The Impact of Network Topology On Rational-Function Models of the Cycle Time-Throughput Curve

Carl Allen
November 15, 2003
Industrial Engineering Honors Program
McCormick School of Engineering, Northwestern University

Advisors
Professor Barry Nelson and Professor Bruce Ankenman
Northwestern University
Department of Industrial Engineering and Management Sciences

Abstract

The use of cycle time – throughput curves in the semiconductor manufacturing industry is imperative to strategic planning. Exhaustive simulation runs of complex queueing networks are analyzed here with the intention of finding network characteristics that vary the conventional functional form of the mean and higher moment cycle time – throughput curves. The functional form for the mean and higher moment CT/TH curves contains an exponential term in the denominator that accounts for the explosion of the cycle time mean and variance at high traffic intensities and is generally fixed. We explore cases where the value of the exponent must be adjusted significantly in order to achieve the best fit of empirical data. The networks we simulate represent a subset of characteristics deemed common in wafer fabrication plants. Presented here are analyses of data generated by models characterized by re-entrant flow, queues in series, batching, prioritization queueing schema, and stochastic machine failure. Results from models exhibiting the "Shortest Processing Time First" queueing discipline and machine failure reveal varying values of the exponent p in the mean CT/TH curve and these as well as other models fit higher moment models with varying exponents.

Introduction

In this paper we examine several features of manufacturing networks and how those network topologies influence the generation of rational-function models of the cycle time – throughput curves. More specifically, we gear our research toward the semiconductor or wafer fabrication industry and topologies that are typical in that sector. This is one portion of a larger study being conducted by Professors Barry Nelson and Bruce Ankenman of Northwestern University in conjunction with the work of Gerald T. Mackulak and John W. Fowler of Arizona State University and funded by a grant from the National Science Foundation. Also contributing to the findings in this paper is Northwestern PhD candidate Feng Yeng.

The project arose out of the great need for more effective methods of strategic planning for manufacturing. Cycle time-throughput curves are extremely useful for managing inventory, planning delivery, and making cost estimates. If the curves are accurate, a firm can confidently determine the start rate needed to achieve a total system processing time or alternatively the amount of time to allocate to a task that requires a set throughput. If an estimated cycle time – throughput curve is not indicative of the true system behavior, costly mistakes can be made in planning. Making the curve accurate over a large range of possible throughputs or start rates is particularly important to firms that experience a great deal of variability in order sizes and frequency.

Sophisticated models are needed to approximate the cycle time – throughput curves of manufacturing networks due to the exponential behavior of the process cycle time as the system nears capacity. Nelson, Ankenman, and Yeng developed robust models to account for this explosive behavior that vary fundamentally from previous research in the area. Specifically, Cheng and Kleijnen examined functional forms of models for the M/M/1 queue using FIFO, SPTF, and LPTF, as well as a network of queues in a packet assembly/disassembly device. Their generalized model called for a polynomial factor to account for error in the theoretical formula while fixing the other aspects of the functional form. The models used in this research, on the other hand, allow the fundamental functional form to change while always using the lowest degree polynomial factor possible, sometimes only a constant. The functional form for the mean cycle time – throughput curve that we use is shown below where *f(TH)* is the polynomial factor as a function of the throughput and *p* is allowed to vary.

$$Mean\ Cycle\ Time = f(TH)/(1-TH)^p$$

It is the intention of this portion of the research to find network topologies and characteristics typical in semiconductor manufacturing processes that necessitate varying values of *p*. In previous research, *p* was always fixed at one for the mean cycle time model. The higher moments of the cycle time – throughput curve traditionally adhere to similar functional forms, with the exponent in the denominator also fixed. We hypothesize that when allowed to vary, certain topologies will necessitate vastly different exponent values to achieve the best fit of the data.

In designing our simulation experiments, we use principles outlined by Ward Whitt in his 1989 paper "Planning Queueing Simulations." Due to increasing variance at higher throughputs, it is necessary to generate more data at higher throughput points in order to maintain a uniform relative error. A uniform relative error based on a 95% confidence interval at the highest throughput is used as a basis for calculating the

necessary number of replications run at each design point. Initial condition bias is also an influential factor in large, complicated networks of queues and portions of the simulation runs are truncated to account for this "warm-up" period and ensure steady-state measurements. Consistent with Cheng and Kleijnen's work, we also choose design points that are concentrated at high throughputs and more spread out at low traffic intensities.

