10th International Command and Control Research and Technology Symposium

The Future of C2

TOPIC: C2 Modeling and Simulation

Modeling Supervisory Control in the Air Defense Warfare Domain with Queueing Theory[1]

Joseph DiVita, Robert Morris and Glenn Osga

San Diego Systems Center, Space and Naval Warfare
Code 246210
53560 Hull Street
San Diego, CA 92152
Phone 619 553 6461/fax 619 553 3650

[1] This work was sponsored by the Office of Naval Research, Cognitive, Neural and Social Science & Technology Division. Research Area: Decision Support Systems and Models for Intelligent Mission Management.

Abstract

In this paper, we hypothesize that the performance of a supervisory control operator that must process tasks recommended by a system task manager is analogous to the performance of a vacationing server, $M/E_r/1$ queue. Thus, we assume that the input process is Markovian and that service consists of r- stages of processing each of which is exponentially distributed. In addition, we assume that when there are no tasks in the queue to process, the operator "takes a vacation," i.e., goes off and performs other duties. The model assumed vacation time was exponentially distributed. We derive the queueing statistics for this system. These statistics include (1) the average number of customers, tasks, in the queue, (2) the average time a task spends in the queue, and (3) the average waiting time in the queue. We extend this model to a two-class priority $M/E_r/1$ vacationing server system. The results of these predictions were compared to actual operator performance. This operator was also modeled using GOMSL. Both the GOMSL and queueing models provided effective prediction of actual operator performance.

Approach

The goal of our research program is to develop quantitative models of operator and system performance that will form the basis of a scientific design approach that can be utilized by future Combat System Design Engineers. These models are being developed for Air Defense and Land Attack combat systems, and are being incorporated into prototypes of the future Multimodal Watchstation (MMWS) and Land Attack Weapons Systems (LAWCS). Several projects (e. g., MMWS, LAWCS, and Combat Supervisory Support Systems, see Osga, Van Orden, Campbell, Kellmeyer & Lulu, 2002) have demonstrated tools that form the foundation for further development of interface concepts that will enable operators to plan and execute complex tasks within dynamic and multiple warfare areas. There is a growing need to model these interface concepts so that future interface designs may evolve in a principled and systematic fashion.

Currently, the method of evaluating interface design alternatives is through usability testing. There are several drawbacks to this approach. Proposed design alternatives contain information that is deemed to support critical cognitive processes for each task domain. There is no precise method to prescribe a display layout or design based upon the task information requirements, thus the iterative testing of hypothesized best layouts is required. The "size" of the design space and the constraints of the design space are unclear and unbounded. Another problem is that the proof of the value of these design hypotheses lies solely in usability testing and data collection. The degree to which this testing can be effectively done is debatable since time constraints pose various limitations - for example, a small number of test subjects and prototypes with limited fidelity are typical drawbacks for these studies. At best this design process can produced a heuristic set of "lessons learned" and hopefully a usable interface. As viewed from the most negative perspective, this design process may require many cycles of empirical testing of ad hoc systems that is only terminated when project resources are expended or when performance results are finally achieved. Unfortunately if resources are expended,

a design that is just “good enough” may be accepted vs. one that is optimal for task conditions.

Our research approach supports model-based design as opposed to creative engineering. We believe that the latter approach lacks the ability to predict human performance. Performance predictability is essential to good design. The goal of model-based design is to establish a "true engineering method for interface design" (Kieras, 2002).

In previous research (DiVita, Osga & Morris, 2004), performance for two Air Defense Warfare (ADW) Teams was analyzed using queueing theory statistics. This analysis revealed that task allocation, work- flow and the internal dynamics of the two teams were very different (DiVita, Osga & Morris, 2004). Queuing theory provides formulas - quantitative predictions for these statistics. These formulas are based on assumptions of input and output task flow and task prioritization. Our research develops a predictive model for the ADW team viewed as a queueing network. This paper focuses on the performance of one operator – one node – in the network. Our ultimate goal is to use these formulas to predict and evaluate operator and system performance. These quantitative models may then be used to simulate and quantify the effects of increasing and decreasing team size and will provide a model of manning and automation requirements. The nature of task allocation and dynamic task reallocation schemes among team members and autonomous agents may be tested with these models.

I. Queueing Theory

The increased automation of combat weapon systems is radically changing the role of the human operator from that of controller to supervisor. As a supervisor, the operator is responsible for monitoring and performing multiple tasks. In order to support the multitasking activity associated with supervisory control, a Task Manager (TM) Display is being incorporated into future combat weapon systems such as the Multimodal Watchstation (MMWS) and the Land Attack Combat System (LACS) (Osga, Van Orden, Campbell, Kellmeyer & Lulu, 2002). As pictured in Figure 1, the TM Display represents tasks, in the form of icons on a display screen, which the system has determined actionable given the current tactical information and Rules of Engagement (ROE).

