Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

REALTIME OPERATIONS SCHEDULING FOR FLEXIBLE MANUFACTURING SYSTEMS

J. S. Dhingra
K. L. Musser
G. L. Blankenship

Systems Research Center
and
Department of Electrical Engineering
University of Maryland
College Park, MD 20742, USA.

ABSTRACT

In this paper we present a closed loop operations scheduling and control design for a flexible manufacturing system. The closed loop system consists of a simulated annealing based predictive scheduler and the manufacturing system model in the feed forward path, and a reactive re-scheduler in the feedback path. Deviations of the observed sequenced operation timings from the scheduled timings, quantified by a cost functional, are used to trigger the re-scheduling loop. The capabilities of the system are demonstrated using a discrete event simulation of a PC board assembly operation.

1 INTRODUCTION

Operations scheduling in a flexible manufacturing system has been studied by many researchers in the past few years. Finding computationally feasible solutions for this NP-complete problem has been of continuous interest in the industrial engineering, operations research and manufacturing technology communities. Various approaches have been suggested including, expert systems, neural network methods, heuristics based systems and other combinatorial optimization based methods. In general, these methods generate schedule plans assuming ideal operating conditions, i.e., no machine failures, constant setup and processing times, etc.; that is, they do *predictive scheduling*. Additional *reactive scheduling* functions are necessary to maintain the actual operations schedule near the planned schedule.

In this paper, we present a reactive scheduling methodology, which uses a predictive scheduler, in conjunction with a manufacturing system dynamical model, and a threshold triggered, feedback rescheduler for on-line scheduling of the manufacturing system. The closed loop configuration is shown in Fig. 1. In the next section we present the simulation model of the manufacturing system used in our study. We then describe the predictive scheduler and the reactive scheduler.

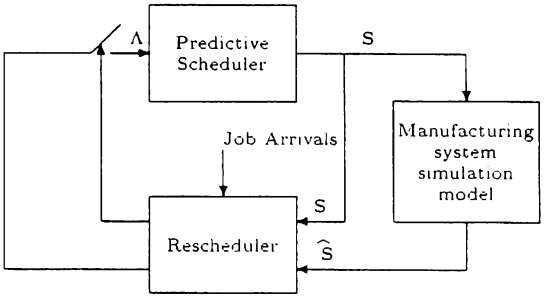

Figure 1: Block Diagram for Reactive Operations Scheduling

2 MANUFACTURING SYSTEM MODEL

A flexible manufacturing system is composed of a collection of machines, each capable of performing a variety of operations on different sets of parts. The parts are processed in a predefined sequence of operations at different machines. The processing at each machine can be either batch or continuous mode. Batch mode operation involves processing a single part type over a given time interval (determined by the batch size), and then switching to process another part type. Batch mode processing is used if there is a significant setup time involved for the machine to switch from processing one part type

to another. Continuous mode processing is used in cases where the processing for different part types is very similar and changeovers do not require a significant reconfiguration of the machine setup. In our model, we have grouped sets of physical machines to form "virtual machines" which process parts in tandem and the manufacturing system operates in a hybrid mode, i.e., use batch mode processing for inter-virtual-machine operation and continuous mode processing (single product) within each virtual-machine. In this paper, we will only discuss scheduling issues for batch mode processing.

A batch is a physical order for some number of units of a specific part type. It can be represented as a sequence of operations (batch-mode) done in a predefined order on different pre-assigned machines. Each constituent batch operation represents the processing of the set of parts on a particular machine. The sequence of these batch operations defines a *batch-order precedence* for each batch. In a similar fashion, given a schedule, one can define a *machine-order precedence* for each machine to be the ordering of different batch operations on that machine. The precedence constraints in the whole schedule, both machine-order and batch-order, can be represented as a single sequence (not necessarily unique) of operations. The two precedence sequences can be readily derived from this single sequence.

