Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

The project approach to simulation language comparison

F. Bradley Armstrong
Staff Engineer
Hughes Aircraft Company C&DP
P.O. Box 9399, LB/C05/2016
Long Beach, California 90810

and

Scott Sumner
Member of the Technical Staff
Hughes Aircraft Company S&CG
P.O. Box 92919, SC/S65/J323
Los Angeles, California 90009

ABSTRACT

We present a new approach to comparing simulation software which is based on the use of a representative project. We introduce our approach and the need for it. We then demonstrate it by considering Space and Communication Group's dock-to-stores system and six simulation languages (MAP/1, SIMAN, SIMFACTORY, SLAM II, WITNESS, XCELL+). Finally, we summarize our results and provide a perspective on the work performed.

1. INTRODUCTION

The application of simulation as an analysis tool to manufacturing problems within Hughes Aircraft Company (HAC) has dramatically increased in the last few years. Interestingly, new users are predominantly manufacturing personnel themselves instead of dedicated simulation staff. As a consequence, the frequency of language selections and purchases by newcomers having little or no background in simulation has also risen. These purchasers, usually under tight time constraints and without guidance, sift through vendor literature and demonstrations to determine which language best suits their needs. Although some guidelines have been published such as Haider and Banks (1986), all address the problem generally and fail to offer environment specific assistance.

It is difficult to make meaningful language comparisons ***with respect to the individual user's needs*** because the languages do not readily lend themselves to direct comparison. For novice users the problem is compounded because most do not yet know how to identify their needs and translate them into the set of modeling features they require. To address this problem, we present and demonstrate a new approach to comparing different simulation languages which uses a representative project. This paper presents our approach and then demonstrates it by applying it to a HAC project and six simulation languages. We then summarize our results and provide a perspective on the work performed.

2. THE APPROACH

2.1. Approach Justification

Many newcomers believe that the success of a simulation project depends solely on the selection of the "best" language. As with any selection process, the different languages must be compared in some manner. A first approach is to get literature from the various vendors and make comparisons based on language features. There are two difficulties with this approach. First, comparable information is difficult to obtain because different vendors emphasize different features in their literature. Second, because of different implementations, a feature of one language may be a subset or superset of a feature in another language. Worse still, it may even have some overlap where it serves parts of several features of another language.

Once this situation becomes apparent, some newcomers simply get a recommendation from a current simulationist in their organization. Although this can lead to standardization which has many benefits, it also has some drawbacks: adoption, time, and incompatibility. In our experience, once someone selects a language and becomes a user, he tends to adopt the language and defend it, sometimes consciously or unconsciously ignoring any drawbacks of that language. Second, because languages are constantly being enhanced, evaluations have a time window of applicability. Users who have made previous evaluations may not now have a working knowledge of new languages or the latest enhancements to established languages. Finally, if their applications differ from those of the newcomer, the language recommended may not be suitable.

We believe newcomers do themselves a great disservice if they do not acclimate themselves to simulation by performing an evaluation. We propose a comparison approach based not on language features but on a representative project. By identifying the features necessary to satisfy the requirements of the project, a common ground is established for comparison. Not only does our approach provide a meaningful methodology, but it also forces newcomers to gain an understanding of the simulation process.

2.2. Approach Description

Our approach determines which languages should be considered further for selection. It eliminates those languages that do not meet the standards of a project-derived template. We expect the final selection to ultimately depend on factors that fall outside the scope of what we propose here. The approach consists of five steps: 1) establish ground rules, 2) select candidate languages, 3) select a representative project, 4) build an evaluation template, and 5) evaluate each language. Because creating a template requires the newcomer to think through a project in detail, it is the key to this approach.

ESTABLISH GROUND RULES:

The first step in our approach establishes the ground rules; it requires users to think through their objectives, goals, and constraints. These ground rules focus the comparison and minimize false starts. If ground rules are laid out carefully, the remaining steps are straightforward.

SELECT LANGUAGES:

Once the ground rules for language selection are complete, the selection of languages for comparison becomes a process of elimination. Ground rules governing hardware environments, cost, support, etc., can make the process simple. Each language considered represents a significant investment of time so it pays to keep the selection as limited as possible.

SELECT REPRESENTATIVE PROJECT:

Because our approach uses one project to make comparisons, the importance of choosing one that is representative can not be overemphasized. We recommend the coverage be as limited as possible to ensure the appropriateness of the project. However, because completing an evaluation template requires research (i.e., studying the users manuals to find out if the language can model the system), newcomers will learn more than just the information needed for the project. The project provides a focus when reviewing the literature.

BUILD EVALUATION TEMPLATE:

The languages considered in this paper are all evaluated using the same evaluation template, consisting of:

1. **General description:** general information about the language
2. **Model construction:** In most models, only a few key points require special attention. The evaluation should focus on these points because they highlight where modeling with a particular language may prove difficult or impossible.
3. **Model input:** The user needs to consider the interface provided for model construction; it determines how comfortable the user will be with the language.
4. **Model execution:** The user should consider those language features that support model execution with respect to the analyses to be performed.
5. **Model output:** The user must evaluate the outputs he will require from the model to achieve project objectives. These outputs may or may not be available from a particular language.
6. **Other features:** Items of interest may arise during comparisons that should be noted even though they are not covered by the other categories.

EVALUATE LANGUAGES:

The completed template is essentially a set of requirements the simulation language will have to satisfy. It might be tempting to simply eliminate any language that does not fill all of the requirements; however, this would discount other benefits gained from studying the language. Rather, we suggest the user consider each language failure using the perspective gained from all information learned; the ability to temper the quantitative tests with qualitative information gives the truest measure of a language's suitability to a specific user in a specific environment.

2.3. Limitations

By considering a single project, newcomers run the risk of overlooking desirable language features not highlighted by the selected material. Our approach, by its very nature, focuses on the features that are most important to the project being modeled -- to the exclusion of all other language features. Therefore, although this approach will make an excellent departure point for language selection, it is specific to the evaluation environment.

3. DEMONSTRATION APPLICATION

3.1. Establish Ground rules

The languages for this evaluation meet four requirements. First and foremost is their availability to us. Fortunately, because HAC is large and its different Groups run independently, at least one purchased copy of almost every commercial simulation language is available within the company. Therefore, this restriction will not limit our effort. Second, the languages must have manufacturing-specific features. Third, we impose a "no-coding" restriction on our model building. For our evaluations, we define coding as the use of a high-level programming language such as FORTRAN. The last restriction involves the host environment: all languages must run on personal computers and/or workstations. These restrictions are important to us because available, non-coding, manufacturing-tailored software, coupled with the free computer usage provided by stand-alone environments, has been a major factor in the recent appeal of simulation at HAC. In addition to these requirements, we restrict ourselves to considering each language as purchased. This means we will not consider other available support software that enhances the language.