The posting of tasks to the TM display for operators to perform, is analogous to service calls arriving at a Help Desk or calls to any telephone system. Other examples include: “jobs” arriving at a computer processing system and customers waiting in line for some service or other such as at a bank, a post office or grocery checkout counter. In the area of supervisory control we are interest in the flow of tasks – work - through a system that is comprised of both human servers and automated servers- computers. Quantitative models and methods that analyze dynamic systems of flow have been developed in the domain of Queuing Theory. We briefly review some critical aspects and results of Queueing theory. For a thorough introduction to the topic the reader is referred to Kleinrock (1975 & 1976, Volumes I and II).

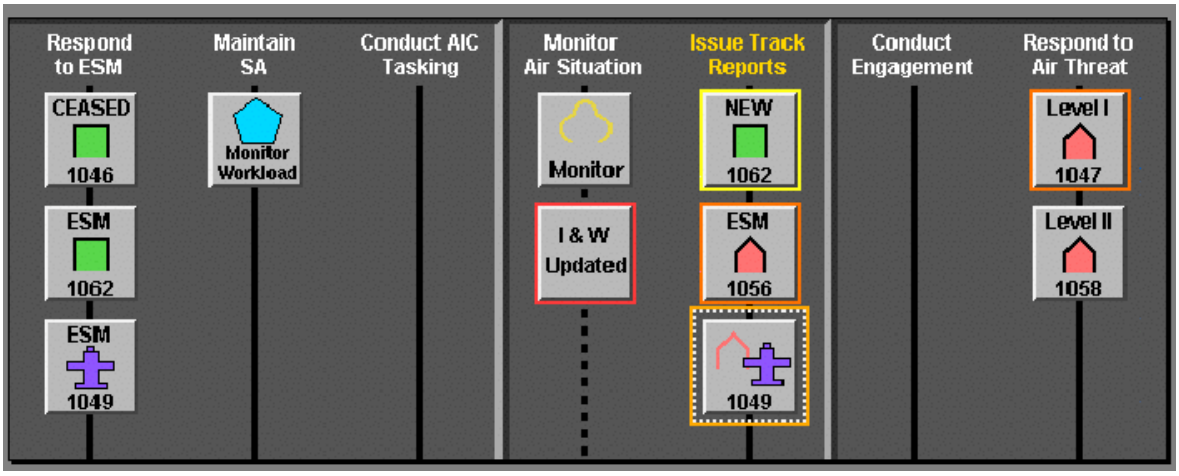

Figure 1. Task Manager TM display. The posting of tasks to the display is analogous to customers arriving for service at a help desk.

A queueing system is comprised of three main components:

A) The input or arrival process. The arrival of customers to a queue is often unpredictable. In this case, arrival is modeled as a random or stochastic process. The arrival process is often assumed to be Poisson in nature in which case the arrival rate, λ, is simply the reciprocal of the mean inter-arrival time of customers. For the Poisson distribution with parameter λ, the probability, P_k, that k arrivals occur in the time interval (0,t) is given by:

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

B) The service mechanism. Service refers to the number of "servers" and the lengths of time the customers hold servers. In our case this is the number of operators and the distributions of reaction times it takes operators to perform various tasks. This is often modeled by a continuous random variable, x, exponentially distributed with parameter μ :

$$f(x) = \mu e^{-\mu x}$$

The negative exponential is uniquely suited to describe the behavior of the human server because there is a considerable body of evidence demonstrating that human reaction time to various tasks, and task components, are in fact exponentially distributed (see Townsend & Ashby, 1984). As discussed below, service time may be viewed as a process composed of several serial stages each of which is expontentially distributed (see the section entitled, The Method of Stages – the $M/E_r/1$ Queue),. In this case an Erlang distribution is used to model service time.

C) The queueing policy entails the method by which the system selects customers: first-come-first-served (FCFS), last-come-first-served (LCFS), by priority, or at random. Initially we will consider a FCFS policy. Later we will derive the appropriate queueing statistics for a queueing system that prioritizes customers into classes of high and low priority.

A Queueing system with a single server and Markovian input and output is referred to as an M/M/1 queue. In this notation, the first field refers to the nature of the input process (Markovian), the second field, the nature of the output process (Markovian) and the last field the number of servers. Often queueing systems describe non-Markovian stochastic processes. For example, service time distribution may be Erlangian, in which case we have a $M/E_r/1$ queueing system. For our purposes, assuming exponential service time does not pose a problem; however, we are on shakier ground when assuming that the arrival of tasks follow a Poisson process. Our research has shown that, during the course of a scenario, the arrival rate varies as a function of time. This requires a more sophisticated model of the arrival process (for example, the Markov-modulated arrival process, see Hock, 1996) that we do not address in this paper.

