Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

THE COMMAND AND CONTROL COMMUNICATIONS AND INFORMATION NETWORK ANALYSIS TOOL (C3INAT)

John P. Mullen
Jason Rupe
Srinagesh Gavirneni
Way Kuo

IMSE Department
209 Engineering Annex
Iowa State University
Ames, Iowa 50011

ABSTRACT

The development of a large scale communications system is complicated by the great scope of design considerations. The Command and Control Communications and Informatin Network Analysis Tool (C3INAT) simplifies this problem by use of a stratified model which separates *Command and Control* C^2 and communication sub-models which may be developed independently. The C3INAT then integrates the two sub-models so that a communication system may be evaluated in light of its effect on C^2. C3INAT also provides efficient interfaces to existing data bases and the modular General Simulation System (GSS). Its output is compatible with most data base and statistical analysis packages.

1. INTRODUCTION

Telecommunications and computer storage and retrieval of information can greatly enhance the capability of Command and Control (C^2). However, the task of designing modern Communications and Information Networks (CIN) is very complex. This complexity is partially due to the large scope of the problem, encompassing many areas of expertise [Athans 1987; Bohannan 1985; Bleistein et al. 1985; Farrel et al. 1986; Georgiou and Lammers 1985; King and Martin 1985; Kirshna and Shin 1987; Kroening 1986; Wohl et al. 1984]

The *Command and Control Communications and Information Network Analysis Tool* (C3INAT) is a collection of software that facilitates the task of comparing alternative CIN designs. It is used to determine *Measures of Performance* (MOP) and *Measures of Effectiveness* (MOE) through stochastic simulation. Its distinguishing features are

- the use of established data and knowledge bases
- the separation of the C^2 and CIN functions.

The identification and use of established data bases including the *Communications Data Base* (CDB) capture the vast body of knowledge available on C^2 and on CIN. [Johnson et al. 1988] discusses these features of C3INAT. The separation of C^2 from CIN which simplifies modeling and makes analysis more meaningful and efficient, has not been discussed previously and is the main focus of this paper.

2. THE TWO-LAYER MODEL

Figure 1 depicts the basic model which contains aspects of the *Military Command and Control Evaluation Structure* (MCES) [Sweet et al. 1986] and is a logical extension of the OSI layered network model [Schwartz 1987]. There are, however, only two major layers and one interface. This condition neither precludes nor requires the use of layers in the C^2 or CIN sub-models. Analysts are thus free to model the major divisions as they see fit.

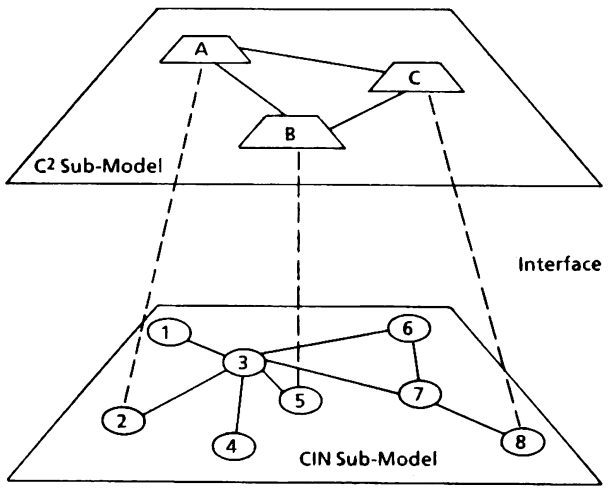

Figure 1. The Two Layer Model

The C^2 sub-model is essentially a description of C^2 communications and information requirements among a set of participants. The participants are usually people, but may also be information storage devices. Each *Needline* is a specific requirement between two or more participants that is conditional on the current activities and the state of the environment (This differs from the usual definition of needline, which specifies requirements among classes of participants). At this time, almost all needline information is derived from the CDB. For example, CDB needline type U504430 specifies that a Combat Support Aviation Company Commander will need to communicate with his operations officer about four times a day in the voice mode with a message length of 1000 bytes. A needline would be an instance involving a particular CSA.

Each attempt to communicate or gain information is termed a *call attempt*. It is initiated in the C^2 sub-model and, if the scheduled originator is idle, transformed into a request from the CIN model. The final outcome of the attempt is determined first by the response of the CIN model and then by the C^2 sub-model's response to the CIN result.

For example, suppose a needline specifies that Party A must communicate with Party C. The CIN sub-model attempts to find a path from instrument 2 to instrument 8. If one is found, the C^2 sub-model then checks the status of Party C. If that party is idle, the CIN's resources are assigned to the call and the call is connected.

The details of the CIN's action are not known to the C^2 sub-model, and the merit of the CIN is measured in how well or poorly it satisfies the C^2 requirements. In the earlier example, the CIN sub-model may connect via 2-3-6-7-8 or 2-3-7-8. The C^2 sub-model will not know which was the case. The only thing of concern is whether of not the call is completed. No other measures are used to determine MOEs, although other MOPs may be collected from the simulation in order to determine weak points of the CIN. Note that this call attempt could have been thwarted if no path could be found, or if either party were busy. The first case indicates a weakness in the CIN, but the second indicates a weakness in the overall communication plan, or a C^2 problem.