Although most newcomers are interested only in applications within their own areas, we desire a project that has appeal to Hughes as a company. It should be a representative system within a HAC Group (but common to a majority of them), familiar, and of general interest. We include these ambitious ground rules to realize maximum HAC coverage because of corporate level project funding.

Since language updates are continually being released by vendors, we realize our evaluation is a "snapshot" of the marketplace. We use the most current version of the software that we have in-house, regardless of whether it is the current release. We also limit ourselves to selecting five key points for the model construction portion of the template.

Language evaluation for this demonstration is limited to the completed language templates; broader evaluation would be biased for the following reasons:

1. Uniqueness of each project environment
2. The influence of our previous modeling experience

These unique circumstances are crucial to every evaluation, so they cannot be included in this demonstration. Therefore, we recommend newcomers use the approach presented here -- not the results.

3.2. Select Languages

Based on the ground rules for languages, the following languages are included in this evaluation: MAP/1, SIMAN, SIMFACTORY, SLAM II, WITNESS, and XCELL+. Although four of the languages allow coding, we do not use this feature when building our models.

3.3. Select Representative Project

The following project meets the criteria specified in the ground rules for. project selection and has many features common to many manufacturing projects.

Hughes' Space and Communications Group has a central receiving dock for the receipt of material. The workers in this area are responsible for correctly receiving arriving hardware, testing it (if required), and routing it to its appropriate stores area for use. Parts arrive in "lots" which travel through the system together. End users can expedite certain lots through the system with a "line-stop request." Once flagged, these lots receive priority over others that may not be needed as soon. A simplified flow diagram of the system is shown if Figure 1.

Once received, a lot of flight parts is classified as either "Fab," "Normal flow," or "Hi-reliability":

- "Fab" (fabricated) lots are sent to the Precision Mechanical Inspection (PMI) area. Once tested, they are either accepted or rejected.
- "Normal flow" lots are sent to the Mechanical/Visual (M/V) Inspection area. They are accepted or rejected here.
- "Hi-reliability" (HR) lots face up to three separate tests:

1. Particle Impact Noise Destruct (PIND) analysis
2. Electrical testing
3. Mechanical/Visual inspection

In addition, samples may be split off to go through Destruct Physical Analysis (DPA) and sometimes also Radiation (RAD) testing. HR lots without samples can be accepted or rejected at the conclusion of Mechanical/Visual Inspection, but lots with samples must wait for their test results at Production Control

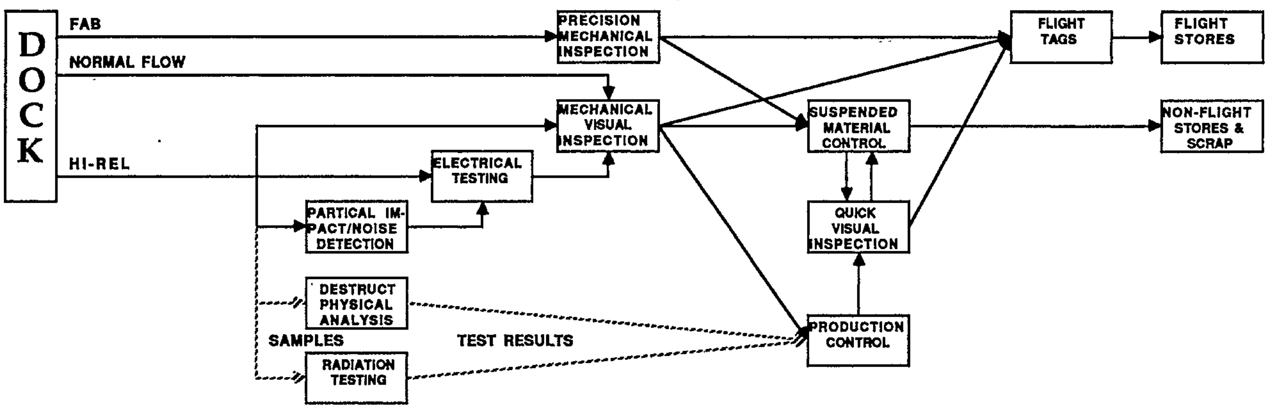
DOCK
FAB
NORMAL FLOW
HI-REL
PRECISION MECHANICAL INSPECTION
MECHANICAL VISUAL INSPECTION
ELECTRICAL TESTING
PARTICAL IMPACT/NOISE DETECTION
DESTRUCT PHYSICAL ANALYSIS
RADIATION TESTING
SAMPLES
TEST RESULTS
SUSPENDED MATERIAL CONTROL
QUICK VISUAL INSPECTION
PRODUCTION CONTROL
FLIGHT TAGS
FLIGHT STORES
NON-FLIGHT STORES & SCRAP

Figure 1: Flow Diagram of Hughes S&CG Dock-to-Stores System

(P/C) before being sent to Quick Visual (Q/V) inspection, where a final evaluation is made.

Accepted lots are routed to the Log-Out area, where flight tags are affixed to them. Rejected lots are sent to the Suspended Material Control (SMC) area, where a disposition is made. Problems fall into two categories:

1. Paperwork problems involve collecting the appropriate documents and correcting the deficiencies found.
2. Engineering problems suggest a discrepancy with a lot itself. These lots may be dispositioned either:

    a. Use as is: in spite of the discrepancy, use the lot
    b. Downgrade or scrap: downgrade the lot to "non-flight" status, or scrap it
    c. Return to vendor: return the discrepant lot to the vendor for rework

    Reworked lots return to the Suspended Material area, where they may then be re-dispositioned as acceptable or unacceptable. Acceptable lots are sent through the Quick/Visual area to stores; unacceptable lots may face downgrade or another trip back to the vendor.

Lots may only pass through SMC once; no lots may return to be reconsidered.

3.4. Build Evaluation Template

The languages considered in this paper are all measured against the evaluation template below; the format of this template has already been described (see **Approach Description**, above).