To illustrate let us take a simple two machine (m_1 and m_2), two batch (b_1 and b_2) problem. Each batch consists of two operations. In batch b_1 the operations are sequenced first on m_1 and then on m_2; the reverse order is used by batch b_2. If operation ${}_bO_i^m$ represents the i^{th} operation of batch b to be performed on machine m, then, for the above example, a possible schedule operation sequence is given by $< {}_1O_1^1, {}_2O_1^2, {}_1O_2^2, {}_2O_2^1 >$. Since operations ${}_1O_1^1$ and ${}_2O_2^1$ are for the same machine, this schedule operation sequence indicates that ${}_1O_1^1$ must complete before ${}_2O_2^1$ begins. This schedule operation sequence and the corresponding derived machine schedules are shown in Fig. 2. The arrows denote the operation sequence for each batch. The same schedule often can be derived from other schedule operation sequences, such as $< {}_1O_1^1, {}_2O_1^2, {}_2O_2^1, {}_1O_2^2 >$ for our example.

Each operation is characterized by a set-up time and the total expected processing time. Based on the characteristic operation times, machine-order precedence and the batch-order precedence, we can assign start and finish times to the various operations. This assignment of operation sequences on different machines along with their start and finish times, defines the manufacturing system schedule. The initial schedule is generated on the basis of addition of new

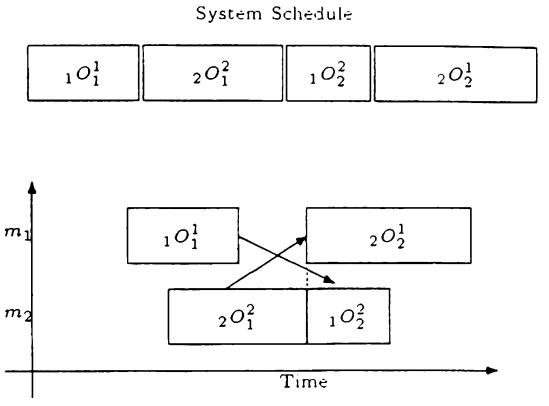

Figure 2: Derived machine schedules

batches to the schedule. The constituent batch operations for the new batch are inserted in the machine-idle times in the existing schedule, for an earliest possible finish time.

We have used the software package QNAP2 (Queuing Network Analysis Package) to develop a manufacturing system simulation model. The optimal schedules generated by the predictive scheduler are used as an input to the simulator. The model consists of a source or a warehouse station which generates the batch orders and routes them to the appropriate work stations for processing. The machine (work) stations are modeled as single server queues where the batch operations are processed on first-come first-serve basis. We allow the machines to be either single physical machines or virtual machines (a sequence of physical machines processing parts in tandem).

For each batch operation, the machine station simulates the setup time as a constant time delay. The actual operation processing is simulated as a sequence of random time delays corresponding to each individual part processing. In our model we take the part processing time to be a Gaussian random variable with known mean and variance. We permit three different modes of failure for each machine. Failure Mode 1 (FM1) allows for random repair time and is more frequent than FM2 or FM3. FM2 and FM3 represent different types of drastic failures which have large deterministic repair times. The mean time between failures (MTBF), for each failure mode, is assumed to be exponentially distributed with a time varying mean. The failure rate for each mode is incremental with respect to number of operations processed since the previous failure. This is consistent with the machine failure probabilities in an actual manufacturing system.

The simulator has a central controller which keeps track of the batch operation precedence and also the machine ordering. On completion of a batch operation on a machine, the batch is routed to the machine corresponding to the succeeding operation. The sequencing in the predictive schedule $\mathcal{S}$ is preserved in the simulation. However, the actual operation times, which differ from the scheduled time because of random processing and setup times, define the actual schedule $\hat{\mathcal{S}}$. Fig. 3 shows a sample output of the simulator, including predicted and actual (simulated) processing times. (The schedule shown is a partial schedule of a 30 batch schedule. It may appear inconsistent in terms of the operation timings due to the omitted operations.)