The CIN is modeled in the conventional manner as a set of nodes interconnected by links. This sub-model is usually built by telecommunications and computer experts to accurately reflect the performance of the CIN to be evaluated. There are no significant restrictions on the construction of this model.

The two sub-models are interconnected by the interface. The interface may be as simple as a "telephone directory," listing one instrument for each participant, or as complex as a pick order for instruments and a pecking order for participants who share instruments.

This design permits C^2 experts to develop the C^2 sub-model and CIN experts to model the CIN with little interference and interaction. It also gives the analyst great flexibility because it permits any C^2 sub-model to be used with any CIN sub-model and any CIN sub-model to be used with any C^2 sub-model.

3. AN OVERVIEW OF C3INAT

The C3INAT, described fully in [Kuo et al. 1990], is summarized briefly in this section to provide the necessary context for the discussion of the simulation model. As depicted in Figure 2, the C3INAT consists of four major subsystems

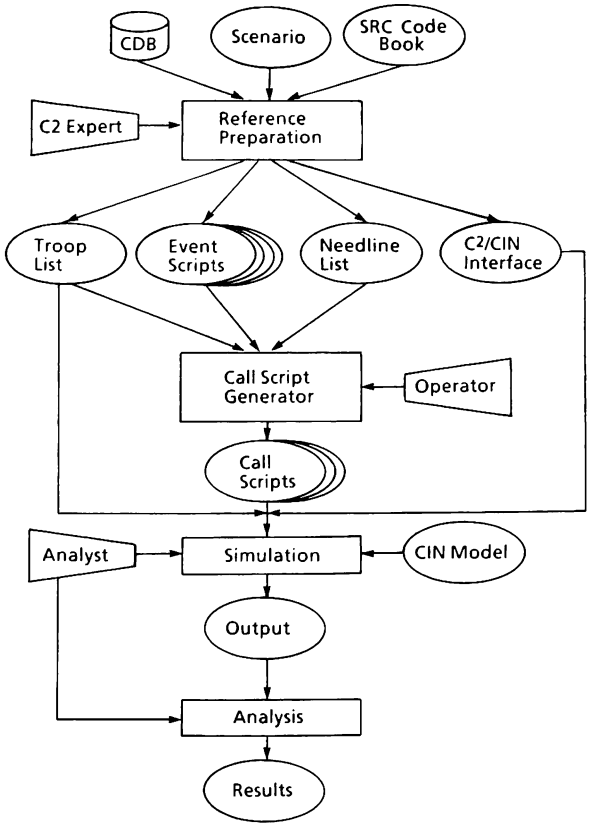

Figure 2. The C3INAT

- reference preparation
- call script generation
- simulation
- stochastic analysis.

Each of these is discussed very briefly below.

3.1 Reference Preparation

The main purpose of the reference preparation phase is to identify necessary information and put it into a standard form suitable for use in subsequent subsystems. This entails

- identifying the primary participants and producing a troop list
- producing the needline list, a working extract of the CDB
- synthesizing one or more event scripts on the basis of the contents of the scenario and anticipated analysis
- developing an interface to anticipated CIN models.

Because input data formats vary greatly, this phase is the most open-ended and least structured. Its success depends in great measure on the expertise of the C^2 specialists who perform this step.

One component of C3INAT used in this phase is a fast extraction program called `CDBXTRCT`. This program may be used to generate working extracts of the CDB, once each individual in the troop list has been identified by SRC code. These working extracts are then converted to dBASE III format for further processing.

Because of the unstructured nature of this phase, the most helpful tools are written procedures, the SRC codebook in dBASE format and the modeler's expertise.

The interface file details which participant in the troop list has access to which instrument in the CIN. It may be quite dependent on the CIN to be modeled, but is usually just a list of "telephone numbers" that can be assigned to specific instruments later.

3.2 Call Script Generation

Stochastic simulation requires the use of several independent replicates to accurately characterize the CIN's response. In addition, such inputs should be exactly repeatable in order to reduce between-treatment variance in contrasts of different CINs [Ripley 1987]. The Call Script Generator can produce randomly generated call scripts to satisfy both of these needs.

Event Scripts for steady state experiments are static, but those for terminating systems are not. This subsystem is designed so that one may generate either many random call scripts from each event script for terminating experiments or a few long call script steady state experiments. Thus a robust input to the simulation model can be assured in either case.

3.3 Simulation

Simulation models were developed in the *General Simulation System* (GSS), produced by Prediction Systems, Inc. in Manasquan, New Jersey. The Call and Event Scripts, which are used to model the C^2 requirements in the simulation model, are automatically read in. The C^2 sub-model is automatically interfaced to independently-generated CIN models during the C3INAT simulation. This allows the efficient testing and comparison of CIN models without significant rewriting or recompiling of the simulation model.