Vital Statistics of a Queueing System.

The load or intensity, ρ, to a queueing system is defined to be the ratio of the rate of arrivals λ to the rate of service μ. Thus we have:

(1) $$\rho = \frac{\lambda}{\mu}$$

Various distributions and statistics can be derived for a queueing system. Most of these expressions involve the quantities ρ, λ and μ. For example in analyzing the performance of a system, one may be interested in 1) The average number of customers in the system at any time, 2) the average time a customer spends waiting for service, and 3) The total average time spent in the system by a customer. This time includes both service and waiting time. One useful result that relates several of these quantities, is Little's theorem which states that the average number of customers to the system, N, is equal to the product of the rate of flow of customers, λ, and the average time spent in the system, T:

(2) $$N = \lambda T$$

Another useful characterization of the system is the probability distribution for the number of customers in the queue. For example, what is the probability that a visitor to the queue finds n customers in the queue? Below, we list four essential formulas for an M/M/1 system:

a) The probability, P_k, of having k customers in the system is given by:

(3) $$P_k = (1-\rho)\rho^k$$

Thus the probability of having n or more customers in the queueing system is given by:

$$p[N \geq n] = \sum_{k=n}^{\infty} P_k$$
$$= \rho^n$$

b) The average number of customers, N, or the expected length of the queue, *E[L]*, is given by the equation:

(4) $$\mathrm{N} = E[L] = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$$

c) Using Little's Theorem, the average Time, *T*, spent in the queue, (including both service and waiting time) is:

(5) $$T = \frac{N}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda}$$

d) Lastly, the average waiting time, *W*, spent in the queue equals:

(6) $$W = T - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

The Vacationing Server

One critical difference between the queueing system described above and an operator engaged in supervisory control, is that when no tasks are present, the server is idle; however, this hopefully is not the case with human operators. When there are no tasks on the TM display, operators should go off and examine other information sets available on their workstation. For example, in the arena of air defense warfare there is a tactical map referred to as the TACSIT display that is continually examined.

Thus, there are "tasks" that the servers may perform that fall outside the flow of tasks represented on the TM display. These non -TM tasks must be taken into account in order to quantify system performance because they clearly will have an impact on the queueing statistics. For example, if the operator is evaluating information on a tactical display and a task arrives on the TM display, the operator may finish his analysis (non-preemptive service) before beginning the task on the TM display. Thus the waiting time and overall time the task on the TM display spends in the system will be affected by the operator's extra activity.

Fortunately, a queue with "service vacations" can be adapted to model our situation (Tadecki, 1984). The idea behind such queues is as follows: if there are no customers in the queue that need to be served, the server takes a vacation. No assumptions about the distribution of vacation times need to be made. If upon completion of a vacation, the server returns only to find that there are still no more customers, the server takes another vacation. The application of this queue to our situation is straightforward. If the operator has no tasks on the TM display he "takes a vacation" by analyzing information on other displays. When he completes this task he "returns from vacation" to see if there are any tasks on the TM display. For our case, we will assume that the operator's 'vacation times' and service times are both exponentially distributed however the parameters v and μ for vacation time and service time, respectively, are not necessarily equal.

We have shown (DiVita & Morris, 2005) that the average number of customers to a M/M/1 vacationing server is given by:

$$N = \frac{\rho}{1-\rho} + \frac{\lambda}{v}$$

Thus the average number of customers is equal to the sum of the average number to the non-vacation server, $\frac{\rho}{1-\rho}$ plus a factor that is proportional to the build or "loading-up" of customers while the server was away - $\frac{\lambda}{v}$. The $\frac{\lambda}{v}$ term is very similar to $\rho = \frac{\lambda}{\mu}$. It represents a "load" in the sense of customer build up. We can continue from here to derive all the other usual queueing statistics. A pattern emerges in the sense that in all cases we get the standard result for the non-vacationing server plus a factor associated with the server taking a vacation.

Using Little's theorem equation (2), we can now derive the expected amount of time, T, a customer will spend in the system:

$$T = \frac{1}{\mu - \lambda} + \frac{1}{v}$$

This time, T, includes both waiting and service time. If we subtract out the average service time, $\bar{x}$, (where $\bar{x} = \frac{1}{\mu}$), we obtain W, the average waiting time for a customer to the queue:

$$W = \frac{\rho}{\mu - \lambda} + \frac{1}{v}$$

An M/M/1 Non-preemptive Priority Queue with Vacation Time.