We develop simulation models exhibiting various topological characteristics typical in semiconductor manufacturing. Firstly, we examine re-entrant flow of entities into various parts of the system. Coupled with this analysis, we manipulate the number of queues in series while varying the number of entity re-entries and the length of the network segment that is repeated. A second set of topologies involve queues in series where entities are batched, processed as batches, and unbatched multiple times throughout a network. Embedded with this is an analysis of the location of a bottleneck queue or multiple bottlenecks located within the system. This is to confirm Ward Whitt's well known stipulation that a queueing network can be generalized to the behavior of any bottleneck queues, and to see if any exceptions to the rule can be found.

Similar to Cheng and Kleijnen's study, we test the robustness of our functional models by simulating systems utilizing queueing disciplines other than FIFO. In the semiconductor industry, multiple part classification and complex prioritization schemes can be represented by a myriad of queueing methods. Our work focuses on LIFO and SPTF, as well as a random queueing procedure, which we call SIRO or "Service in Random Order." Random machine downtime and failure can also be a major disruptive factor in manufacturing networks and has been seldom modeled in past research. We present data generated by simulation models exhibiting the aforementioned conditions and examine the resulting fits to a set of our models.

Method

Initial Experimental Design/Model Development

The network topologies we analyzed were modeled in Arena, an advanced simulation software package. Entities in our Arena models represent individual silicon wafers traversing a manufacturing system. We modeled the entity arrivals into the system as a Poisson process, which means arrivals occur one at a time at independent and stationary increments. The exponentially distributed mean time between arrivals in our Poisson arrival process is a parameter that can be set for each simulation run to obtain the desired throughput. In reality, arrival processes in wafer fabrication processes are likely to be much less variable, so using a Poisson process in our models accounts for any conceivable level of variability in the field. Process modules in our models simulate different delays and processing stations encountered by the wafers. The time spent in these modules can be represented by statistical distributions with user-defined means and standard deviations. Like the time between arrivals, we modeled the mean service time in each of our process modules to be exponentially distributed to account for sources of variability in wafer processing times. The queues in our process modules used the FIFO or "First-In-First-Out" queueing discipline with infinite capacity queues for most of our simulations. Arena is also able to model the "Last-In-First-Out" queueing discipline as well as queue sorting based on entity attributes. In some of our later models, we experimented with LIFO, SPTF (Shortest Processing Time First), and SIRO (Service in Random Order) to account for any amount of complexity in queue sorting. The paths of the wafer entities through the manufacturing network are determined by our arrangement of the process modules and parameters of the system. After a simulation run, Arena can calculate statistics such as the average time an entity spends in the system, average queue length, and server utilization that can be written to file for further analysis.

Initially, we examined several network topologies that our research team determined were common in the manufacture of silicon wafers. Preliminary data was gathered with models using processes in series, reentrant-flow, batching, bottleneck queues in different locations, and combinations of these phenomena. In addition, we later examined models incorporating machine failure. Our first major focus, however, was on three of these topology characteristics: re-entrant flow (entities going back and repeating certain processes), different numbers of queues within the re-entry loops (entities repeating a series of several processes or only a single process), and different numbers of queues in series. Respectively, the parameters K, Seg, and Q made up our experimental design in 12 scenarios.

K (# Loops)	Seg (Segment Length)	Q (# of Queues)
3	1	3
5	1	3
3	3	3
5	3	3
3	1	10
5	1	10
3	3	10
5	3	10
3	1	20
5	1	20
3	3	20
5	3	20

Figure 1: Experimental Design

The processes were arranged linearly with the repeated segments always at the end of the chain. For example, with Q = 20, Seg = 1, and a K = 5, the last process is repeated by each entity five times. With Q = 3, Seg = 3, and K = 3, entities flow through the entirety of the system (only three processes long) a total of three times. All in all, these scenarios can subject the entities to going through as few as 5 total processes (Q = 3, Seg = 1, K = 3) and as many as 32 (Q = 20, Seg = 3, K = 5).

The Arena model was constructed such that the parameters could be easily manipulated and data gathered efficiently. The basic structure of the first model we used (see Figure 2 below) was two process modules, one that contained a sub-module where the entities actually traversed the manufacturing network, and one with a sub-module where the entity data was written to file. The rest of the top view of the model is simply the create and dispose modules where the entities enter and leave the system, and some assignment modules that attach a creation time attribute to the entities and count the entities as they leave the system.