Because differing subprocesses may be exercised to greatly differing degrees, post simulation analysis is used. The output of the simulation subsystem is a record of each call attempt and its

disposition. This permits the analyst maximum flexibility in the analysis phase since the state of network at any particular time can be reconstructed from this record.

3.4 Analysis

The methods used in C3INAT include post-simulation truncation and batching. The output record is designed to facilitate isolation of key items from the rest of the record by using such data base software as dBASE III or FoxBase. In addition, the input is compatible with numerous data analysis packages, such as SYSTAT, SAS, and MINITAB.

4. THE SIMULATION MODEL

This simulation model of communication systems is divided into three main parts. They are C^2 sub-model, CIN sub-model, and C^2-CIN interface (see Figure 3). This design has been developed to make it easier to simulate various configurations with as few changes as possible in the software. In most cases, changes in the CIN sub-model do not affect the C^2 sub-model at all.

Independent environments exist for C^2 and CIN modeling. This separation allows the communication experts and C^2 experts to work independently of each other. It also allows much freedom in the pairing of C^2 and CIN sub-models.

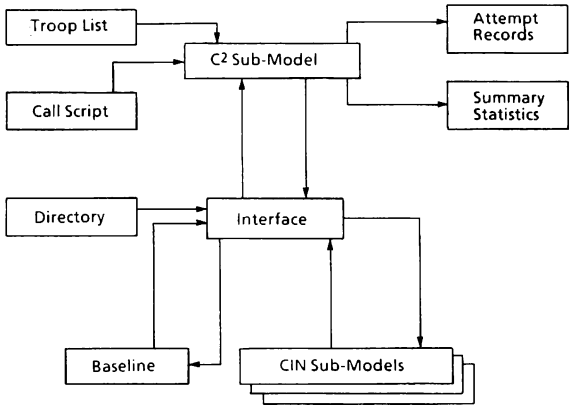

Figure 3. Simulation Overview

4.1 Overview

The C^2 sub-model addresses various communication needs of the parties and their reactions to various situations. It is based on the theory that the communicating parties react to various communication delays or problems in a manner that is independent of the CIN sub-model used. All communications originate with a human being and terminate with another human being. The current states of the origin and destination parties help determine the results whether at the start of a call attempt or in reaction to CIN response.

That is, the party's reaction to a busy signal is independent of whether it was caused by a lack of a trunk, poor propagation conditions or the destination's instrument being off-hook. Therefore, changes of state in the C^2 sub-model are enacted in response to information that would normally be available to the party rather than the exact CIN response.

The CIN sub-model describes the arrangement of the instruments in the communications and information network. This sub-model includes information regarding the various devices, links, and trunks being used and their availability at any particular moment. This part of the model determines whether the required device or network trunk is available and whether the particular call attempt connects. Thus this sub-model is mainly involved in the middle part of a call attempt.

The *baseline model* is an idealistic CIN sub-model that has unlimited resources. Its primary function is to establish a baseline response for a given call script. This special CIN sub-model is also used to test the C^2 sub-model.

The Interface forms the connection between the C^2 sub-model and the CIN sub-model. It passes call requests from the C^2 sub-model to the CIN sub-model and passes the CIN responses back to the C^2 model.

4.2 Order of Events during Simulation

A simulation *session* consists of one or more simulation *runs* that form an experiment. Each run's input is a distinct random Call Script based upon a common Event Script and its output is a replicate of the experiment. The operator specifies a single call script for each session, but may specify a different CIN or CIN option for each run within the session. There are four phases to each simulation session.

- initialization for the session
- initialization for each run
- simulation for each run
- summary statistics for each run.

During the session initialization, the simulation reads in the trooplist file which contains information about the communicating parties, and the interface file which contains the data regarding which parties have access to which device. In initialization phase for each run, all the queues are made empty, all the parties, trunks, and devices are made idle and the statistical variables are initialized. Once the system has been initialized, the operator indicates which CIN sub-model to use for that run. Then the interface reads in the data regarding that sub-model being used. It is possible to have a large number of CIN sub-models and use a different one for each run.

As the simulation proceeds, the calls are read in one by one from the call script and are placed in the pending calls list. Each call attempt entry contains information regarding a call mainly divided into the following three parts

- call sequence number
- call data
- call label

The call sequence number is an ID to the call and can be used to access information regarding that call. It is assigned during the call's entry into the model. The call data includes origin and destination party indices, call length and priority, maximum number of retries, mean time between retries, initiation time and the deadline time. The label is an eight-character field that can be used during simulation or analysis to group calls.

While a call is pending, each attempt is recorded in the attempt record file. This file, generated during the simulation run, documents all system state changes and can be used with a statistical package to calculate required statistics.

A few summary statistics such as average calls in the model, average retries per call, and number of attempts blocked by the CIN sub-model are calculated at the end of each simulation run and written into a separate file. These are rough estimates of the system response which have been used mainly to test the model, but may be also used for preliminary analysis.