3 JOB SCHEDULER

The predictive job scheduler begins with a feasible schedule and iteratively attempts to improve it by making small changes. We use *simulated annealing* as the algorithm which determines whether or not to accept the small changes. The cost function being optimized is a composite penalty for tardiness, work-in-process, and finished goods storage. The user may tailor the weightings of the cost elements to suit a given operational scenario. See Musser, *et al*, 1991 for details.

The simulated annealing algorithm treats the process of optimization as a Markov random process, where each schedule is a state in a Markov chain. An initial schedule, S_0, is chosen at random. Next, a nearby "trial" schedule, S_1^{trial}, is chosen by some probabilistic rule, and the two are compared. If the "trial" schedule has lower cost, it is accepted, with probability 1, as the new state of the Markov chain. Otherwise it is accepted with a lower probability determined by the relative costs. More precisely,

$$\Pr\{S_{k+1} = S_{k+1}^{trial} | S_{k+1}^{trial}, S_k\} = \begin{cases} 1 & \text{if } c(S_{k+1}^{trial}) < c(S_k) \\ e^{-\frac{c(S_{k+1}^{trial}) - c(S_k)}{T_k}} & \text{otherwise} \end{cases}$$

$$\Pr\{S_{k+1} = S_k | S_{k+1}^{trial}, S_k\} = 1 - \Pr\{S_{k+1} = S_{k+1}^{trial} | S_{k+1}^{trial}, S_k\}$$

Here the value, T_k is called the "temperature," and the sequence $\{T_k\}_{k=0}^{\infty}$ is called the temperature schedule. If the trial schedule is only accepted when it has lower cost, the algorithm will remain in any local minimum it may find. This is the case when the tem-

```
Batch No.  5          (QTY = 5000)
mach    setup        start time       finish time     proc time(hr)
W/E        0   18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
Sched times    18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
M/C 4     30    3:48:48(10/ 1/91)    2: 1:36(10/ 2/91)    22.21
Sched times     5:32:48(10/ 1/91)    2:14:56(10/ 2/91)    20.71

Batch No.  6          (QTY = 1000)
mach    setup        start time       finish time     proc time(hr)
W/E        0   10: 0: 0( 9/29/91)   10: 0: 0( 9/29/91)     0.00
Sched times    10: 0: 0( 9/29/91)   10: 0: 0( 9/29/91)     0.00
M/C 1    150   17:49:20(10/ 1/91)   19:12:32(10/ 2/91)    25.38
Sched times    15:15:44(10/ 1/91)   15: 4:32(10/ 2/91)    23.81
M/C 5     24   23:24:16(10/ 2/91)    8:31:28(10/ 4/91)    33.12
Sched times    22:12:16(10/ 2/91)    7:32:16(10/ 4/91)    33.33

Batch No.  7          (QTY = 1500)
mach    setup        start time       finish time     proc time(hr)
W/E        0   18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
Sched times    18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
M/C 1    150   21:42:24(10/ 2/91)   11:32:48(10/ 4/91)    37.84
Sched times    15: 4:32(10/ 2/91)    2:46:56(10/ 4/91)    35.71
M/C 4     24    5:43:28(10/ 3/91)    6:28:48(10/ 7/91)    48.76
Sched times     1:20:32(10/ 3/91)    3:20:32(10/ 7/91)    50.00

Batch No.  8          (QTY = 1005)
mach    setup        start time       finish time     proc time(hr)
W/E        0   18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
Sched times    18: 0: 0( 9/28/91)   18: 0: 0( 9/28/91)     0.00
M/C 3     90   14:58: 8(10/ 1/91)   17:44: 0(10/ 1/91)     2.76
Sched times    16:18:40(10/ 1/91)   18:54:56(10/ 1/91)     2.61
M/C 4     30    0: 5:52(10/ 3/91)    1:44:32(10/ 3/91)     1.65
Sched times    23:27:28(10/ 2/91)    0:56:32(10/ 3/91)     1.48
```

Figure 3: Sample simulation run

perature is zero. A nonzero temperature, however, gives the Markov chain the opportunity to escape from local minima. In simulated annealing, we let $T_k \rightarrow 0$ slowly, so that the state of the Markov chain converges in probability to the collection of schedules with globally minimum cost. In practice, since we can keep track of the best schedule visited so far, we are more interested in having ever reached a minimum than reaching it and staying there.