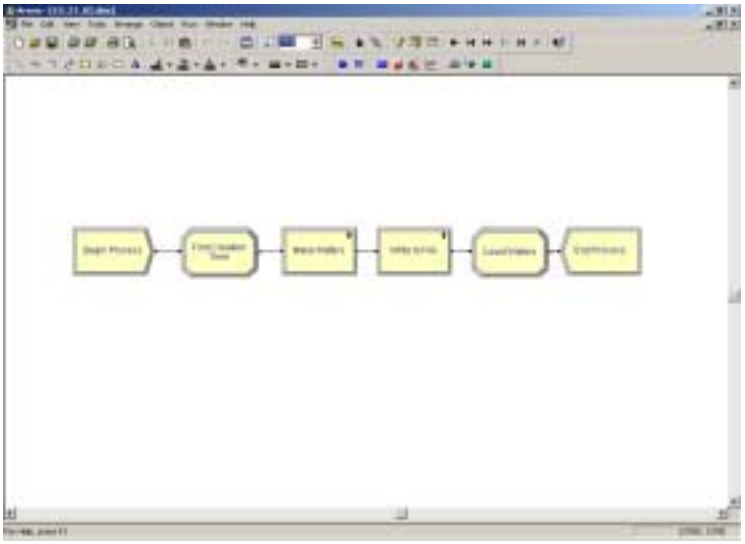

Figure 2: Basic Arena Model View 1

The sub-model showing the actual network topology is the heart of the experiment and is shown in Figure 3 below. Upon entering the sub-module, entities are

assigned attributes that tell them how many times they must loop back into the system after completing the series of 20 processes and how many processes they must repeat in their loop. In the model, the first 17 or the first 10 queues can be set with process times of zero so the entities can behave as though they were only really encountering 3 or 10 processes in addition to being able to traverse all 20.

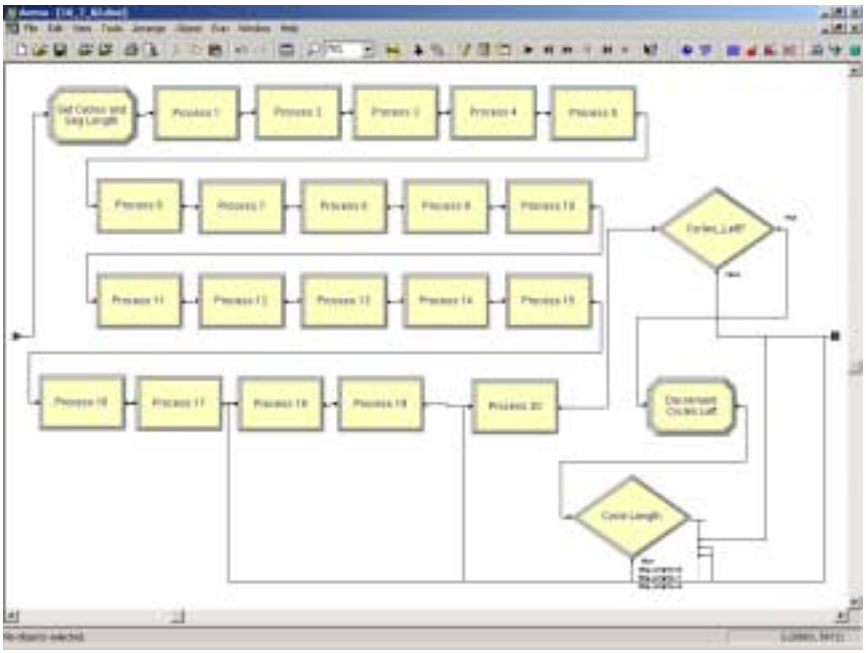

Figure 3: Basic Arena Model View 2

After making their way through all the processes in series, the entities encounter a set of decide modules that direct them to go back to a certain spot in the series (depending of the value of Seg) and repeat those processes a set number of times (depending on the value of K). After completing the main sub-model, each entity's data is written to file via write modules contained in the sub-module shown in Figure 4. Each write module creates a file of a different name that's dependent on the scenario being run. In our first models, the files contained all raw data points for all reps showing the throughput, the rep number, and the cycle time for each entity. These .dat files were then imported into Excel where we could compute across-replication statistics. Later, we modified the data output to print only the mean and standard deviation of the wafers' total time through the system per each replication. The majority of our analysis was performed on the mean cycle times, but having the standard deviation for each replication enabled us to do higher moment analyses on a small subset of our models.