4.3 The C^2 Sub-model

The command and control sub-model closely emulates human behavior as a call is handled. Call status codes representing intermediate and final call states each unique situation. Correct transition requires only knowledge of the C^2 and call current

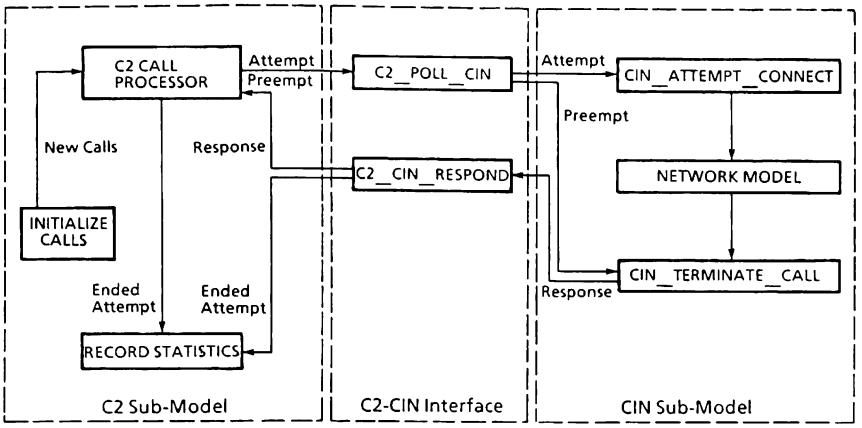

Figure 4. The Simulation Model

states because the states are Markovian. So although, a call's state may change many times in the process of handling a call, these codes allow consistent handling.

Figure 4 illustrates the inner working of the C^2 sub-model. INITIALIZE CALLS reads in each call from the call script at its simulated first attempt time and places it in the pending calls list, notifies the C2 CALL PROCESSOR which immediately attempts the call. The call is then transferred through the interface to the CIN sub-model. At this point, a number of things may happen depending on the status of the network. The CIN sub-model reports the result of the attempt back through the interface to the C2 CALL PROCESSOR. The C2 CALL PROCESSOR interprets the status codes returned from the network and determines the way a party or the network reacted to the call. The C2 CALL PROCESSOR reacts in a similar manner to subsequent attempts of calls that are not completed on the first try. As each attempt is resolved, the result of the attempt is recorded by RECORD STATISTICS.

Each possible intermediate or final status is represented by a unique call state. As a call is processed through the model, the pseudo-human reaction and the network response may cause the call's state to change often. Some states are terminating, but others may occur many times before reaching an ending condition. Once a call reaches a terminating state, it can undergo no further change, so it is dropped from the pending call list.

Figure 5 shows the possible states of a call at the C^2 level. The call states are nearly self-explanatory, but the reasons that a call changes state are not. Therefore how a call enters and leaves each state is explained below.

4.4 Terminal Classifications

A call has three possible final outcomes. Once a call reaches one of these states, it can have no further changes of state and is therefore removed from the ordered list of pending calls and is no longer retained in the model.

4.4.1 "Complete"

A call becomes a complete call when a connected call is finished, the parties hang up, and the network is cleared of the call. This triggers a search of the waiting call lists for the involved parties (see section *4.5.3*).

4.4.2 "Deadline Delete"

If a call remains incomplete beyond a certain time its information becomes worthless. Such a deadline is specified for each call on the basis of the Perishability field in the CDB. If the call is not completed by that deadline, it is classified as a deadline delete.

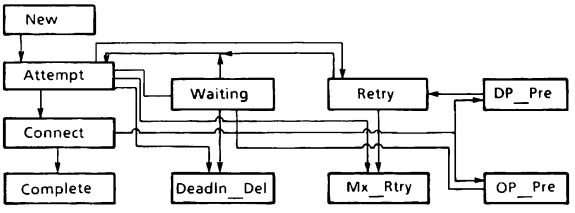

Figure 5. C^2 States and Possible Changes

4.4.3 "Maximum Attempts"

An originator would not retry a call indefinitely, but would give up after a number of attempts. If the number of attempts exceeds the maximum specified for call, then an attempt is not scheduled, and the call is classified as terminated because of maximum attempts.

4.5 Intermediate Classifications

A call can pass through many intermediate states before reaching a final classification. Each of these states could apply to a given call at several separate times while it is being processed through the model.

4.5.1 "New"

This status represents an originating party's first attempt at this call. Each call is read in from the call script at its simulated initial attempt time, is classified as a new call, and immediately reclassified as an attempt.

4.5.2 "Attempt"

An attempt is defined to be the initialization of a sequence of events by a participant wishing to communicate with a given party. In the C^2 sub-model, a call is classified as an attempt when any C^2 sub-process seeks to connect a call. An attempt status can arise from a new call, a waiting call, or retry call.

In an attempt, the origin is checked first. If the origin party is busy, the call is reclassified as a waiting call. If the origin party is idle, the CIN is polled. If the CIN sub-model reports that a connection is possible, the call's destination is checked. In the event that the destination is idle, the call is reclassified as a connect. If the destination is busy then the call is classified a retry.

