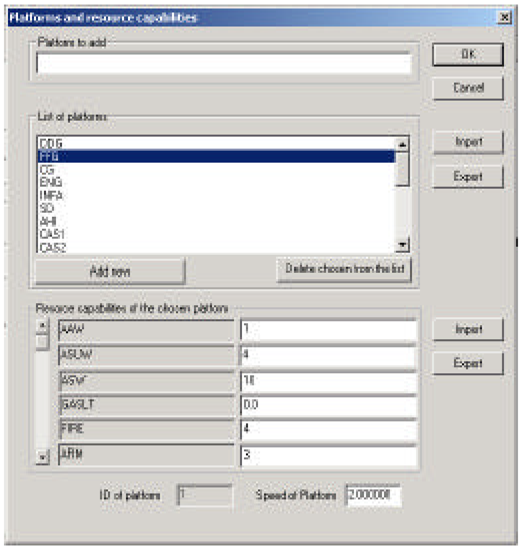
3.3 *Mission Modeling*

This component allows an analyst to model a mission hierarchically. In our software environment, there is no limit on the number of levels used to decompose a mission. One can start with a general view of the mission and decompose it into a set of sub-missions. These can be further decomposed into even simpler missions and eventually into individual tasks. This hierarchical view of missions is consistent with the Effects based Operations being advocated in the Global Experiments.



	Platforms	Resource Capability Vector								Velocity
1	DDG	10	10	1	0	9	5	0	0	2.0
2	FFG	1	4	10	0	4	3	0	0	2.0
3	CG	10	10	1	0	9	2	0	0	2.0
4	ENG	0	0	0	2	0	0	5	0	3.0
5	INFA	1	0	0	10	2	2	1	0	1.35
6	SD	5	0	0	0	0	0	0	0	4.0
7	AH1	3	4	0	0	6	10	1	0	4.0
8	CAS1	1	3	0	0	10	8	1	0	4.0
9	CAS2	1	3	0	0	10	8	1	0	4.0
10	CAS3	1	3	0	0	10	8	1	0	4.0
11	VF1	6	1	0	0	1	1	0	0	4.5
12	VF2	6	1	0	0	1	1	0	0	4.5
13	VF3	6	1	0	0	1	1	0	0	4.5
14	SMC	0	0	0	0	0	0	10	0	2.0
15	TARP	0	0	0	0	0	0	0	6	5.0
16	SAT	0	0	0	0	0	0	0	6	7.0
17	SOF	0	0	0	6	6	0	1	10	2.5
18	INF (AAAV – 1)	1	0	0	10	2	2	1	0	1.35
19	INF (AAAV – 2)	1	0	0	10	2	2	1	0	1.35
20	INF (MV22 – 1)	1	0	0	10	2	2	1	0	1.35

Table 1. Asset parameters


Platforms and resource capabilities
Platform to add
OK
Cancel
List of platforms
Import
Export
Add new
Delete chosen from the list
Resource capabilities of the chosen platform
Import
Export

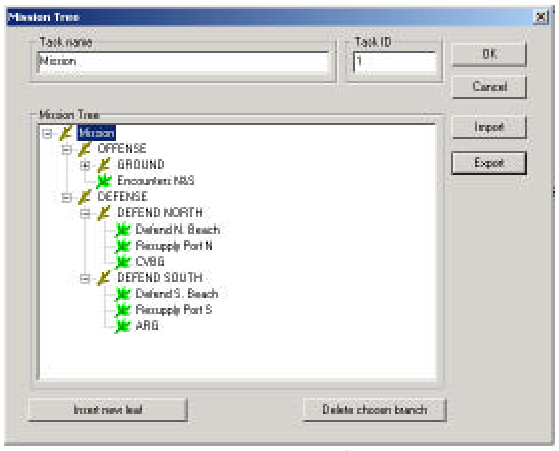
List of platforms: DDG, FFG, CG, ENG, INFA, SD, AHI, CAS1, CAS2