Generating the trial schedule, S_{k+1}^{trial} simply involves perturbing S_k. As discussed in earlier, each schedule can be represented by a global operation sequence. To perturb the schedule, we must permute the corresponding sequence. We do this by randomly selecting a short subsequence and a new position in the sequence which remains. Then the subsequence is reversed and moved to the new position. The resulting global sequence may not satisfy all batch precedence constraints. To resolve this potential violation, each offending operation is swapped with the (unique) operation in the same batch which does not violate the batch precedence constraint. Finally, when all batch precedence constraints are satisfied, the start and finish times of all operations are computed using the rule that each operation begins as soon as its predecessors have completed (with an appropriate adjustment for setup time). The start and finish times are used to compute costs of tardiness, work-in-process, and finish goods storage. This method is shown to converge to the optimum in Musser, *et al*, 1991. We stop the iterative procedure after a certain number of iterations or when the cost stops improving, which results in a schedule which is good, though sub-optimal.

4 RESCHEDULER

The desired optimal schedule $\mathcal{S}$, generated by the scheduler, and the actual schedule from the (simulated) model (or the actual factory) $\hat{\mathcal{S}}$, are compared by the rescheduler (Fig. 3). The deviation between the two schedules is measured on the basis of some cost criteria and a rule-based, threshold triggered scheme is used for rescheduling. At each rescheduling instant new jobs can be added to the schedule; this is our formulation of on-line reactive scheduling.

For notational convenience let us drop the batch precedence ordering and use O_i^m to represent the i^{th} operation to be performed on machine m (The automatic process flow control in the simulation model makes the batch precedence information redundant). In effect, we are re-labeling our operations on each machine and will carry out the analysis on the basis of individual machine schedules instead of the complete manufacturing system schedule. Let $t_s(O_i^m)$ and $t_f(O_i^m)$ respectively, denote the start and the finish times for the operation O_i^m in the predictive schedule $\mathcal{S}$. Now the schedule for machine m can be defined as:

$$\mathbf{S}_m \triangleq < (O_k^m, t_s(O_k^m), t_f(O_k^m)) >_{k=1}^{N_m}$$
$$t_s(O_k^m) < t_f(O_k^m) \leq t_s(O_{k+1}^m)$$

Let there be M machines in the manufacturing system (numbered $1, \ldots, M$). Then the overall schedule can be written as

$$\mathbf{S} = < \mathbf{S}_m >_{m=1}^{M}$$

In the following we use t to represent "real time," that is, the position in the actual schedule, $\hat{\mathcal{S}}$. For each machine, there is a point in the predictive schedule corresponding to t in $\hat{\mathcal{S}}$. This point is the "active time" corresponding to real time t, which we write $\tau_m(t)$. Precisely, suppose operation O^m is in process at time t in the actual schedule, $\hat{\mathcal{S}}$, and suppose it is expected to complete at time $t + \delta$. Then the active time for machine m is given by

$$\tau_m(t) = t_f(O^m) - \delta.$$

Active times are used by the rescheduler as indicated below.

Initially the active times on all machines are equal to the real time, say $\tau_m(t_0) = t = t_0$. Because of the various random features of the simulation, the two schedules differ in terms of the operation timings. To correlate the two schedules, at real time t ($t_0 \leq t$), we can define the active time vector $\Lambda(t) = [\tau_1(t), \tau_2(t) \ldots \tau_M(t)]^T$ ($\Lambda(0) = [t_0, t_0 \ldots t_0]$). $\Lambda(t)$ represents the present status of $\hat{\mathbf{S}}$ in $\mathbf{S}$. For illustration, Fig. 4 shows the two time scales for a single machine schedule.