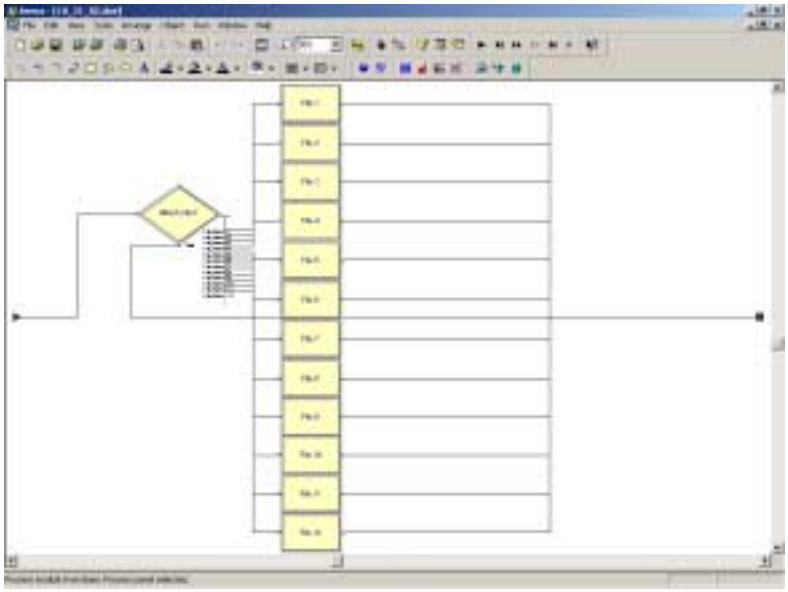

Figure 4: Basic Arena Model View 3

For our exploratory runs, we ran an arbitrary 10,000 entities per replication for 10 reps. The arrival rate was taken from Professors Nelson and Ankenman's "Make-to-order with low volumes and high marginal profits" example in their National Science Foundation proposal which gave an example of a typical silicon wafer manufacturer operating at about 25,000 wafer starts per month. This translated to an average time between arrivals of 0.864 minutes or an arrival rate of 1.16 wafers per minute. We set a uniform mean process time ($1/\mu$) of 0.1 minutes for all 20 process modules and increased the arrival rate (λ) to achieve higher throughputs. Inserting these values into the formula for throughput (Formula 1) gives us:

$$TH = \lambda/\mu = 1.15/10 = 0.12$$

Since we were interested in high throughputs near system capacity where the cycle time rises explosively and becomes highly variable, we needed either a slower service time or a higher arrival rate with which to work. Initial data was run with arrival rates ranging from 6.67 wafers per minute (TH = 0.67) to 9.8 wafers per minute (TH = 0.98).

Before determining the design points and specifications for the final experiment, we fit the data generated by our preliminary runs to the simplest model to which we would eventually fit our final data.

$$Cycle\ Time = C0/(1-TH)^{p}$$

This model is derived from a family of models suggested by Cheng and Kleijnen (1999) where the expected cycle time can be approximated by a function of the system throughput divided by the term (1-TH)^p. Cheng and Kleijnen contend that the best model for the mean CT/TH curve is one where the value of *p* is fixed at one and the numerator contains a polynomial function of the throughput. Consistent with theoretical findings for the M/M/1 queue, Cheng and Kleijnen also suggest fixing the value of the exponent in the denominator for the variance and 2nd moment CT/TH curves (i.e. in the theoretical M/M/1 2nd moment model, the exponent *p2* in the denominator is fixed at 2).

Our first model contains only a constant term in the numerator *C0* that is not a function of throughput as well as the parameter *p* in the denominator. It is our contention that there exist topologies where the best model will consist of a low-order polynomial in the numerator and an exponent in the denominator that is not fixed. Hence, we set out to find topologies that exhibit varying values for *p* in the mean CT/TH curve and exponents in the higher moment curves that are contrary to known M/M/1 results. Intuitively, the parameter *p* in the mean curve gives an indication of the steepness of the graph where the cycle time "explodes." Fitting our trial data from our first Arena model to our cycle time equation showed that there's good reason to believe *p* might change under our re-entrant flow topology and more data needed to be gathered. Increases in the parameter *C0* result in a vertical translation of the cycle time graph on the y-axis but have no effect on the actual shape of the curve.

Figure 5 below shows a plot of our trial run mean cycle times with three queues in series for all other values of Seg and K. It is evident from this plot that *C0* increases with Seg*K, but it is not clear whether or not there is a trend in the values of p. Upon close inspection, it appears that there may be a spread in the graphs and not just a vertical translation, which would indicate changes in *p*. The same trend in *C0* appears with the plots of the 10 and 20 queues data, but a regression analysis was needed to see the effect on *p*.