GENERAL DESCRIPTION:

To provide an overview of the product, we report the following information:

1. Name
2. Type
3. Host environments
4. Software version number
5. Any artificial restrictions placed on the product
6. Vendor

MODEL CONSTRUCTION:

DPA: We attempt to model the following characteristics of the HAC Destruct Physical Analysis (DPA) lab:

1. 135 lots can be concurrently processed in-house
2. The DPA in-house waiting area is limited to 100 lots
3. Lots are offloaded to an off-site vendor if there is no available storage in-house
4. Off-site facilities can handle any number of lots concurrently
5. Processing times in-house and off-site are both variable, but are not the same

Line-Stops: We attempt to model the following characteristics of "line-stopped" (LS) lots :

1. 10% of arriving Fab lots are designated LS
2. 10% of arriving Normal Flow lots are designated LS
3. 20% of arriving High Reliability lots are designated LS
4. LS lots are always processed before NLS lots

Personnel: We attempt to model the following characteristics of dock-to-stores personnel:

1. Six teams of personnel dedicated to a single function (Normal flow/Fab. receiving, HR receiving, PMI, in-house DPA, RAD, and Log-Out)
2. Three teams of personnel shared between functions (M/V and Q/V personnel, PIND and Electrical test personnel, and SMC personnel)
3. LS lots are processed before NLS lots; ties are broken based on function

Samples: We attempt to model the following characteristics of those HR lots with samples:

1. 32% of arriving HR lots have DPA samples only
2. 10% of arriving HR lots have both DPA and RAD samples
3. Separation of samples from their lots
4. Disjoint routing of samples and lots
5. Reuniting sample test results with their lots

Testing/Inspection: We attempt to model the following characteristics of the inspection points in the dock-to-stores system:

1. Determination of whether the lot passes or fails an inspection
2. Alternative remaining routing based on whether the the lot passes or fails

MODEL INPUT:

We consider the following features of model input:

1. Non-coding input format
2. Model can be built as desired within size constraints

MODEL EXECUTION:

Because of the stochastic nature of our system, we consider the following execution features:

1. Statistics clearing at a specified time
2. Random number seed changes
3. Multiple runs

MODEL OUTPUT:

To satisfy our project objectives, we require the following output information:

1. Flow time statistics for flight and non-flight lots
2. Throughput for flight and non-flight lots
3. Utilization statistics for all personnel in the system
4. Work-in-process statistics for DPA, RAD, and Production Control
5. Input echo

OTHER FEATURES:

This section contains information discovered during the evaluation process that does not fit into any of the other categories.

3.5. Evaluate Languages

Our ground rules state that we evaluate languages for this project solely against their completed templates.

4. THE EVALUATIONS

4.1. General Comments

Throughout the evaluations, the following abbreviations are used:

1. LS = Line-Stopped
2. NLS = Non-Line-Stopped
3. HR = High-Reliability
4. DPA = Destruct Physical Analysis
5. RAD = Radiation testing
6. PMI = Precision Mechanical Inspection

When discussing the models, language-specific terminology is **bolded** the first time it appears.

4.2. MAP/1 Evaluation

GENERAL DESCRIPTION:

MAP/1 is a manufacturing simulator and is designed specifically for manufacturing applications. It runs on workstations but not personal computers. MAP/1 Version 3.1 and Version 1.0 of the Interactive Input System are used for this evaluation. It is offered and supported by Pritsker and Associates.

MODEL CONSTRUCTION:

DPA: We use two **regular stations** to model the in-house and off-site DPA. The in-house station has a **preprocess inventory** of 100 and a size of 135, and the off-site station has a preprocess inventory of 999 and a size of 999. We model the offload logic by conditionally **routing** lots based on whether the in-house preprocess inventory is full. Both stations specify processing times by sampling from triangular distributions but have differing parameter values.

Line-Stops: We use probabilistic routing to determine whether an arriving lot is to be LS. The Fab. and Normal flow lots are routed to one of two dummy **produce stations** where they are renamed to reflect their status. HR lots are discussed under **Samples**, below. NLS lots are assigned a priority of 1, and LS lots a priority of 2. The preprocess inventories of the PMI, RADTEST, DPATEST, and LOGOUT stations are ranked by highest priority. The MAP/1 allocation policy polls stations in a fixed order but searches for a waiting lot regardless of its type (i.e., ignores whether the waiting lot is LS or not). Consequently, for those processes which require shared personnel, we use pairs of stations to segregate waiting LS lots from NLS lots.

Personnel: We model dedicated personnel implicitly using **regular stations**. Because dedicated personnel process the lot the entire time it is at a station, the personnel and station utilizations are the same. We explicitly model shared personnel using **personnel** and **operators**. For each personnel class, we use a personnel statement/form to specify the number of operators and the stations they support. The list of supported stations also specifies lot precedence in the event lots at several different stations are requesting personnel. For each station, we use an operator statement/form to define the station personnel requirements.

Samples: As shown in Figure 2, we determine the number of samples associated with an arriving lot using probabilistic routing from a dummy station named HRTYPE. After determining whether the lot is LS or NLS, the lot is routed to a produce station where the output is the lot and the appropriate samples. We use an **assemble station** to reassemble the one-sample and two-sample lots with the samples having the same lot type (e.g., a LS, one-sample HR lot is always reassembled with a DPA sample that was split off from a LS, one-sample lot). However, there is no way to guarantee a sample split from a lot will be reunited with its original lot.

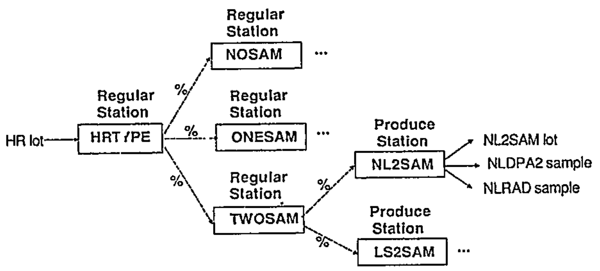

Figure 2: Flow Diagram of MAP/1 HR Lot Identification

Testing/Inspection: We use probabilistic routing at the points in the system where a lot is to be accepted or rejected. This routing allows us to specify different destinations for the lot based on the outcome. When routing HR lots with samples to Quick Visual, we use **station operation numbers**. An untested HR lot with samples is routed to operation 1 at station Quick Visual which specifies probabilistic routing (pass/fail) after processing. If the lot subsequently returns, it is routed to operation 2 which specifies deterministic routing (pass) after processing.