AAW	1
ASUW	4
ASW	10
GASLT	0.0
FIRE	4
ARM	3

ID of platform: 1 Speed of Platform: 2.000000

Figure 4. Asset Description

In the case of our simple mission, at the topmost level, the mission is divided into Offense and Defense. Each of these is further decomposed into North and South. In the case of Offense in the North, the sea mines may need to be cleared before attacking and landing on the beach. In addition, capturing the hill may be necessary to control the unique commanding elevation. In the same vein, Offense in the South may need to land on the beach, and then push forward to remove enemy ground forces along the road. A military commander may first come to such a brief plan without going into the details of a specific task, such as the specific process for removing the enemy ground force. Then, he may give a specific task to a DM of a lower level, who will continue to decompose the specific task to even smaller tasks and finally build a sub-plan of that task. On each level, the commander may also estimate the amount of resources that may be needed to complete the task. In addition, the processing time for a task will be estimated based on the expertise of a DM group assigned to prosecute the task. Evidently, the global mission will be decomposed into a tree structure. For our example, the tree structure is shown in Figure 6.



Task name: Mission Task ID: 1

```
Mission
  OFFENSE
    GROUND
    Encounters N&S
  DEFENSE
    DEFEND NORTH
      Defend N. Beach
      Resupply Port N
      CVBG
    DEFEND SOUTH
      Defend S. Beach
      Resupply Port S
      ARG
```

Figure 6. Mission Tree of the example

There is a dependency among the tasks. That is, some task can start only when some other set of tasks have been completed. For example, the forces may not be able to land on the beach unless the sea mines are removed. Before one can defend the beach, one needs to attack and control the beach. In general, tasks may have precedence structure among them, and some tasks can only start when they receive some positive sign of successful completion from other tasks. Once the mission tree is generated, the software will start the Task Dependency Graph module that guides the analyst to specify the dependencies among tasks. Since, in the mission tree structure, a task may contain a tree of sub-tasks, the task dependency graph can be built on any of the hierarchy levels. In our example, since there are not many elementary tasks, we will illustrate the task dependency graph at the bottom level in Figure 7.