4.5.3 "Waiting"

A waiting call is a call which is blocked by a source visible to the originator. Thus, the originating party will know when the obstacle dissolves. The C^2 sub-model classifies a call as waiting either because of a busy origin party or a busy origin instrument. Either the party or instrument can be busy because currently an equal, or higher-priority call is going through or either can become busy if the call gets preempted by a more important call.

Each time a call is classified as a waiting call, the number of attempts is compared to the number allowed for that call. If this is less than the maximum allowed, the call is then put on a list of waiting calls for the originator. If not, the call's state becomes "maximum attempts".

The list of waiting calls is logically a separate list for each party ordered first-come, first-serve according to priorities. When a call is completed and the parties become idle, the top call is pulled from this ordered list of waiting calls, and this call is reclassified as an attempt. If a call's deadline is reached while it is on this list, it is removed from the waiting calls list and reclassified a deadline delete termination.

4.5.4 "Retry"

A party may attempt to make a call but be blocked by some impediment not visible to the originating party. In that case, the originator will make another attempt for that call after a random amount of time. In the C^2 sub-model, a call is classified as a retry if an idle party can obtain a device to make the call, but cannot, for some reason, connect. For example, a call could be blocked by the lack of a connect path in the CIN or a busy destination device. It is also possible that a call that gets preempted or interrupted at the CIN level can become a retry.

Assuming the call has not exceeded its allowed number of attempts, the call goes on the retry list, and its next attempt is scheduled to occur in the future. If, however, its attempt limit is reached, the call is classified a maximum attempts termination.

A retry call becomes an attempt again when the simulation clock reaches call's scheduled retry time. Of course, a retry call can also be terminated as a deadline delete if its deadline is reached before its next retry time (see section *4.5.7*).

4.5.5 "Connect"

A connected call is one in which the origin and destination are currently conversing with one another. In the C^2 sub-model, a call is connected if a path is found through the network and the destination party is idle. The CIN sub-model is polled for a path, and the destination party is checked on the C^2 side. Only a call with an attempt status can become a connected call.

A connected call can have any of a number of things happen to it. If it stays connected long enough for all of its information to be transferred, it becomes a complete call. If a CIN interruption occurs, it can become either a waiting call or a retry call. It is also possible that a higher priority call can be attempted for either the destination or origin, in which case this connected call would become a C^2 preempt.

4.5.6 "CIN Interruptions"

A connected call may be interrupted because of some event in the CIN sub-model. Situations that cause an interruption include network failure and preemption of network facilities by a higher priority call. These interruptions can occur only to a call that is either attempting to find a path through the CIN sub-model or that is connected.

A call interrupted by the CIN will become a retry call unless the call origin's instrument or origin party becomes busy, in which case the call becomes a waiting call.

4.5.7 "C^2 Preempt"

A party to a conversation may realize that he or she must interrupt the current call to make a more important one. When a call with a high priority is attempted and involves a party busy with a lower priority call in connect status, the connected call becomes a C^2 preempt. Therefore, only a connected call can be preempted on the C^2 level.

The C^2 sub-model waits for the CIN sub-model to free the resources allocated to the call and then frees the parties to that connected call and reclassifies the disconnected call. If the call was preempted at destination, it's originator does not know when the destination will be free again. Thus, if the attempt limit is not exceeded, the preempted call will be classified as a retry.

If the call was preempted at the origin, the originating party will know when the preempted call can be attempted again. Thus, if the deadline has not been exceeded, the call will be classified as a waiting call. Of course, if the maximum attempts are exceeded or deadline for the preempted call is exceeded, the call is terminated with the appropriate classification.

4.6 The CIN Sub-Model

The main purpose of the ISU team is to develop the C^2 sub-model and the necessary supportive software. However, a requisite modeling task is a simplistic CIN sub-model, TN1, to assure that this C^2 sub-model works properly. This example CIN presents typical responses to the higher-level model. In particular, it

- possesses limited network resources
- produces random delays before confirming connections
- preempts network resources, if necessary
- generates random failures.

Thus, it also serves as an example of a compatible model to communications and computer experts.

As shown in Figure 6, the TN1 sub-model consists of two clusters of terminal equipment interconnected by a number of trunks. A terminal device is assigned to Cluster 1 if its phone number is odd and to Cluster 2 if it is even. The operator may select the number of trunk lines at run time as well as whether or not to enable network preempts and random failures.

The code for the TN1 sub-model, which can be found in [Kuo et al. 1990], contains many examples of ways to implement such models, but the basic rules are these:

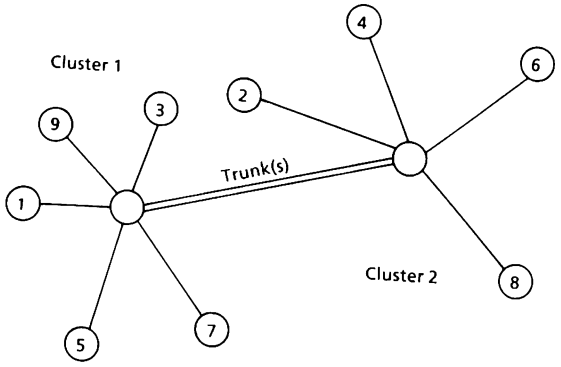

Figure 6. The TN1 Network Sub-model

- CIN sub-model identifiers may begin with anything except `C2_`, which is reserved for the C^2 sub-model.
- The C^2 sub-model communicates with the CIN sub-model only through the process `C2_POLL_CIN`.
- The CIN sub-model communicates with the C^2 sub-model only through the process `C2_CIN_RESPOND`.
- No shared (common) resources of the C^2 sub-model are shared by any of the CIN sub-models.