Our next topic introduces the prioritization of customers (tasks) for the vacationing server queuing system. A server whose queueing policy prioritizes tasks allows tasks with a higher priority to be served before those with a lower priority. Thus higher priority customers have "bumping" or "cutting ahead in line" privileges. This server may preempt service to a lower priority customer when a higher priority customer appears, or the server may complete service– a non-preemptive priority queueing policy. We studied this latter queueing policy. Thus the server completes service to the current customer regardless of the priority class of customers that may have arrived to the queue while the current customer is being serviced. "Bumping rights" apply only to tasks waiting in line.

Hock (Queueing Modeling and Fundamentals, pages 142 – 143, 1996) derives the waiting time, W, for a M/G/1, non-preemptive Priority Queueing System. Once the waiting time is derived, the average time spent in the queue, T, may be computed by simply adding the average service time, $\bar{x}$, to W. In other words, $T = W + \bar{x}$. Once T is known, Little's theorem can be use to compute N, the average number of customers to the queue. Hock argues that the waiting time of a typical customer of class i, who arrives at the system, is made up of four components: 1) The mean residual service time for a

customer currently being served upon the arrival of the new customer. 2) The mean total service time of customers in the same class as the new customer found in the waiting queue at the time of arrival. 3) The mean total service time of customers of class j , j < i, found in the queue at the time of arrival. 4) The mean total service time of those customers of class j, j < i, arriving to the system while the "new" customer of class i is waiting for service.

It can be shown that the mean residual service time (1) is equal to:

$$R = \frac{1}{2}\sum_{k=1}^{n} \rho_k \left(\frac{\bar{x_k^2}}{\bar{x_k}} \right) = \frac{1}{2}\sum_{k=1}^{n} \lambda_k \bar{x_k^2} = \sum_{k=1}^{n} \frac{\rho_k}{\mu_k}$$

Combining this formula with the other three factors mentioned above we obtain the average waiting time for a customer of class i:

$$W_i = R + \bar{x_i} N_q^i + \sum_{j=1}^{i-1} \bar{x_j} N_q^j + \sum_{j=1}^{i-1} \bar{x_j} \lambda_j W_i$$

For a class 1 customer, the third and fourth components are not a factor. If we note that $N_q^1 = \lambda_1 W_1$, and that $\rho_1 = \lambda_1 \bar{x_1}$, then we may solve for W_1 to obtain:

$$W_1 = R + \rho_1 W_1$$

$$W_1 = \frac{R}{1 - \rho_1}$$

Using these results we can determine the average waiting time for class 2 customers:

$$W_2 = R + \rho_2 W_2 + \rho_1 W_1 + \rho_1 W_2$$

$$W_2 = \frac{R}{(1 - \rho_1)(1 - \rho_1 - \rho_2)}$$

It is a straightforward matter to extend this model to the case of the vacationing server. We first note that factors 2, 3 and 4 discussed above remain unchanged for the vacationing server. This is because once a customer arrives, we are assuming that the server will not take another vacation until this customer and all the customers ahead of the new customer are served. Thus the only component to the model that has to be modified is residual service time (1). A new customer to the queue may experience either residual service time or residual vacation time, R_v, that is the time they must wait for the server to come back from vacation and resume service. In order to compute this residual time we need to know 1) the probability of a customer entering the queue to find the server on vacation, and 2) the average time left for the server's vacation. If we assume that the server's vacation time is exponentially distributed, that is, memoryless then (2) poses no problem. Given that there are n classes of customers, the probability that a new customer arrives to find the server on vacation is $1 - \rho_1 - \rho_2 - ... - \rho_n$. The reason behind this is as follows. The probability that a new arrival finds a customer of class i in service

is ρ_i. Thus the probability that a customer of any class is in service upon arrival to the queue is $\sum_{i=1}^{n} \rho_i$, and the probability that the server is on vacation i.e. not serving is $1 - \sum_{i=1}^{n} \rho_i$. We will consider a queueing system that prioritizes customers into two classes. Thus if we let $\hat{R}$ equal the new residual time associated with waiting for service or vacation time completion we obtain :

$$\hat{R} = R + (1 - \rho_1 - \rho_2)R_v$$

With this we can compute the waiting time for a class 1 customer to a queue with a vacationing server, W_1^v:

$$W_1^v = \hat{R} + \rho_1 W_1^v$$

$$W_1^v = \frac{\hat{R}}{1 - \rho_1}$$

Making the substitution for $\hat{R}$ we obtain:

$$W_1^v = R + (1 - \rho_1 - \rho_2) + \rho_1 W_1^v$$

$$W_1^v = \frac{R}{1 - \rho_1} + \frac{(1 - \rho_1 - \rho_2)R_v}{1 - \rho_1}$$

We see the same pattern emerge for the vacationing server as above. That is, the new waiting time is equal to the original waiting time formula for Class 1 customers, $\frac{R}{1 - \rho_1}$, plus a factor associated with the vacationing server, $\frac{(1 - \rho_1 - \rho_2)R_v}{1 - \rho_1}$.