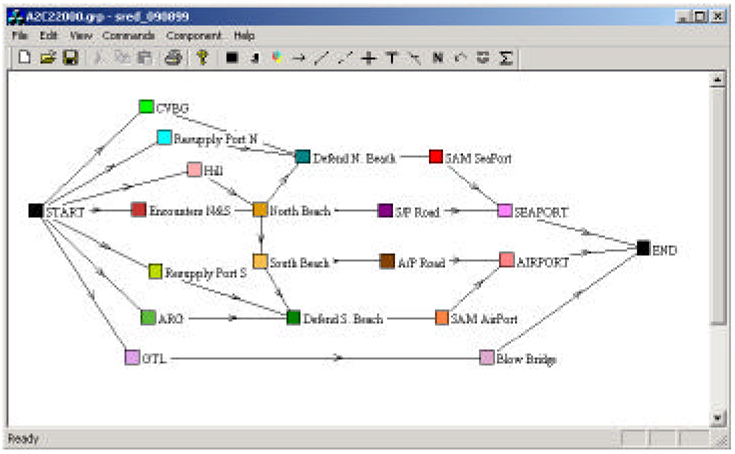


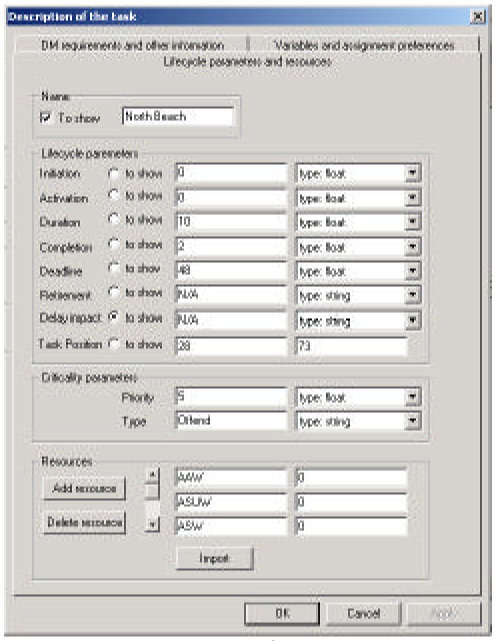
Figure 7. Task Dependency Graph

The elementary tasks are assigned different colors to help the analyst in identifying them. From the tree structure to the dependency graph, the analyst is guided from a broad scope to specifying individual tasks with well-defined dependency relations. At any of the above stages, the analyst can activate the Task Parameter module to input the parameters of a specific task. These include the resource capability requirements, which means the resources required to complete this task, task location and task processing time and so on. These parameters can be defined according to the military plan, or based on the responses from subject matter experts. An illustration of task parameters is shown in Figure 8. In addition, an analyst can choose to decompose the task at any stage.

The software components specifying the Organizational Assets and Mission provide a highly flexible means to model the great diversity of modern C2 missions. They also guide an analyst to describe the mission in a natural way so that the analyst can gain an understanding of the mission model and provide the required data.

3.4 ***Internal Software Interface***

This module can be treated as a data processing center inside the software environment. It is designed to provide a bridge between a multi-disciplinary team of algorithm and software developers so that they can design and implement their respective components with minimal coordination. Typically, software engineers program the user-interface components (Organizational Assets, Mission Modeling), while researchers are developing the optimization algorithms. The Internal Software Interface module serves as a translator between different data structures used by software engineers and researchers. It also provides an external interface that


Description of the task
DM requirements and other information
Variables and assignment preferences
Lifecycle parameters and resources
Add resource
Delete resource
Import
OK
Cancel
Apply

Name: [✓ To show] North Beach

Lifecycle parameters

Initiation	to show	0	type: float
Activation	to show	0	type: float
Duration	to show	10	type: float
Completion	to show	2	type: float
Deadline	to show	48	type: float
Retirement	to show	N/A	type: string
Delay impact	[✓ to show]	N/A	type: string
Task Position	to show	28	73

Criticality parameters

Priority	5	type: float
Type	Offend	type: string

Resources

AAW	0
ASUW	0
ASW	0

Figure 8. Example of task parameters

can link with other environments (e.g., DDD simulator, MATLAB-based algorithm development), thereby providing a convenient way of testing new algorithms, as well as distributed adaptive agent-based simulations planned for future research.

3.5 *Three-phase Organizational Design Process*

After Mission Modeling, the data structures are sent through the Internal Interface to the Optimization components of the Organizational Design Environment. After the *Performance Criteria/Measures* component is used to define objective functions for the design process, the last three components of our software environment (*Schedule Generation*, *Resource Allocation*, and *Hierarchy Construction*) allow an analyst to perform a step-by-step design of the organizational structure. Subject to the constraints defined in the Mission Modeling part, the Organization Structure is obtained in three phases. Algorithmic details are provided elsewhere [Levchuk *et al.*, 2000a,b].

Phase I (scheduling).
The scheduling phase of the organizational design process can be generally described as follows. A set of tasks with specified processing times, resource requirements, locations and precedence relations among them need to be executed by a given set of platforms with specified resource capabilities, ranges of operation and velocities. Tasks are assigned to groups of platforms in such a way that, for each such assignment, the vector of task's resource requirements is component-wise less than or equal to the aggregated resource capability of the group of platforms assigned to it. The task can begin to be processed only when all its predecessors are completed and all platforms from the group assigned to it arrive at its location. A resource can process only one task at a time. Platforms are to be routed among the tasks so that the overall completion time (called *Mission Completion Time* – the completion time of the last task) **or** a combined objective function assembled from individual mission objectives such as the completion time, accuracy,

workload, expended resources, external coordination etc., is minimized. An output of scheduling phase is an optimal or sub-optimal platform-task allocation and platform-task schedule.