MODEL INPUT:

When using a workstation that does not have the VAX/VMS operating system, MAP/1 models are built by creating a statement file using a standard text editor. However, under VAX/VMS a forms input system is also available. Unfortunately, the two input methods are incompatible with each other.

MODEL EXECUTION:

Statistics clearing, random number seed changes, and multiple runs can all be specified before executing the model by using the **clear**, **seed**, and **begin** statements, respectively.

MODEL OUTPUT:

MAP/1 collects flow time statistics automatically for each lot named in the model. Unfortunately, the determination of whether lots go to flight or non-flight stores is random and occurs after the lots have spent significant amounts of time in the system under a variety of names. Therefore, it is not possible to collect this data without convoluting the model by predispositioning the lots upon entry. MAP/1 supplies all other required output automatically.

OTHER FEATURES:

None noted.

4.3. SIMAN Evaluation

GENERAL DESCRIPTION:

SIMAN is a general purpose simulation language which has manufacturing specific features. It runs on both personal computers and workstations. The statement portion of SIMAN Version 3.5 is used for this evaluation (i.e., no user code is allowed). It is offered and supported by Systems Modeling Corporation.

MODEL CONSTRUCTION:

DPA: We use two **resources** to model the DPA function. The in-house resource has a capacity of 135 and the off-site resource has capacity of 999. As shown in Figure 3, the lots go through a sequence of blocks which includes: 1) going to a **queue block** to wait for a resource; 2) going to a **seize block** where the resource is seized; 3) going to a **delay block** where the DPA processing occurs; and 4) going to a **release block** where the resource is released. There are two sets of these blocks, one for in-house and the other for off-site. The queue block for the in-house DPA Queue has a capacity of 100, and the queue block for the off-site DPA queue has a capacity of 999. The two delay blocks both specify a processing time based on samples drawn from a triangular distribution but have different parameter values. We model the offload logic by sending all arriving lots to the in-house queue block which, if full, specifies lots to balk to the off-site queue block.

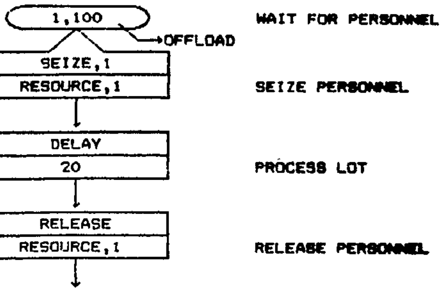

Figure 3: Example of SIMAN Resource Modeling

Line-Stops: We determine whether an arriving lot is to be LS or not using probabilistic branching. **Attribute** 2 is set equal to 2 for LS lots, and set to 1 for NLS lots. Lots waiting in the PMI, DPA, RAD, and LOGOUT queues are ranked high value of attribute 2 first. When lots of different seize blocks request a resource (i.e., shared personnel), SIMAN determines precedence using **seize block priorities** which consider the queue blocks but not the lots waiting. Therefore, we use pairs of queue and seize block sets to segregate the LS and NLS lots with seize block priorities set to favor LS lots.

Personnel: We model both dedicated and shared personnel using resources. For shared personnel, we use the seize block priorities to indicate lot precedence in the event lots at several different seize blocks are requesting the resource.

Samples: We determine the number of samples associated with an arriving HR lot probabilistically at a **branch block**. We then separate the samples from their lots using a branch block which specifies the lot and its samples as outputs. We rematch lots with their samples using **match blocks**. When the samples and lots have finished their independent routings, they are placed in one of four queues: 1) Lots, 2) DPA test results for one-sample lots, 3) DPA test results for two-sample lots, or 4) RAD test results. One match block reunites the two-sample lots with their samples' test results, and the other match block rematches the one-sample lots with theirs. The match blocks reunite lots and samples by matching their arrival time to the system (their attribute 1), thereby ensuring samples are reunited with their original lots.

Testing/Inspection: We use branch blocks having probabilistic branching at the points in the system where a lot is to be accepted or rejected. We model the different routings of failed and passed lots by sending the different branch block outputs to different blocks. For HR lots with samples at Quick Visual, we use an attribute and conditional and probabilistic branching. Attribute 2 equals 3 for HR lots with one-sample, and 4 for the two-sample lots. After Quick Visual we use a branch block to probabilistically branch lots having attribute 2 equal to 3 or 4 to determine whether the lot passes or fails. Lots having attribute 2 values other than 3 or 4 are passed. If the lot fails Quick Visual, its attribute 2 is reassigned to 5 and if it returns it will be passed.

MODEL INPUT:

SIMAN models are built by constructing two files, one containing the system description (model file) and the other containing the data values (experiment file). Both of these files are built in statement form using any standard text editor. Model size is limited if the model is to execute on a personal computer.

MODEL EXECUTION:

Statistics clearing, random number seed changes, and multiple runs can all be specified before executing the model by using the **replicate** (clearing and multiple runs) and **seeds** (seeds) **elements.**

MODEL OUTPUT:

Flight and non-flight flow time statistics and throughput are collected using two **tally blocks** and two **tally elements.** Utilizations and work-in-process statistics are collected using a **dstat element.**

OTHER FEATURES:

SIMAN has two support products which provide graphical (Blocks) and forms input formats (Elements) but because they are distinct products they are not considered in this evaluation.

4.4. SIMFACTORY Evaluation

GENERAL DESCRIPTION:

SIMFACTORY is a manufacturing simulator and is designed specifically for manufacturing applications and is offered and supported by CACI. It runs on personal computers but not workstations. Version 1.6a is evaluated in this paper.

MODEL CONSTRUCTION:

DPA: We use two **chamber stations** to model in-house and off-site DPA processing. The in-house station has a capacity of 135 and the off-site station has a capacity of 999. We use chamber stations because they are the only type of station that allows lots to process concurrently. We use two **queues** to model DPA storage. The in-house queue has a capacity of 100, and the off-site queue has a capacity of 999. As part of the **partflow network**, we model the offload logic by using two 1-way links emanating from the station where the DPA samples are split off from their lots to queues which feed the in-house and off-site DPA processing stations. A sample will always go to the in-house DPA queue unless it is full due to the order in which we input the links. Because the processing time for an operation is associated with the lot and not the station where the operation is performed, there is no way to specify different distributions for the in-house and off-site DPA operations.