For each operation O_m^i, we associate a cost $c_m^i = c(t_s(O_m^i), t_f(O_m^i), \hat{t_s}(O_m^i), \hat{t_f}(O_m^i))$. This cost represents the penalty involved for the actual operation times to differ from the scheduled times. Based on the individual operation costs, the net schedule cost can be calculated as

$$C(\mathbf{S}, \hat{\mathbf{S}}, t_0, t) = \sum_{m=1}^{M} \sum_{\substack{i=1 \\ t_0 \leq \hat{t_f}(O_m^i) \leq t}}^{N_m} c_m^i$$

The rescheduler keeps track of the two schedules $\mathbf{S}$ and $\hat{\mathbf{S}}$ and at the end of the each operation updates the net schedule cost $C(\mathbf{S}, \hat{\mathbf{S}}, t_0, t)$. The rescheduling loop is triggered under the following conditions:

- $C(\mathbf{S}, \hat{\mathbf{S}}, t_0, t) > C_T$. Here C_T is some predefined cost threshold.

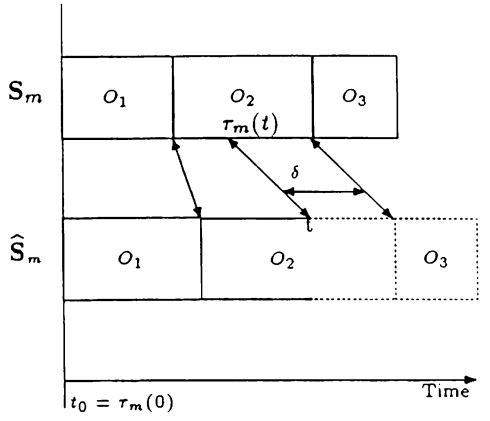

Figure 4: Active Times for a single machine

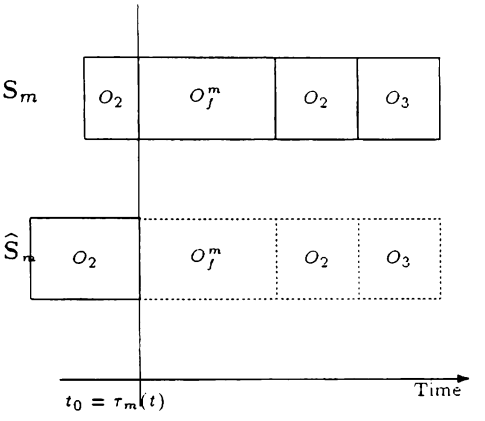

Figure 5: Schedule representation for machine failure induced rescheduling

- Failure Mode 2 on any machine
- Failure Mode 3 on any machine
- New Job arrivals

At each such rescheduling instant t, the rescheduler polls in the current status ($\Lambda(t)$) of the manufacturing system schedule. This status is then input to the scheduler. If the rescheduling is necessary due to failure 2 or 3 on any machine, an extra input in the form of the operation O_f^m along with its associated start and finish times is sent to the scheduler.

The Scheduler uses this information to update the present schedule S. Past operations are discarded, and the current real time t becomes the new initial time $t_0 \leftarrow t$. Hence, $\tau_m(t) \leftarrow t$ also. The scheduler then allows the user to make changes (add/remove batches or operations) in the schedule S[1]. Next the simulated annealing based optimization is activated to generate an optimal schedule. Since the manufacturing model is continuously simulating the schedule S, the optimization is not done for the immediately scheduled operations. Basically we do not reschedule the operations scheduled to start between t_0 and $t_0 + T$, for some disable period $T > 0$. This allows for on-line optimization of the current schedule while the simulation is still running.