The user is able to choose between different optimization algorithms according to the computational resources, problem size and allowable tolerances (degree of sub-optimality). The algorithms include optimal scheduling (based on dynamic programming) and heuristic methods (such as *dynamic list scheduling* based on *critical path*, *level assignment* or *weighted length* procedures; *pair-wise exchange* improvement; *Lagrangian relaxation* and various *decomposition algorithms*).

Figures 9 and 10 show the scheduling result for the Dynamic List Scheduling using weighted length sub-optimal algorithm and the Pair Wise Exchange sub-optimal algorithm, respectively. We can see the overall processing time for the first result is worse; however, it requires fewer assets, since the first 2 assets are not used in the mission.

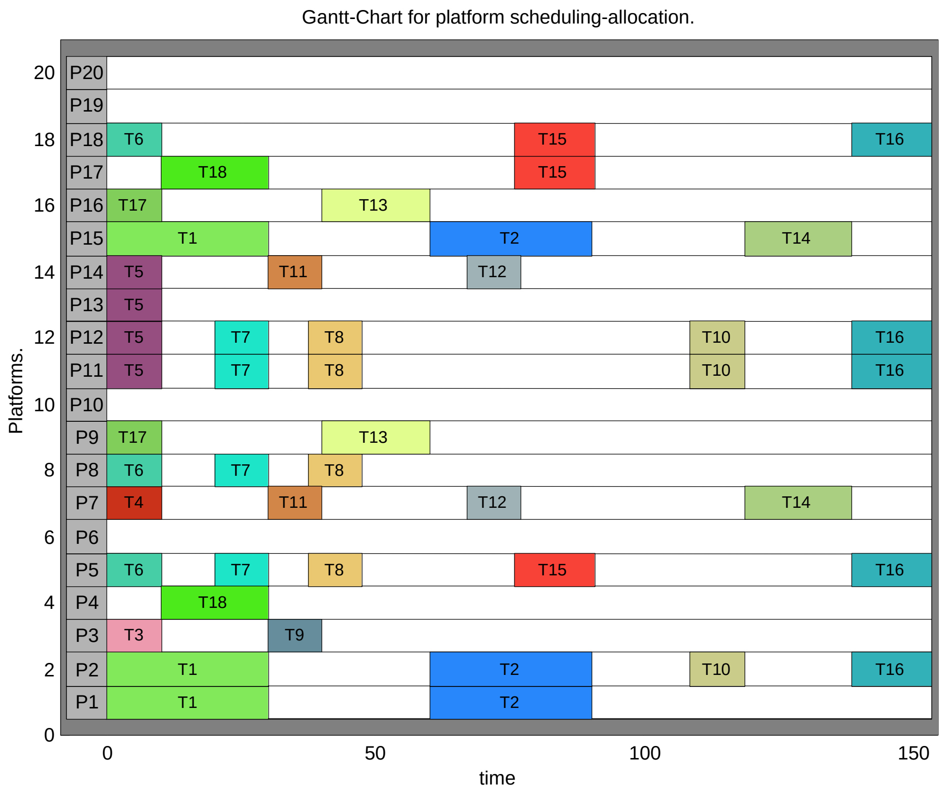


Figure 9. Scheduling result for Dynamic List Scheduling algorithm

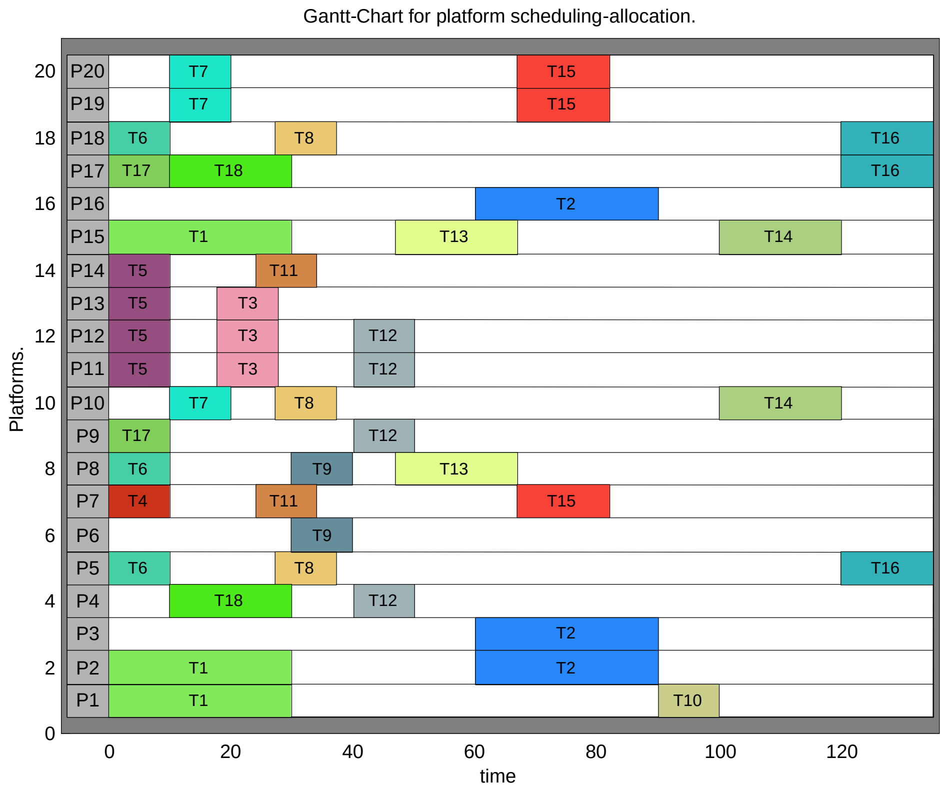


Figure 10. Scheduling result for Pair Wise Exchange sub-optimal algorithm

Phase II (clustering).

In phase II of our organizational design process, the assignment results obtained in phase I are used to allocate platforms to decision-makers (DMs). A platform-task assignment gives us information about required coordination among platforms. This coordination among platforms stems from the need to process the same task; it is carried out through DMs assigned to these platforms as information/decision carriers. The coordination that occurs is one of information, decision, and action. Any two DMs are said to coordinate in processing a task if they are "owners" of platforms that are required simultaneously to process this task. In Phase II, only the workload due to direct one-to-one coordination among DMs and internal workload are considered.