Line-Stops: We determine whether an arriving lot is to be LS by ending its **process plan** probabilistically. Each type of arriving lot generates two new process plans. The LS process plan assigns LS lots a priority of 2, and the NLS process plan assigns NLS lots a priority of 1. Lots waiting in the PMI, RAD, DPA, and LOGOUT queues are ranked based on highest priority. For shared personnel, requesting lots go to one of several queues depending on their status and their **operation** requiring the resource. The lots then wait to be pulled into a **normal station** for the requesting operation. We force this station to poll its queues in a fixed order by specifying a fixed order **upstream selection rule**. The splitting of lots into different queues is necessary since the SIMFACTORY **foreman** uses a station's upstream selection rule and searches for the first non-empty upstream queue linked to it regardless of whether the waiting lot is LS or NLS.

Personnel: We model dedicated personnel implicitly using chamber stations since dedicated personnel process the lot throughout the operation and there is only one operation having a duration associated with each station. We model shared personnel explicitly using resources since they are needed for several different operations. As shown in Figure 4, we control resource allocation by forcing all lots requiring the same shared personnel to go through a common dummy **normal station operation** to request the resource before going through their respective chamber station operations.

MODEL OUTPUT:

Personnel utilization and WIP statistics are automatically reported as station utilization and queue statistics, if these reports are requested. Because we implicitly model dedicated personnel using chamber stations, and because utilizations are reported by stations, we avoid having other operations at these stations -- they would bias the results. Flow time and throughput statistics are also automatically reported for all lot names having terminating process plans (i.e., flight and non-flight lots). Even though these lots flow through the system under a variety of process plans, flow time statistics are based on when the lot first arrived to the system.

OTHER FEATURES:

SIMFACTORY has concurrent animation capabilities.

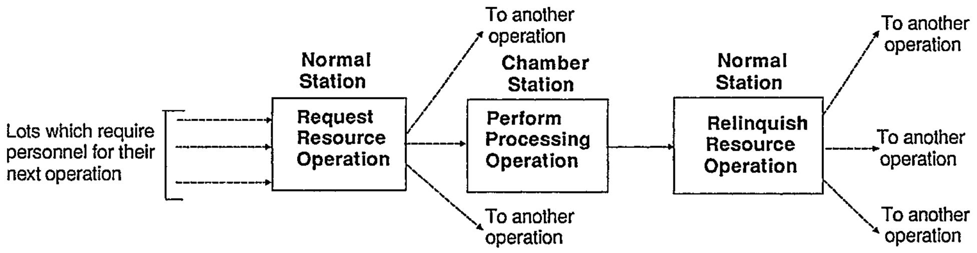

Figure 4: Example of SIMFACTORY Resource Modeling

Samples: We determine the number of samples associated with an arriving HR lot by ending its process plan probabilistically. In turn, the resultant process plans for the HR lots with samples end with multiple outputs (i.e., the lot and its samples). The only modeling restriction for this splitting is that the HR lot exiting has to have a name different from the name used for the input. The process plans for reunited lots start with a reuniting normal station and has the samples and the lots as inputs and the renamed lot as output. Because each type of lot and sample are named differently, lots and samples of the same type are reunited (e.g., LS one sample HR lots are always reunited with DPA samples that were split off from LS, one-sample lots). However, there is no guarantee that samples reunite with their original lots.

Testing/Inspection: We end a lot's process plan probabilistically at the points in the system where a lot is determined to be either accepted or rejected. The last operation occurs at dummy normal stations instead of the chamber station where the inspection operation occurs because we relinquish the resource before ending the process plan. We represent the different routings of passed and failed lots by specifying different process plans. For HR lots with samples at Quick Visual, the process plans for untested lots end probabilistically (pass/fail) while the process plans for failed lots end by being renamed passed lots.

MODEL INPUT:

Models are input using a forms system. SIMFACTORY requires the construction of a graphical layout as part of the model building process. Model size is limited if the model is to execute on a personal computer.

MODEL EXECUTION:

Clearing statistics is supported by SIMFACTORY, but there are no provisions within the language to support multiple runs. Also, random number seeds must be changed manually between runs.

4.5. SLAM II Evaluation

GENERAL DESCRIPTION:

SLAM II is a FORTRAN-based general-purpose simulation language. It runs on a personal computers and workstations. We use Version 4.0 for this evaluation. SLAM II is offered by Pritsker & Associates of West Lafayette, Indiana.

MODEL CONSTRUCTION:

DPA: The DPA lab is modeled as a **service activity** with 135 **servers**. The **queue node** that precedes the activity has a maximum capacity of 100; when the maximum is reached, **entities** balk to a **goon node** (i.e., a "go on" node) which initiates offloaded sample processing. Offloading is modeled as a **regular activity**, which can process an infinite number of entities at the same time (and is therefore not preceded by a queue node). The processing times for both in-house and offloaded samples are (different) triangular distributions.

Line-Stops: All lots are given a priority **attribute** (attribute 3) at **assign nodes** that designates their line-stop status; attribute 3 = 2 for LS lots; attribute 3 = 1 for NLS lots. This assignment is accomplished by probabilistically branching entities through different assign nodes. It is made as soon as a lot leaves the receiving area and maintained throughout the model. In this way, queues that hold waiting lots may be "high-ranked" on attribute 3; this ensures that LS lots are always processed before NLS lots. In some cases, lots must be segregated and wait in separate files for processing (see **Personnel**, below).

Personnel: Dedicated personnel are modeled as **servers** for **service activities** preceded by a single queue node that is high-ranked on the priority attribute the lots carry, attribute 3.

Shared personnel are modeled as **resources** that are seized by a given process for the processing time and then released. Three teams of dock-to-stores personnel are shared by different operations and are therefore modeled as resources, which poll

await nodes in a specified order for waiting entities. LS lots wait in separate await nodes from NLS; all await nodes holding LS lots must be empty before shared personnel of a given type will process NLS lots. In Figure 5, two await nodes hold lots for Mechanical/Visual testing.