The situation implied by the last three rules is illustrated in Figure 4. Note that if a new CIN model is added, the CIN modeler will have to modify `C2_POLL_CIN`, but not `C2_CIN_RESPOND`.

4.7 The C^2–CIN Interface

The C^2–CIN interface is the sole interconnection between the C^2 and CIN sub-models. In addition, it acts as a filter, screening out responses from the CIN model that are inappropriate and therefore reducing the chance of a bug in a CIN sub-model causing subsequent difficulties in the C^2 sub-model. Finally, the interface contains the Baseline Reference sub-model.

As illustrated in Figure 4, all outputs to the CIN sub-model pass through `C2_POLL_CIN`. This process selects the current CIN sub-model and performs the necessary actions for that model. In this way, several CIN sub-models may be exercised during a single session. If a new CIN sub-model is added or significantly modified, this process may have to be modified.

All responses from the CIN are funneled through a process called `C2_CIN_RESPOND`. This module is very seldom changed, for its output must be compatible with the C^2 sub-model. To avoid sharing resources, the call index and status are both represented as passed indices. `C2_CIN_RESPOND` checks the CIN status codes, and if valid, takes the proper action in the C^2 sub-model. If the code is not valid, it generates an appropriate warning and ignores the input.

The Baseline Reference sub-model is essentially a perfect CIN. It behaves as if it has unlimited resources, performs all actions instantaneously, is 100% reliable, and completes all voice transmissions at the rate of 30 characters per second. This model provides a point of reference for a reasonable assessment of any CIN sub-model's performance.

For example, if a test reports a completion ratio of 70%, the CIN might not appear to be too good. But, if the completion ratio under the Baseline Reference sub-model is 80%, then a better measure of that CIN's performance is

$$\begin{aligned} R_r &= \frac{R_{CIN}}{R_B} \\ &= \frac{.7}{.8} \\ &= 87.5\% \end{aligned}$$

where R_r is the Relative Completion Ratio, R_{CIN} is the CIN Completion Ratio, and R_B is the baseline Reference Completion Ratio.

5. STATISTICAL ANALYSIS

Statistical analysis is necessary to extract important information from the simulation results. As mentioned earlier, the output from this simulation is mainly divided into two parts. One is the session record containing the summary statistics. The other is the call attempt log containing a record of each attempt.

5.1 Session Record

The session record indicates the files used for the run as well as the time and date the run was executed. It also contains some summary statistics calculated during the simulation. These summary statistics include event-dependent statistics, such as number of calls made by each party, number of calls blocked by a busy network, and number of calls that were successful, as well as time-dependent statistics such as average number of waiting calls for each party, and average number of calls in the system. The summary statistics are used mainly for rough checking and debugging, but may be used for preliminary analysis, too.

5.2 The Attempt Record

As each call attempt is resolved, a record is written into the call attempt log. Each attempt record consists of

- the call sequence number
- the attempt number
- the time at which the attempt was initiated
- the time at which the attempt was resolved
- the result of the attempt.

Note that when an attempt is connected, it is not yet resolved because it may be interrupted. Only when the attempt is either completed or interrupted is the result known. The result of each attempt is coded in the record as a two digit number to make analysis easier. The first digit gives general information according to the following table:

0	A terminating status.
1–4	A status from the C^2 sub-model.
5–9	A status from the CIN sub-model.

For example, code 70 indicates a CIN preempt, while code 30 indicates a C^2 preempt by the origin of the preempted call and code 02 indicates deadline delete. Many codes in the CIN range are unassigned to allow the CIN modeler to record specific information not important to the C^2 sub-model.

This output design allows independent performance measurement in the two sub-models. For example, if the first digit of the result code is 5,6,7,8 or 9, the call attempt was terminated due to some CIN action. Thus, the analyst may determine the ratio of attempts terminated by CIN action by computing the ratio of attempts with codes 5–9 to attempts with codes 1-9.

All changes in the C^2 sub-model's state are caused by either the initiation or termination of a call attempt. Since the attempt log records the initiation time, termination time and result of each attempt, it is a complete history of the C^2 sub-model's state. Following the same pattern, the CIN modeler can also determine each change of state. Thus, the entire history of the C^2 and CIN sub-models can be reconstructed from the attempt log.

5.3 Analysis

As shown in Figure 7 this attempt record file is part of a relational data base including the troop-list file, the needline reference file, and the call script file. The output file is indexed to the call script file by the call sequence number. The call script file is indexed to the troop-list file and the needline reference file through the party and needline indices respectively. This structure of information in the four files allows efficient data storage and analysis of the simulation run.