(1) Given the data from phase I, platforms are clustered into groups to be assigned to DMs. The objective is to minimize the DM coordination workload associated with DM-platform-task assignment. The workload is defined as a weighted sum of the internal and direct one-to-one external coordination, as well as the task workload. An output of the clustering phase is the DM-resource allocation.

The analyst can choose different algorithms according to the specified objectives. They include optimal algorithm (based on linear binary programming formulation) and heuristic methods (various hierarchal clustering procedures).

Phase III (organizational hierarchy).

In phase II, allocation of DMs to resources (platforms) is obtained. An external DM-DM coordination is determined based on joint task processing. DMs with their inter-DM coordination represent a network, where nodes are the DMs and edges denote coordination induced by joint task processing. Edge weights are equal to the required amount of coordination.

The hierarchy consists of links through which it is permitted to communicate inside the hierarchy. These links form a tree in the network of DM nodes. The goal is to match the organizational hierarchy to the coordination network that is necessary for completing the mission. Different definitions of matching create different formulations of the hierarchy construction problem. The output of phase III is the hierarchy tree with defined *superior-subordinate (supported-supporting) relations* between the nodes.

According to the specified performance criteria, the algorithms are chosen to construct the hierarchy tree. The algorithms include optimal constrained tree (with constraints on information flow), optimal coordination tree (based on minimizing the overall addition coordination added into the DM coordination network) and heuristic methods (e.g., maximal spanning tree).