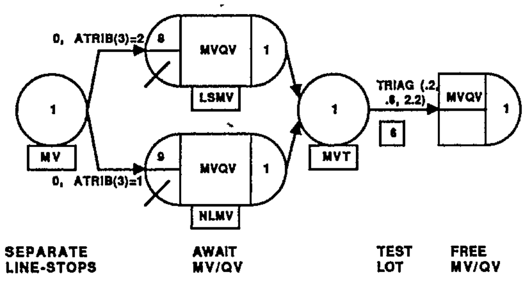

Figure 5: Portion of SLAM II Dock-to-Stores Model

Samples: HR lots are probabilistically routed to different assign nodes that release the appropriate number of entities (either 1, 2, or 3) for processing. Activities route lots and samples to their destinations (testing, DPA, and RAD). Once all tests are complete, HR lots must be mated with their sample test results. Four queue nodes hold entities for mating: one for lots, two for DPA samples, and one for RAD samples. Two are required for DPA samples so that those that should wait for RAD results to not exit prematurely. The mating of lots with their sample test results is accomplished by **match** nodes that match entities from the waiting queues based on their arrival time into their model. This ensures that sample test results are mated with the lots from which they were originally split.

Testing/Inspection: Lots pass and fail inspections through **regular activities** that probabilistically route them to different destinations. At Quick/Visual Inspection, only HR lots with samples are subject to probabilistic routing (all others are automatically passed based on their part type attribute, attribute 2). For HR lots with samples that fail this inspection, attribute 2 is reset so that they are passed the second time through.

MODEL INPUT:

A SLAM II is built in statement form with any standard text editor. Our model fit without modification into PC SLAM II.

MODEL EXECUTION:

The **montr** statement clears statistics; the user must specify the **clear** option and the time at which statistics are to be cleared. The **seeds** control statement will re-initialize the seed for a random number stream. Multiple runs are requested with the **gen** statement.

MODEL OUTPUT:

SLAM II provides a standard output report that gives all output required for this evaluation: file statistics (for queue and await nodes), activity statistics, and resource statistics. The standard report also includes an input echo.

Collect nodes are used to collect time-in-system statistics flight and non-flight lots as they exit. These provide flow times and throughput counts for each.

OTHER FEATURES:

SLAM II also has a standardized graphical node structure which may be used in planning the model before it is coded. An external package, TESS (also marketed by P&A), allows the user to input the nodes directly on a graphics screen; TESS will then generate the statement file automatically. A portion of the dock-to-stores model is shown graphically below. TESS also accepts output data from SLAM II simulation runs. Its databases can be used for graphical display of output and to animate simulation runs.

The lot arrival schedule used in the model is user-defined. This was modeled using an **array** statement in the model file.

4.6. WITNESS Evaluation

GENERAL DESCRIPTION:

WITNESS is a FORTRAN-based graphical interactive simulation tool. It runs on personal computers (IBM AT or equivalent) and workstations. We use Version 3.1.3 on an IBM AT but permit no user code in our model. WITNESS is offered by Istel, Inc., headquartered in Boston.

MODEL CONSTRUCTION:

DPA: The DPA lab is modeled as 135 **single machines** that sample processing times from a triangular distribution. Because a user cannot designate more than 99 machines of any kind at once, these are organized into two "banks" of machines (one bank of 80, the other of 55). A **buffer** precedes these machines to hold those samples for which a machine is not available. The buffer is fed by another single machine that evaluates the quantity of samples in this buffer using the WITNESS function **nparts.** If the buffer holds 100 samples, incoming samples are offloaded.

Offloading is modeled as a **batch machine.** We batch offload processing because of model size limitations; although this does not allow lots to travel independently, the following technique minimizes the consequences.

Samples cycle through the machine until they have spent an acceptable amount of time being processed (see Figure 6). The cycle time for the machine is set to 5% of the mean of the actual processing time distribution, a (different) triangular distribution for offloading; the machine processes all waiting lots as a single batch. As lots arrive to the buffer, they are given an **attribute** named TAUX, the desired processing time less 50% of the machine cycle time. (This is because we can reasonably expect samples to wait for 1/2 cycle in this buffer while the previous batch is being processed and don't want to penalize them for it). As this batch machine completes each cycle, its **action** decreases TAUX by the machine cycle time. Now TAUX, for each lot, represents how much processing time remains for that lot. If TAUX is greater than 0, the lot must go back to the buffer for another cycle through the machine.

Line-Stops: As lots leave the receiving area, a **random output rule** probabilistically routes them to machines whose action is to assign a priority attribute named PRIO to the incoming lots. PRIO keeps track of which lots are LS (PRIO = 2) and which are NLS (PRIO = 1). For operations with dedicated personnel, all lots wait in one buffer that is high-ranked on PRIO. For operations that share personnel, lots wait in separate buffers; personnel of a given type empty all buffers containing LS lots before other lots are considered.

Personnel: Because lots travel individually through the system, single machines are preferred to process them. The six operations with dedicated personnel are all modeled as banks of single machines.

For operations that share personnel, single machines are also used, but **labor** is defined to attend to them as they cycle. Because **labor priorities** are set by the machines they work on (rather than within the labor definition itself), a different machine must be provided for every different operation/priority combination for each individual worker. For example, the SMC team works on three different operations, each of which may process LS and NLS lots. Therefore, 6 single machines are necessary for each SMC worker ((3 operations)(2 priorities) = 6).

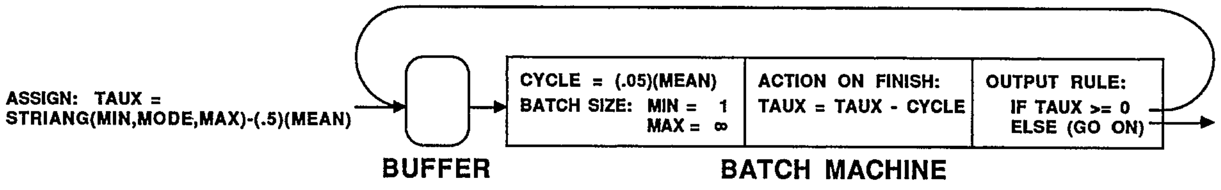

Figure 6: Flow Diagram of WITNESS Lot Batching

Samples: HR lots are probabilistically routed to **production machines** whose job it is to split off samples. A **sequence output rule** on each production machine routes lots, DPA samples, and RAD samples to their appropriate destinations.

The sample test results are later mated with their appropriate lots by **assembly machines.** Four assembly machines are used to perform the four kinds of mating (1-sample and 2-sample mating for LS and NLS lots). To ensure that lots are only available for the appropriate mating procedure, 10 buffers precede the assembly machines (4 for lots, 4 for DPA samples, and 2 for RAD samples). This is to segregate 1-sample, 2-sample, LS, and NLS lots for the appropriate mating procedures. The oldest waiting lots are matched with the oldest waiting samples, whether they entered the system together or not.