The call attempt label in the call script also helps in the analysis. This field is completely under the control of the modeler and can be used to achieve a wide range of effects. For example, one column could be a flag which indicates the calls of highest interest to the analyst. The analyst can use this flag to easily make a sub-file containing only these calls.

Every simulation creates a new attempt record file, and it is very important that the correct set of files is used for the analysis. To prevent the analyst from using the wrong set of files, C3INAT internally labels these files in the first record. This record is in a format compatible with the other records.

The first record in the call-script file, the troop-list file, and the needline reference file contains the respective file name and

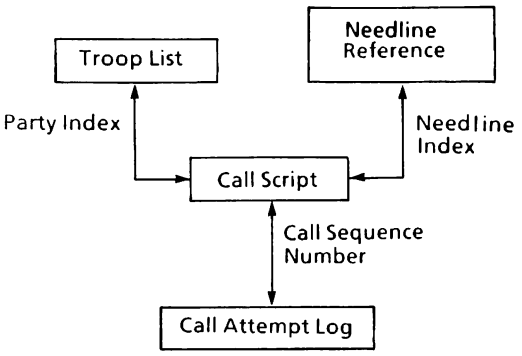

Figure 7. The Relational Data Base of C3INAT

the date and time at which the file was last updated. The first record in the attempt record file contains the time at which the run was initiated, and that time is also written in the file of summary statistics. The session record also indicates the call-script, troop-list, and the needline reference files used for the run.

A large variety of statistics can be calculated from the information given in the attempt log. These statistics can be time-dependent or event-dependent. The event-dependent statistics are averaged over the number of occurances while the time-dependent statistics are averaged over time. Since both events and times are recorded in the attempt log, either type of statistics relating to parties, the sub-models, and the model as a whole can be calculated.

Because all records in a given file are compatible in format, virtually any database or statistical analysis package can be used for analysis. The package considered include SYSTAT, SAS, NAG, dBASE III, and MINITAB.

6. CONCLUSION

The stratified approach used in C3INAT shows great promise. By using the principle of information hiding, the complex C^3I design problem is reduced to two less complex problems. The use of a well-defined interface allows modeling teams to work independently with a high assurance of compatible results.

A key factor to this approach is the recognition of the different scope of information needs in the two sub-models. C3INAT contains tools that facilitate data management in each context. Also, the C3INAT is designed to allow teams to work in either context during modeling or analyses yet enforces coordination among the efforts.

The final significant feature of C3INAT is the implicit recognition of the need to communicate relevant needs and performance factors between the sub-models. This serves to predict C^3I effectiveness more reliably than more loosely-coupled designs.

ACKNOWLEDGEMENTS

This work has been sponsored in part by the U.S. Army Communications-Electronics Command at Ft. Monmouth, N.J. Additional support was supplied through the Industrial and Manufacturing Systems Engineering Department at Iowa State University.

REFERENCES

Athans, M. (1987), "Command and Control (C2) theory: a challenge to control science." *IEEE Transactions on Automatic Control, AC-32*, 4, 286-293.

Bleistein, S., S. Chow, and R. T. Goettge (1985), "Analytic performance model of the U. S. en route air traffic control computer systems." *ACM, 13*, 2, 105-115.

Bohannan, A. G. (1985), "C3I in Support of the Land Commander," In *International Conference on Advances in Command, Control and Communication Systems*, IEE, Bournemouth, Dorset, England,1-8.

Farrell, R. J., S. Bonder, L. P. Proegler, G. Miller, and D. E. Thompson (1986), "Capturing Expertise: Some Approaches to Modeling Command Decisionmaking in Combat Analysis," *IEEE Transactions on Systems, Man and Cybernetics, SMC-16*, 6, 766-773.

Georgiou A. S. and G. H. Lammers (1985), "C3 Effectiveness Studies,"In *International Conference on Advances in Command, Control and Communications Systems*, IEE, Bournemouth, Dorset, England, 80-85.

Johnson, K., Y. Wang, J. Mullen, and W. Kuo (1988), "Database Needs for Modeling C3I System Performance," In *Proceedings of the 1988 Command and Control Research Symposium*, IEEE, Monterey, CA, 526-530.

King, T.E.G. and A. F. Martin (1985), "The Modeling of Communications Networks," In *International Conference on Advances in Command, Control and Communications Systems*, IEE, Bournemouth, Dorset, England, 90-98.

Kuo W., J. Mullen, J. Rupe, S. Gavirneni, and F. Lin (1990), "C3 Information Network Analysis Tool (C3INAT): Final Report," Technical Report, Prepared for the U. S. Army Communications-Electronics Command and Fort Monmouth, NJ. (in process).

Kroening, D. W. (1986), "Army Command and Control Information System Requirements Definition," *IEEE Transactions on Systems, Man and Cybernetics, SMC-16*, 6, 974-979.

Krishna, C. M. and K. G. Shin (1987), "Performance Measures for Control Computers," *IEEE Transactions on Automatic Control, AC-32*, 6, 467-473.