Next we derive the formula for W_2 the waiting time for customers of Class 2. In this case all four components come into play:

$$W_2^v = R + (1 - \rho_1 - \rho_2)R_v + \rho_1 W_1^v + \rho_1 W_2^v + \rho_2 W v_2$$

Substituting the term we obtained for W_1 above and collecting like terms we obtain:

$$W_2^v = \frac{R}{(1 - \rho_1)(1 - \rho_1 - \rho_2)} + \frac{R_v}{(1 - \rho_1)}$$

Or equivalently:

$$W_2^v = \frac{\hat{R}}{(1 - \rho_1)(1 - \rho_1 - \rho_2)}$$

Again, we have the "original" formula plus a factor associated with the vacation.

As mentioned above, from the waiting time W_i^v, we can compute the average time in the system: $T_i = W_i^v + \bar{x}_i$, where $\bar{x}_i$ is the average service time for Class i customers. Thus the average number of Class i customers is given by $N_i = \lambda_i T_i$. What is also of interest here is the average number of customers in the queue, N, the average waiting time, T, and the average time spent in the queue, W. The average number of customers is given by: $N = N_1 + N_2$. If we let $\lambda = \lambda_1 + \lambda_2$ then $T = \frac{N}{\lambda}$. Lastly $W = T - \frac{1}{\mu}$. Using the formulas $\rho = \rho_1 + \rho_2$ and $\rho = \frac{\lambda}{\mu}$, we obtain the average service time, $\frac{1}{\mu} = \frac{\lambda_1\mu_2 + \lambda_2\mu_1}{\lambda\mu_1\mu_2}$ which allows us to compute W.

The Method of Stages – the M/E$_r$/1 Queue

The servicing of a task may be viewed as a process consisting of several serial stages. The Erlang method of stages was designed to quantify this situation. This method assumes that the distribution of service time for each stage is exponential. The distribution of overall service time is thus given by the convolution of these functions. This distribution is no longer exponential; hence this method allows a more general service time distribution to be used in formulating predictions. When dealing with convolutions it is easier to work with Laplace transforms. Thus if we assume that there are r stages of processing each with identically distributed service times, the service time distribution is given by the Erlang distribution:

$$b(x) = \frac{r\mu(r\mu x)^{r-1}e^{-r\mu x}}{(r-1)!}$$

The Laplacian, B*(s), for b(x) is given by:

$$B^*(s) = \left(\frac{r\mu}{s + r\mu}\right)^r$$

We may generalize this situation by assuming that the various stages, although exponential, are not identical. If each has a rate of $a_i\mu$ then the Laplacian becomes:

$$B^*(s) = \left(\frac{a_1\mu}{s + a_1\mu}\right)\left(\frac{a_2\mu}{s + a_2\mu}\right)\cdots\left(\frac{a_r\mu}{s + a_r\mu}\right)$$

The advantage with the Laplacian is that it is a straightforward process to obtain the nth moment of the distribution. The nth moment is derived by taking the nth derivative of the Laplacian evaluated at s=0 and multiplying this by $(-1)^n$, that is:

$$\overline{x^k} = (-1)^k \frac{d^k B^*(s)}{ds^k} \ .$$

In computing R, the residual service time, we used the second moment of the service distribution (see above). If we assume a two-stage general Erlang service time distribution, where the rates of each stage is given by $a_1\mu$ and $a_2\mu$, respectively, then the second moment is given by:

$$\overline{x^2} = \frac{2(a_1^2 + a_1 a_2 + a_2^2)}{a_1^2 a_2^2 \mu^2}$$

This formula may easily be generalized to n stages of service. The use for a two-stage Erlang service distribution arises in our research quite naturally since the operator first decides what task to select and then selects and performs the task. Thus the set up time to select a task is modeled as the first stage of service, with its own average service time a_1, and the time to perform the task as the second stage.

Modeling Air Defense Teams with a Team of GOMSL Models

Our approach is to formulate a "modeled-based evaluation" (Kieras, 2002) of the MMWS interface using Natural GOMS Language (NGOMSL), (Kieras, 1988, 1997; John and Kieras, 1996). The Acronym GOMS (Card, Morgan & Newell, 1983) stands for Goals, Operators, Methods, and Selection rules. GOMS is an engineering model for interface design that attempts to explicitly represent the knowledge a user must have in order to perform certain tasks on a system. The operator uses certain Methods to accomplish specific Goals. The Methods utilize Operators in a series of steps that the user performs. The appropriate Method is chosen by Selection rules that reflect the user's goals and current operating context. NGOMSL is a natural structured language developed by Kieras and his colleagues, which represents user Methods and Selection rules. NGOMSL can predict learning and execution time for accomplishing specific tasks performed on a system. Santoro and Kieras (2002) have used GOMSL (an executable form of NGOMSL) to model a team of 4 operators performing an Air Defense Warfare (ADW) task with a team of GOMSL models.