Figure 11 and 12 shows the results of Phases II and III corresponding to the outputs of two scheduling algorithms shown in Figures 9 and 10. The DM workloads in Figure 11 are also better than those in Figure 12. From these results we can see that minimizing the overall processing time ("maximizing the speed of command") is not the only optimization criterion. The software environment provides a number of optimization choices so that an analyst can tradeoff a shorter processing time versus a better hierarchy structure.

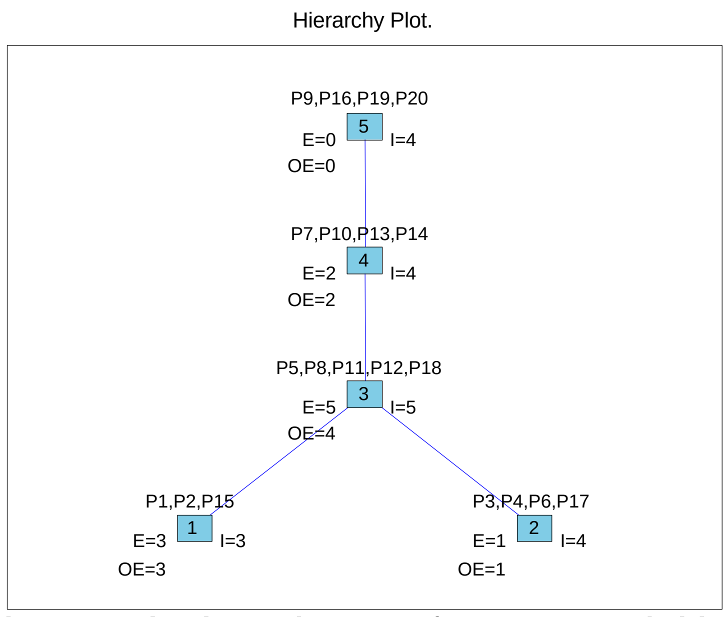


Figure 11. Clustering result and Hierarchy structure for Dynamic List Scheduling Algorithm

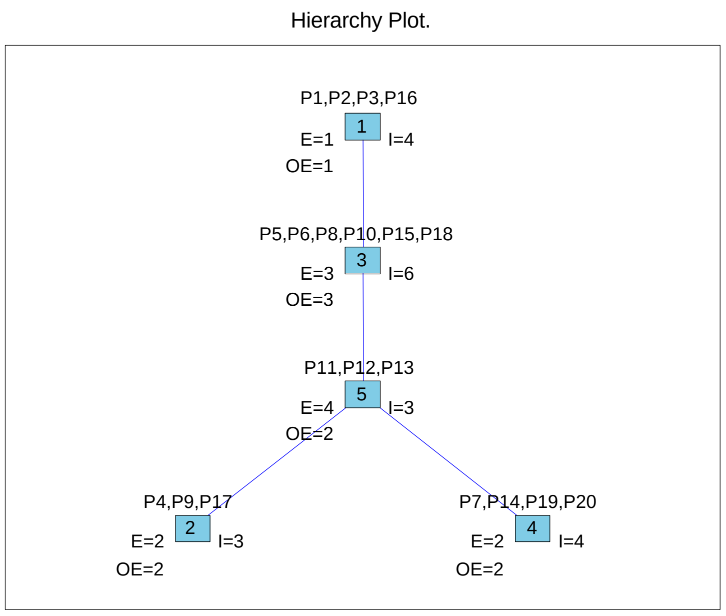


Figure 12. Clustering result and Hierarchy structure for Pair Wise Exchange Algorithm

4. **Summary and Future Extensions**

This paper presented a software environment for the Adaptive Organizational Design, which provides a bridge between the analyst and the researcher. The software environment helps analysts to understand the mission model and describe a complex C2 mission in a natural way. It has been designed to be highly flexible to model the large variety of modern C2 missions. In addition, it is modularized to improve the efficiency of the development of Adaptive Organizational Design environment by a multi-disciplinary team of software and algorithm developers.