Once an optimal schedule is obtained it is input to the simulation model. In case of failure mode rescheduling, the failure operation O_f^m is treated as an operation with fixed processing time. Since in this case $t_s(O_f^m) = t_0$ the operation is not rescheduled and it is simulated as a repair time. Taking the schedule shown in Fig. 4, if a failure FM2 or FM3 occurs at time t on machine m, then the two schedules S and $\hat{S}$ after rescheduling are depicted in Fig. 5. This closed loop process is carried out continuously and leads to reactive operations scheduling for manufacturing system. The complete process can be represented in a flow chart as shown in Fig. 6.

5 CONCLUSIONS

We have used the above methodology to design and implement a reactive scheduler. We used a simulation model of the Texas Instruments PC board manufacturing facility in Johnson City, TN., as our test case. The predictive scheduler has been used in the TI plant; but the on-line reactive scheduler has not been tested.

Fig. 7 depicts a scaled rescheduling trace.[2] The user defined operational cost functions, cost threshold C_T and the rescheduling disable period T allow the methodology to represent a range of operational conditions. The cost configuration can also be changed (online) to address changing operational objectives.

We are now working on multi-level reconfiguration control algorithms for manufacturing systems. Perturbation analysis, in conjunction with stochastic gradient algorithms, is being used for online control of processing rates for the case of continuous-mode processing. Impulse control algorithms based on dynamic programming (quasi-variational inequalities) are being developed for feedback control in the batch mode case. Results will be presented in a forthcoming paper.

[1]To maintain compatibility with the notation, the operations are re-numbered from 1 ... N_m for each machine m

[2]The failure probabilities and time scales have been modified to present the two modes of rescheduling

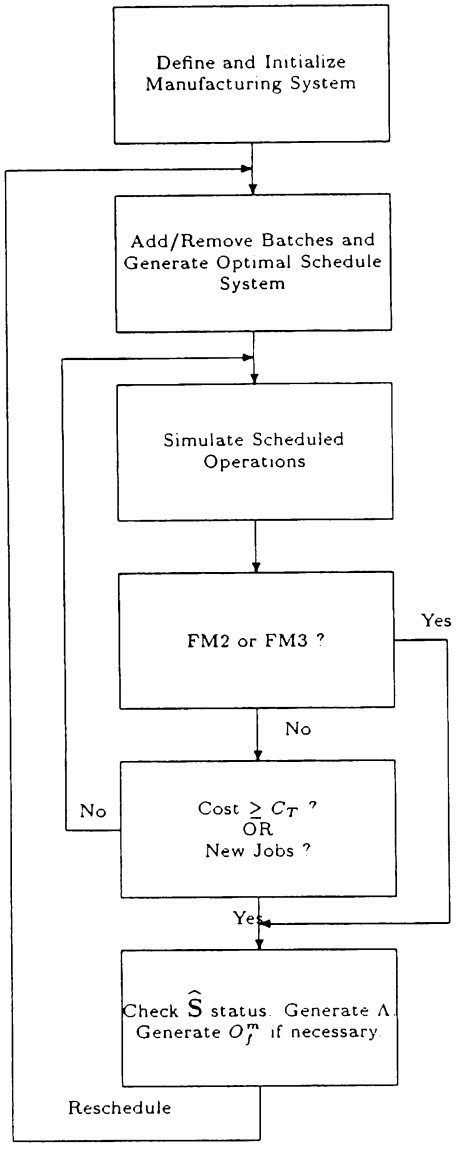

Figure 6: Reactive Operations Scheduling Algorithm

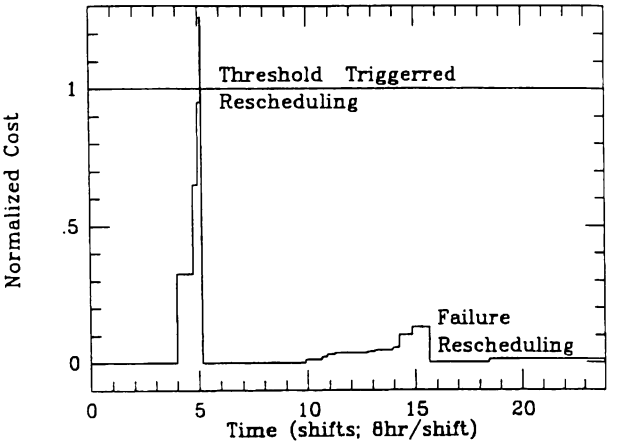

Figure 7: Rescheduling Trace

ACKNOWLEDGEMENTS

This work was supported by NSF Engineering Research Centers Program NSFD CDR 88003012, and by grants from Texas Instruments, Inc.