Summary of Approach

Our goal is to integrate the team of GOMSL models with our Queuing theory model. This two-fold approach provides a powerful analysis of a system. On the one hand, Queueing theory quantifies large-scale aspects of system performance such as workload, input, output and work throughput, along with the dynamic flow of tasks among a team of operators. These statistics represent emergent characteristics of a

system that are not directly modeled. On the other hand, GOMSL modeling explicitly represents the strategies an individual operator and teams of operators may use to perform tasks and then quantifies operator performance based on these strategies.

Both these models are generative, that is, neither depends on a particular scenario to work; rather, given any scenario they will predict and evaluate operator and system performance. Together these two models are able to simulate and quantify the effects of increasing and decreasing team size and will provide a model of manning and automation requirements. The nature of task allocation and dynamic task reallocation schemes among team members and autonomous agents may be tested with these models.

Results

Data from an ADW team consisting of four operators were collected from a one hour and thirty minute ADW scenario entitled the Sea of Japan (SOJ). Data from this scenario were analyzed from the viewpoint of queueing theory. There are three comparisons that may be made: 1) GOMSL model data versus actual data 2) Queueing model predictions versus actual data; 3) Queueing model predictions versus GOMSL model data.

In Table 1 we compare the GOMSL data to the actual Air Warfare Coordinator (AWC) data over the first 33 minutes of the scenario. The first 33 minutes of the scenario was selected because the arrival rate of tasks did not change over this period. As mentioned above, if arrival rate varies as a function of time, then more sophisticated modeling of the arrival process is required. During this time there were only 7 high priority tasks, thus we combined the queueing statistics for high and low priority tasks.

In analyzing the actual data there are some decisions to be made as to when certain tasks ended. Several of the AWC's tasks entailed sending a text-to-speech message (new and update track reports). This message was followed by an acknowledgement of receipt of the message. The question is when did this task end for the operator? There are two extremes. One may argue that the task ended when the AWC selected the send button and a text-to-speech message began; however, upon reviewing video of actual operator performance, it appeared that operator behavior was inconsistent. What we found was that operators often waited to hear an acknowledgement on the part of the recipient of the text-to-speech message before beginning a new task. Likewise, there appeared to be cases where the operators waited to a degree, that is, for some intermediate time between these two extremes. In order to account for this inconsistent behavior, in the GOMSL model we have bracketed performance between these two extremes. Thus there is a GOMSL short-send-time model where the GOMSL operator starts a new task immediately after the send button is pushed. There is also a GOMSL long-send-time model where the GOMSL operator waits until an acknowledgement is received before starting a new task.

In order to estimate the task time for the actual data, as a first pass, we added 9 seconds to the time the operator selected the send button. This number was determined by combining the average length of the text-to-speech message to an estimate of the average acknowledgment time obtained from the GOMSL model. As can be seen in Table 1, the GOMSL models bracket actual operator performance.

AWC Server Node

	N – ave # of tasks	T – ave lifetime	W-ave wait
GOMSL high	1.232	54.06	28.27
Actual	0.936	40.919	20.27
GOMSL low	0.642	27.84	13.9

Table 1: GOMSL model data for the AWC compared to actual Team 1 AWC data.

In Table 2 we compare the queueing model predictions with actual data from the Team 1 Air Warfare Coordinator (AWC). Estimating the set up and vacation time from the data was quite difficult. Instead, we used estimates of these average times derived from the GOMSL model and assumed that the distributions were exponential. The GOMSL model gave an average set-up time of 3.56 sec and an average vacation time of 8.46 sec. In Table 2, the service time distribution was assumed to be a 2-stage Erlang distribution. The first stage consisted of the set-up time – the average of which was estimated using the GOMSL model and the distribution was assumed to be exponential. The second stage represents the actual average reaction time of the AWC operator to perform tasks. Here again we assumed that the distribution for stage-two service times was exponential. In Table 2, we also assumed the AWC prioritized tasks. The model's predictions are in error. The source of the error is our assumption that the distribution of stage-two service times was exponential. Comparing the second moment of the actual data, $\overline{x_O^2}$, to that of the assumed exponential, there is a large difference: 397.61 versus 565.15. One solution to this problem is to model the distribution of reaction times with an r-stage Erlang distribution that minimizes the error between $\overline{x_O^2}$, and the Erlang distribution's second moment, $\overline{x_E^2}$ (Kleinrock, 1975). That is, r is adjusted – stages are added or deleted- until the error is minimized.