The future implementation extensions of the current software environment include the following.

(2) Enhanced organizational design process to include information, action and decision coordination.
(3) Event-driven stochastic mission modeling to characterize dynamic and probabilistic task dependency graphs.
(4) Online adaptation of organizational strategies and structures in the face of changes in mission environment and/or organizational assets.
(5) Measures of organizational performance.
(6) Automated interface between organizational design environment and the DDD-III simulator for experimentation and for evaluating architectures in a realistic simulation environment.

References

[Carley *et al.*, 1995] K.M. Carley and Z. Lin. *Organizational Design Suited to High Performance Under Stress.* IEEE Transactions SMC, Vol. 25, 1995, 221-231

[Kleinman *et al*., 1996] D.L. Kleinman, P. Young and G.S. Higgins. *The DDD-III: A Tool for Empirical Research in Adaptive Organizations.* Proceedings of the 1996 Command and Control Research and Technology Symposium, Monterey, CA, June 1996

[Levchuk *et al*., 1997] Y.N. Levchuk *et al*. *Normative Design of Organizations to Solve a Complex mission: Theory and Algorithms*. Proceedings of the 1997 Command and Control Research and Technology Symposium, Washington, DC, June 1997

[Levchuk *et al*., 1998] Y.N. Levchuk, K.R. Pattipati , and D.L. Kleinman. *Designing Adaptive Organizations to Process a Complex Mission: Algorithms and Applications.* Proceedings of the 1998 Command & Control Research & Technology Symposium, NPS, Monterey, CA, June 1998.

[Levchuk *et al*., 1999a] Y.N. Levchuck, K.R. Pattipati and D.L. Kleinman. *Analytic Model Driven Organizational Design and Experimentation in Adaptve Command and Control.* Systems Engineering, Vol. 2, No. 2 , 1999.

[Levchuk *et al*., 1999b] Y.N. Levchuk, Jie Luo, Georgiy M. Levchuk, K.R. Pattipati, and D.L. Kleinman. *A* Multi-Functional Software Environment for Modeling Complex Missions and Devising Adaptive Organizations. Proceedings of the 1999 Command & Control Research & Technology Symposium, NPS, Newport, RI, June 1999.

[Levchuk et al., 2000a] G.M. Levchuk, Y.N. Levchuk, Jie Luo, Fang Tu, and K.R. Pattipati. *A Library of Optimization Algorithms for Organizational Design*. Dept. of ECE, Univ. of Connecticut, Cyberlab TR-00-102, Storrs, CT 06269-2157.

[Levchuk et al., 2000b] G.M. Levchuk, Y.N. Levchuk, Jie Luo, Fang Tu, and K.R. Pattipati. *A Library of Optimization Algorithms for Organizational Design*. Proceedings of the 1998 Command & Control Research & Technology Symposium, NPS, Monterey, CA, June 2000.

[Papastavrou, 1992] J.D. Papastavrou and M. Athans. *On Optimal Distributed Detection Architectures in a Hypothesis Testing Environment.* IEEE Transactions on Automatic Control, Volume 37, 1992, 1154-1169.

[Pete *et al.*, 1998] A. Pete, K.R. Pattipati, D.L. Kleinman, and Y.N. Levchuk. *An Overview of Decision Networks and Organizations.* IEEE Trans. Syst., Man, Cybern. May. 1998, pp. 172-192.

[Reibman *et al.*, 1987] A. Reibman and L.W. Nolte. *Design and performance comparison of distributed detection networks.* IEEE Trans. Aerosp. and Electr. Syst., vol. 23, November, 1987, 789-79

[Tang *et al.*, 1993] Z.B. Tang, K.R. Pattipati, and D.L. Kleinman. *Optimization of Distributed Detection Networks: Part II. Generalized Tree Structures.* IEEE Transactions on Systems, Man and Cybernetics , vol. 23, 1993, 211-221.