Ripley, B. (1987), *Stochiastic Simulation*, John Wiley & Sons, New York, NY.

Schwartz, M. (1987), *Telecommunication Networks: Protocols, Modeling, and Analysis*, Addison-Wesley, Reading, MA.

Sweet, R., D. Mensh, P. Gandee, I. Stone, and K. Briggs (1986), "The Modular Command and Control Evaluation Structure (MCES): Applications of and Expansion to C3 Architectural Evaluation," Technical Report, Naval Postgraduate School, Monterey, CA.

Wohl, J. G., E. E. Entin, D. L. Kleinman, and K. Pattipati (1984), "Advances in Man-machine System Research," *Human Decision Processes in Military Command and Control, 1*, 261-307.

Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

A CORPORATE STRATEGY FOR IMPLEMENTING PROCESS SIMULATION

Douglas J. Payne
Nandu N. Thondavadi
Phil Francis

Corporate Technology Center
Square D Company
1415 S. Roselle Road
Palatine, Illinois 60067

ABSTRACT

Simulation modeling of complex manufacturing systems has been recognized as a critical link to a successful CIM strategy. Square D Company has embarked on an ambitious program of institutionalizing modern methods of manufacturing practices in its plants worldwide by forming the Corporate Technology Center (CTC), a center of excellence with the charter to provide product and process related advanced technology to Square D worldwide.

While many fortune 500 Companies are debating whether such programs should be centralized or decentralized, Square D CTC has conceived an implementation approach that may work for many other companies as well.

Within a short 8 months of existence, CTC has far exceeded its original goals in institutionalizing simulation methodologies in over 12 manufacturing locations. The acceptance and results have been overwhelming. The challenge ahead is to keep the momentum going, quantify benefits from such advanced practices, and, most of all, make simulation a 'way of life' on the plant floor.

1. INTRODUCTION

Our manufacturing environment today is complex and dynamic. To deal with this complexity, corporate manufacturing strategies are being established based on concepts such as JIT, synchronous manufacturing, time based manufacturing, and total quality management. In this environment, process simulation is being relied on more and more to help manufacturing management understand the implications of these concepts and for identifying and testing specific implementation strategies. The objective of this paper is to describe Square D Company's corporate simulation program and to outline the specific steps which were taken during its implementation.

This paper begins by describing the manufacturing environment at Square D Company and the motivation for using simulation. With this as a base, an eight-step approach for initiating and managing a corporate simulation program is described, and a summary of program results is included.

2. BACKGROUND

Square D is a worldwide supplier of products, systems, and services for the distribution, application, and control of electrical energy. The company currently operates approximately 55 manufacturing plants worldwide producing a variety of electrical distribution, control and sensing products. Over the last two to three years, Square D has used simulation to address issues relating to the specification and design of plant automation. But, each application was isolated and narrowly focused. In 1989, the Corporate Technology Center (CTC) was formed, with its charter to serve Square D by, among other things, institutionalizing modern/advanced design and manufacturing practices. Process and business simulation, synchronous manufacturing, time-based manufacturing, and activity based costing were identified as key practices which the CTC was to implement. The rest of this paper describes the specific implementation of the corporate simulation program.

The basis for Square D's simulation program comes from the recognition that the benefits of simulation fall into two categories: 1) those resulting from using a specific model and 2) those resulting from the practice of simulation methodology. Types of benefits falling into the first category are:

- Identification of cost avoidance during system specification and design
- Evaluation of operational strategies during system design
- Early evaluation of projected schedule/model mix
- Quantification of impacts of proposed system modifications and changes in manufacturing environment

(These types of benefits have traditionally been used to sell simulation.)

The second category of benefits are generally difficult to quantify but, in fact, can have a much greater impact than those in category one. This second type of benefit results from using simulation to increase plant involvement and to facilitate planning of the many aspects of designing and operating a manufacturing plant. Simulation provides a methodology for describing a plant system and forces a system-wide understanding. In order to "successfully" build any simulation model, many steps must be followed:

1. An objective must be defined.
2. The system must be described and flowcharted.
3. An appropriate level of detail must be selected and continually re-evaluated which will allow the user to meet the model objective.
4. Data must be gathered and analyzed which accurately reflects the system to be modeled.
5. The model must be built, verified and validated.
6. Experiments must be designed and conducted.
7. Documentation must be completed.

From Square D's perspective, the key aspect of this methodology is that it involves everyone—from plant manager to operator on the floor. In the Product/Process Design Phase, simulation supports the integration of manufacturing and engineering and thus supports simultaneous/concurrent engineering. In the System Design Phase and Operational Phase, simulation facilitates a team approach and creates a focus on system-wide optimization.

The objective of the simulation program was simply to maximize both types of benefits described above. To achieve this objective, three goals were set:

1. To have a self-sustaining simulation capability at the plant/divisional level.
2. To maximize involvement within each plant.
3. To have simulation become a standard practice ('way of life') within Square D.

With the objective and goals as a basis, an implementation plan was established. Throughout the past year, this plan has been revised and improved and will be described in the remainder of this paper.