AWC Performance: $M/E_2/1$ Model, Task Prioritization Queueing Policy

	N – ave # of tasks	T – ave lifetime	W-ave wait
Predicted Queueing	1.067	44.9	24.55
Actual AWC	0.936	40.919	20.27
% Error	12.24	8.86	17.42

Table 2: Queueing Model predictions compared to actual AWC Team 1 data.

In Table 3 we show the queueing theory predictions when the distribution of AWC reaction times was modeled with a r-stage Erlang distribution that minimized the error between $\overline{x_O^2}$ and $\overline{x_E^2}$. (The first moments of the actual and model distributions are set equal.) Table 4 lists the second moments and the r obtained from this analysis. For high priority tasks this error was minimized when r = 1. Low priority tasks required a 4-stage Erlang distribution, r =4. As is illustrated in Table 3, the error of the model's prediction was greatly reduced by refining our approximation of the service distribution.

AWC Performance: $M/E_r/1$ Model, Prioritization Queueing Policy

	N – ave # of tasks	T – ave lifetime	W-ave wait
Predicted Queueing	0.966	40.668	20.314
Actual AWC	0.936	40.919	20.27
% Error	3.11	0.62	0.21

Table 3: Queueing Model predictions compared to actual AWC Team 1 data.

Fitting Actual Service Time Data with r- stage Erlang Distributions

	r	$\overline{x_E^2}$	$\overline{x_O^2}$	% Error	N
High Priority Tasks	1	126.34	159.77	26.47	7
Low Priority Tasks	4	431.81	440.30	1.97	39
FCFS Tasks Combined	3	383.86	397.61	3.58	46

Table 4: Best fitting r-stage Erlang distributions that minimize seond moment error.

In Table 5 we eliminated the assumption that the AWC prioritized tasks; that is, we assume that tasks were performed on a first-come-first-served (FCFS) basis. The pooled data was modeled with a 3-stage Erlang distribution (see Table 4). Thus in Table 5, total service time is modeled as a 4-stage Erlang distribution (1 set-up stage + 3 service stages). The percent error between the model's predictions and the actual data is quite low. Eliminating the prioritization assumption did little to affect the overall predictions and the goodness of fit of the model. In Figure 2 we present a histogram of the actual service data, collected over the entire scenario, compared to the 3-stage Erlangian approximation.

AWC: $M/E_4/1$ Model, FCFS Queueing Policy

	N – ave # of tasks	T – ave lifetime	W-ave wait
Predicted Queueing	0.969	40.794	20.440
Actual AWC	0.936	40.919	20.27
% Error	3.41	0.31	0.83

Table 5: Queueing Model predictions compared to Actual AWC Team 1 data.

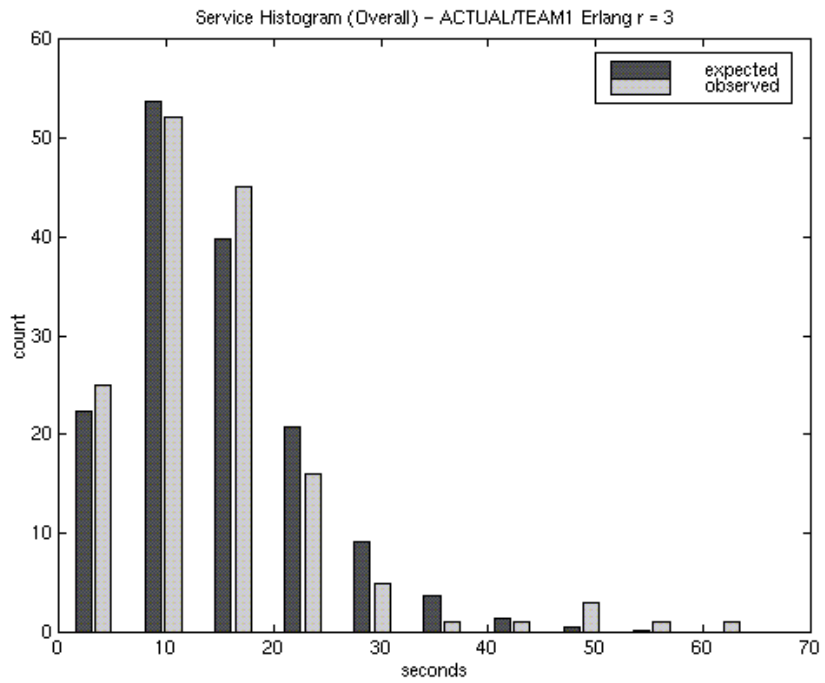

Figure 2. Histogram of Actual data service times versus 3-stage Erlang distribution.