REFERENCES

Dhingra, J. S., K. L. Musser, G. L. Blankenship and L. Ferguson. 1991. Annealing Based Experiment in Scheduling. To appear in *Texas Instruments Technical Journal*.

Musser, K. L., J. S. Dhingra and G. L. Blankenship. 1991. Optimization based Job Shop Scheduling. To appear in *IEEE Transactions on Automatic Control*.

van Laarhoven, P. J. M., E. H. L. Aarts, and J. K. Lenstra. 1992. Job Shop Scheduling by Simulated Annealing. *Operations Research* 40:113–125.

AUTHOR BIOGRAPHIES

JASTEJ S. DHINGRA received his B. Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur in 1986. In 1988, he received the M.S degree from Virginia Tech., Blacksburg. Since Fall 1988, he has been a Ph.D candidate at the University of Maryland, College Park. He was a consultant with Texas Instruments Inc. and Techno-Sciences Inc. during summers of 1990

and 1991 respectively. His research interests include stochastic signal processing, modeling, analysis and control of discrete event dynamic systems. He is a member of Phi Kappa Phi.

KEITH L. MUSSER was born in Sask., Canada in 1964. He received a B.S.E. from LeTourneau College and an M.S.E.E. from Johns Hopkins University in 1986 and 1989, respectively. Currently he is a Ph.D. candidate at the University of Maryland. He worked in the Spacecraft Guidance and Control Group at the Johns Hopkins Applied Physics Laboratory from 1986 to 1989, and has been employed by Techno-Sciences, Inc. since 1991. His current work concerns optimization methods in manufacturing decision support systems, and in the application of numerical methods in acoustic and vibration control problems.

GILMER L. BLANKENSHIP was born in Beckeley, West Virginia on September 11, 1945. He received the S.B., S.M., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, Mass., in 1967, 1969, and 1971, respectively. He is a Professor in the Department of Electrical Engineering, University of Maryland, College Park. He served as Associate Chairman of the Department for more than five years. He is faculty associate with the Systems Research Center and is a member of the Applied Mathematics Faculty. Since 1971 he has held visiting positions with New York University, New York, the University of Illinois, Urbana-Champaign, The University of Maryland, the U.S. Department of Energy, and with Erasmus University, Rotterdam. From 1971 to 1979 he was with the Department of Systems Engineering, Case Western Reserve University, Cleveland, Ohio. His research interests include discrete event systems scheduling theory and applications, nonlinear and adaptive control theory, scattering theory and the mechanics of advanced materials, and the applications of AI methods and computer algebra in these areas. Dr. Blankenship is also Vice-President of Techno–Sciences, Inc., a Maryland high technology firm specializing in advanced control and signal processing. Dr. Blankenship has been an Associate Editor of the *IEEE Transactions on Automatic Control* and Chairman of the Technical Committee on Stability, Nonlinear, and Distributed Systems of the Control Systems Society. He is an Advisory Editor of the journal *Acta Applicandae Mathematicae*. He has also served as Chairman of the AACC Theory Committee, as a delegate to the IFAC Theory Committee, as a member of the IEEE Edison Medal Committee, and as Chairman of the IFAC Working Group on Singular Perturbations and Asymptotic Analysis. Dr. Blankenship is a member of the Society for Industrial and Applied Mathematics, and the Association of Computing Machinery. He is a Fellow of the IEEE.