Testing/Inspection: The random output rule is used at machines where inspections occur. These output rules probabilistically route lots to either Log-Out or SMC. A part type attribute named TYPE makes sure that each lot only goes through SMC once.

MODEL INPUT:

WITNESS supplies a forms system of input. Once elements are defined in **define** mode, the user goes into **detail** mode to fill out the form for each **element**. If the user if familiar with WITNESS, he may create a **library file** with any standard text editor that may be read into WITNESS. Default memory allocation may be overridden by creating a customized file that maximizes memory space where it is needed.

MODEL EXECUTION:

WITNESS allows the user to run simulations interactively (and concurrently animate them) or to run in batch mode. **Command files** may also be built that can automatically make changes to the model at pauses in the simulation or to make multiple runs. Statistics may be cleared after a suitable warm-up period in **detail options** mode. Seeds for random number streams may be reset by choosing the **random** option from the main menu.

MODEL OUTPUT:

Standard output reports for each element supply work-in-process statistics, personnel utilizations, throughput, and flow times for each part type. WITNESS will also create a library file that can be edited by exterior to WITNESS.

Because all flow times are given in the **part** report by part type, another method is needed to get overall flow times. **Variables** are defined that track the number of flight and non-flight lots that have exited the system, as well as the average time in system for all flight and non-flight lots.

OTHER FEATURES:

Animation is optional in WITNESS; to supply animation parameters, the user enters **display** mode and specifies parameters for displaying each element. WITNESS provides four **windows** for animation, although only one may be viewed at any one time.

WITNESS supplies seven **elements** for constructing models: **parts, buffers, machines, tracks, vehicles, conveyors,** and **labor.** In addition, **attributes, variables, and distributions** may also be defined. The **distribution** element is used for the user-defined arrivals employed in the model.

WITNESS also provides an interface to SEE WHY, the general-purpose simulator produced by Istel, Inc.

4.7. XCELL+ Evaluation

GENERAL DESCRIPTION:

XCELL+ is a factory modeling system, now marketed and supported by Pritsker & Associates of West Lafayette, Indiana. It is designed to be very easy to use and runs on personal computers with 256K of EGA graphics memory. We use Version 3.0 in this paper.

MODEL CONSTRUCTION:

DPA: Modeling the DPA lab correctly requires the capability to independently process up to 135 samples at once. However, XCELL+ **processes** must have a specific batch size, which will not allow the lots to travel independently. Now, the **workcenter** that hosts the DPA process is preceded by a buffer. A **minimum holding time** is assigned to this buffer, so that no lot will leave it until at least that much time has elapsed. This minimum holding time corresponds to the actual DPA processing time. In this way, the processing time actually elapses while the lot is in the buffer; the process at the workcenter serves only to route the lot to the next operation. Because processing independence is an integral part of the dock-to-stores system, this scheme is used throughout the model to allow lots to move independently.

Offloading is handled directly from the receiving area. Processes that route samples to the HAC DPA lab may be superceded by other processes that offload by use of a **trigger** mechanism. These processes **trigger "high"** on the number of samples in the NLS DPA buffer. When this number reaches 208 (i.e., 135 samples are being "processed" and 73 NLS samples are "waiting"), the offload processes are triggered, and samples are sent to the offload buffer rather than the in-house buffer. The offload buffer has a (different) minimum holding time that represents offloaded sample processing.

Line-Stops: Lots travel through an XCELL+ model as **parts** that have **names.** In order to differentiate lot priorities, LS lots must have different part names as they travel through the model. However, this requires that two different processes be available to perform each function; one process cannot operate on two different part names. LS processes are assigned a higher **priority** than NLS processes to make sure that they are executed first.

Personnel: XCELL+ provides **maintenance facilities** which may define teams of personnel to workcenters to repair them. However, making this assignment shuts down the workcenter, leaving it unavailable for processing. For this reason, no personnel, dedicated or shared, are modeled.

Samples: Samples are split from HR lots as soon as they are received; the priority status of the lots is assigned at this same time. This is all accomplished at one workcenter that accepts HR lots from the receiving area, sets their priority status, splits off samples, and routes lots and samples to their appropriate destinations. 23 processes are required at this workcenter to accomplish these tasks (see Figure 7). LS is the first process executed; it sets the priority status of the lot. From there, the lot "cascades" through the other processes until the lots are routed to testing and the sample(s) are routed to DPA and RAD.

OTHER FEATURES:

There are five nodes available: **receiving, shipping, buffers, workcenters,** and **maintenance facilities.** Because the capabilities of each of these is rather limited, and because each process can have at most two inputs and two outputs, often **"dummy nodes"** are required to model. Once a node is located on the screen, a forms method is used to detail it.

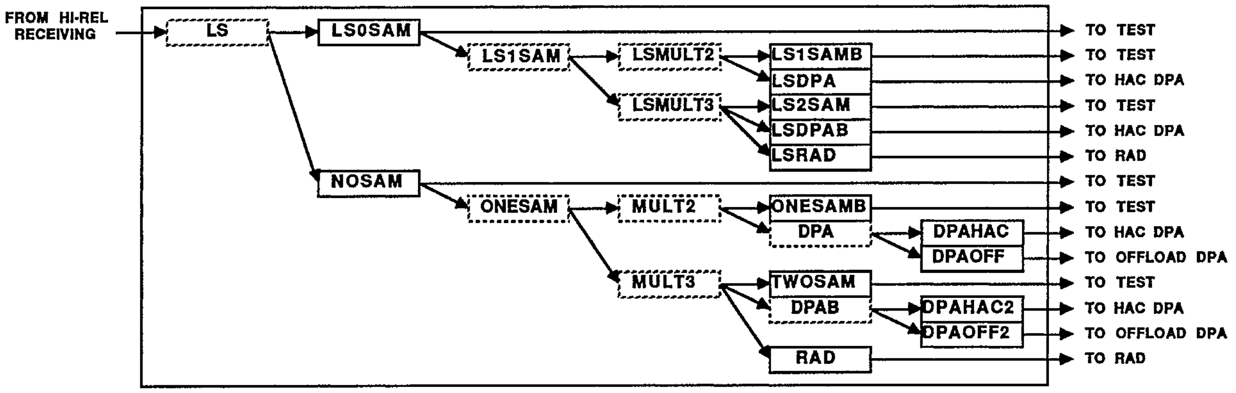

Figure 7: XCELL+ Workcenter Schematic: Samples/Priorities/Routes

Because an XCELL+ process can have at most two inputs (an **X** and a **Y** input), mating a lot with two sets of test results involves two processes, not one. Twelve buffers are required to support this. In order to maintain flow time statistics, all XCELL+ buffers must be **ordered** (FIFO, LIFO, etc). To avoid locking the mating processes, lots must wait among buffers in such a way that this does not happen. The model matches lots with their original samples, but only because the strict ordering of buffers ensures that lots and samples will always be "processed" in FIFO order.