Lastly, in Table 6 we list all the queueing parameters: λ_1: arrival rate of high priority tasks, λ_2: arrival rate of low priority tasks, λ: overall arrival rate of tasks; set-up$_1$: set-up rate of high priority tasks, set-up$_2$: set-up rate of low priority tasks, set-up: overall set-up rate of tasks; μ_1: service rate of high priority tasks, μ_2: service rate of low priority tasks, μ: overall service rate; v: vacation rate. (Note, taking the inverse of the rates, derives average times).

Queueing Parameters

	GOMSL High	GOMSL Low	Actual Data
λ_1	1/252.57	1/252.57	1/252.65
λ_2	1/49.88	1/49.88	1/50.51
λ	1/41.65	1/41.65	1/42.10
set-up$_1$	1/2.46	1/2.44	1/2.46
set-up$_2$	1/3.76	1/3.79	1/3.76
set-up	1/3.56	1/3.58	1/3.56
μ_1	1/22.06	1/15.19	1/7.95
μ_2	1/21.58	1/9.47	1/18.59
μ	1/21.66	1/10.41	1/16.81
v	1/8.46	1/7.76	1/8.46

Table 6: Queueing parameters.

We did not derive the queueing theory predictions for the GOMSL models because the distribution of service times for GOMSL tasks is somewhat arbitrary. Although the GOMSL mean reaction time may be representative of operator task performance, the distribution of service times is clearly not what one would expect from a human operator. This is because service time for the GOMSL model is a sequence of steps each of which is temporally deterministic. Thus, service time only varies in so far as the number of steps to accomplish a task varies. In order to obtain meaningful predictions, we would have to model this arbitrary service distribution.

Conclusion

In this paper, we hypothesize that operator performance may be modeled by a $M/E_r/1$ queue whose server "takes vacations". Thus, we assumed that the input process was Markovian and that service consisted of r- stages of processing each of which is exponentially distributed. In addition, we assumed that when there are no tasks in the queue to process, the operator "takes a vacation," i.e., goes off and performs other duties. The model assumed vacation time was exponentially distributed. We derived the queueing statistics for this system. These statistics included (1) the average number of customers, tasks, in the queue, (2) the average time a task spends in the queue, and (3) the average waiting time in the queue. The results of these predictions compared quite favorably to actual operator performance. The operator was also modeled using GOMSL, and the queueing statistics generated from the GOMSL model were also compared to the actual data. Both the GOMSL models and the queueing models provided effective predictions of actual operator performance.

Future work in this area will 1) extend the analysis to a team of operators – the other nodes of the queueing network. Thus in addition, to predicting the performance of each operator in a manner similar to our AWC predictions, we will obtain a model with predictions of team/system performance. 2) The model will be modified to allow for a

more general arrival process of tasks. 3) Different team structures that alter the flow of tasks through the network will be tested with these models to see if they predict previously observed differences in team performance.

References

Card, S.K., Moran, T.P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM* , 23(7), 396-410.

Card, S., Moran, T. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey: Erlbaum.

DiVita, J., Osga, G., & Morris, R. (2004) Modeling Team Performance In the Air Defense Warfare (ADW) Domain. Command and Control Research and Technology Symposium, San Diego, CA.

DiVita, J & Morris, R. (2005). The Vactioning Server: Queueing Models for Supervisory Control. Space and Naval Warfare Report, San Diego Systems Center (In press).

Hock, N.C. (1996). Queuing Modeling Fundamentals. John Wiley and Sons. New York.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, **3**, 320-351.

Kieras, D.E. (2002). Model Based Evaluation to appear in J. Jacko & A. Sears (Eds), *Human-Computer Interaction Handbook*, Lawrence Erlbaum Associates, in press.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of Human–Computer Interaction* (pp. 135–158). Amsterdam: North–Holland Elsevier.

Kieras, D. E. (1997). A Guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. Landauer, and P. Prabhu (Eds.), *Handbook of human-computer interaction*. (Second Edition). Amsterdam: North-Holland. 733-766.

KleinRock, L. (1975). Queueing Systems, Volume I: Theory; Wiley-Interscience, New York.

Osga, G., Van Orden, K., Campbell, N., Kellmeyer, D., & Lulu, D. (2002) Design and Evaluation of Warfighter Task Support Methods in a Multi-Modal Watchstation. Space & Naval Warfare Center San Diego Tech Report 1874.

Santoro T. & Kieras, D. E. (in preparation). Verification and validation of goms models for team communications and collaboration.

Takagi, H. (1991). Queueing Analysis. A Foundation of Performance Evaluation, Volume 1: Vacation and Priority Systems, Part 1. Elsevier Science Publishing Company Inc., New York.

Townsend, J.T., & Ashby, G. F. (1984). Stochastic Modeling of Elementary Psychological Processes. Cambridge University Press.