Testing/Inspection: From any XCELL+ process, Lots that pass inspections travel along **normal** output paths; failures follow **reject** output paths. This technique is used at all points in the model where probabilistic branching is required, including inspection points.

MODEL INPUT:

XCELL+ is driven entirely by PF keys. The user inputs a model as nodes that reside directly on a graphical "factory floor." Default memory allocation may be overridden by creating a file that maximizes space where it is needed.

MODEL EXECUTION:

The XCELL+ pause processor allows the user to clear statistics, reset random number seeds, and execute multiple runs. However, this must all be accomplished via PF keys with the user in attendance.

MODEL OUTPUT:

XCELL+ provides standard output reports for throughput, work-in-process, and workcenter utilizations. To get flow time statistics, lots must be **tagged** as they enter the model. Because the need for lot independence forced us to have processing time elapse in buffers, our work-in-process and utilization output statistics have little meaning.

Screen dumps can be obtained of any XCELL+ screen. A "factory description" may also be printed.

Because the model is actually input to XCELL+ graphically, animation is automatic. The animation may be turned off for faster execution. If many dummy nodes are employed (with zero-time processes) the animation may be misleading because movement relating to zero-time processes is not displayed on the screen.

Costs may also be assigned to XCELL+ nodes. These may be tabulated for output as the model executes.

XCELL+ also offers an **analysis** option, which checks the structural integrity of a model and offers some guesses concerning bottlenecks and expected model throughput.

XCELL+ allows for user-defined arrivals only in the sense that it will accept them in order from a sequential file. This had to be generated external to XCELL+ for all arrivals that were desired into the model.

5. SUMMARY

5.1. Evaluation

MAP/1: does not pass our template evaluation for the following reasons:

1. Mating of sample test results with their original lots is not guaranteed
2. Flow time statistics for flight and non-flight lots are not available

SIMAN passes our template evaluation.

SIMFACTORY: does not pass our template evaluation for the following reasons:

1. Mating of sample test results with their original lots is not guaranteed
2. Unable to specify different times for in-house and off-site DPA processing

SLAM II: passes our template evaluation.

WITNESS: does not pass our template evaluation for the following reasons:

1. Mating of sample test results with their original lots is not guaranteed
2. Model size limitations forced batching of lots where independent processing is preferred

XCELL+: does not pass our template evaluation for the following reasons:

1. The decision to offload DPA samples can only be approximated
2. Personnel are not modeled; utilizations are not available
3. WIP output is not meaningful because processing time elapses while lots are in buffers

As we state in our ground rules, we use this restrictive evaluation method to emphasize that this example merely demonstrates our approach and does not determine the "best" simulation language.

5.2. Approach Review

We realize that the number of people with access to as many simulation languages is small; however, possible alternatives include: timesharing, renting, or buying on a trial basis. Without the software, much of the benefit from using our approach can still be gained by using only the documentation, which can be purchased from most vendors for a nominal fee. Even if it is impossible for the buyer to experiment on his own, many vendors are willing to build a very simple demonstration model if supplied with the requisite information. Obviously, the model would have to be simple and as such would provide limited information; however, this approach would give the buyer a concrete evaluation strategy to follow. In addition to forcing the buyer to go through the process of identifying a representative system including problem identification and simplified data collection steps, it also provides an opportunity for him to gauge the vendor's willingness to provide support.

Although we feel language comparisons using a representative project are important, we emphasize that there are other considerations to be made when selecting a language. Product maturity, vendor stability, vendor support in terms of training and troubleshooting, hardware restrictions, and cost constraints are a few of the other facets of the selection process to consider.

5.3. Perspective

Currently, simulation vendors are touting languages which "free" the analyst from the drudgery of programming. However, many newcomers consider "programming" to be "modeling." Unfortunately, there is a very real difference between modeling and programming just as there is a real difference between a simulation model and a simulation project. While it is true that many simulation languages are providing powerful non-programming environments, few if any are providing the modeling expertise that is required for the other parts of a project. These newer products relieve the modeler from much of the tediousness of building and debugging a model, but do not substitute for a modelers' experience and knowledge when it comes to scoping and bounding a problem, making assumptions, identifying measures of effectiveness, designing experiments, analyzing results, etc. Based on our experience, it is not the selection of the simulation language that ensures a successful project but rather the person using the language and the support received from management.

REFERENCES:

Haider, S. Wail, and Banks, Jerry, 1986. Simulation Software Products for Analyzing Manufacturing Systems. *Industrial Engineering*, Vol. 18, No. 7, pp. 98-103.

AUTHORS' BIOGRAPHIES:

F. BRADLEY ARMSTRONG is a Staff Engineer at Hughes Aircraft Company in Long Beach, California. He received a B.S. in Mechanical Engineering from the University of Texas at Austin and an M.S. in Industrial Engineering from Purdue University. He is currently an internal consultant providing manufacturing simulation consulting, training, and software support at Hughes. Prior to joining Hughes, he was a Senior Systems Analyst for Pritsker and Associates and an Operations Analyst for General Dynamics in Fort Worth, Texas. He is a professionally registered engineer in Indiana, a senior member of both SME and IIE, and a member of SCS.

F. Bradley Armstrong
Staff Engineer
Hughes Aircraft Company C&DP
P.O. Box 9399, LB/C05/2106
Long Beach, California 90810
(213) 513-5847

SCOTT SUMNER is a Member of the Technical Staff at Hughes Aircraft Company in El Segundo, California. He received a B.S. in Mechanical Engineering from Cornell University and an M.S. in Systems Management from USC. He is currently involved in a number of simulation projects at Hughes' Space and Communications Group. He is also a member of ASME and SCS.

Scott Sumner
Member of the Technical Staff
Hughes Aircraft Company S&CG
P.O. Box 92919, SC/S65/J323
Los Angeles, California 90009
(213) 606-9787