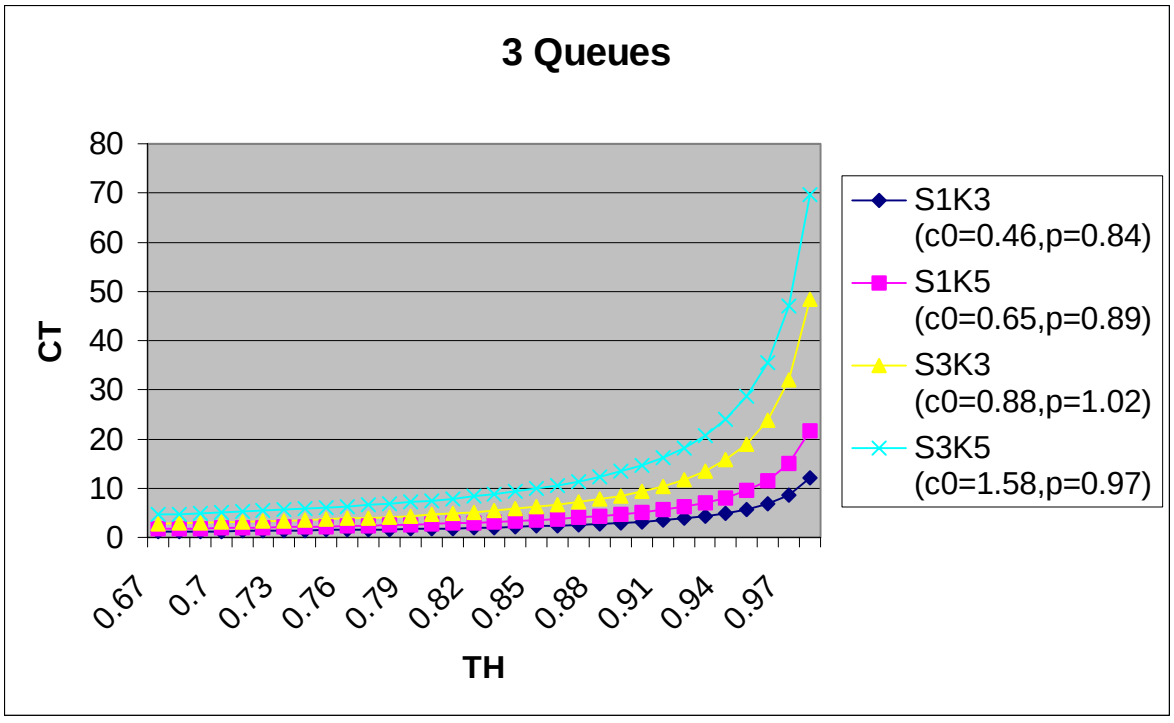

Figure 5: Trial Raw Data Plot

Regression analysis of *p* versus the parameters K, Seg, and Q reveals that *p* tends to decrease as the number of queues increases and may slightly increase with increases in K and the segment length. Figures 6 and 7 on the next page show the main effects of segment length and number of queues as the segment length is increased from one to three and the number of queues from 3 to 10 and then to 20.

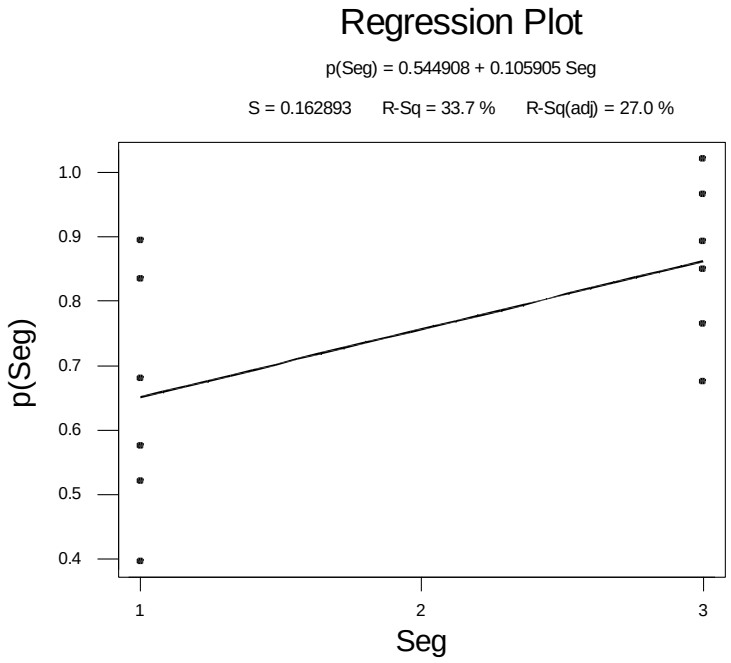

Figure 6: Effect of Seg on p

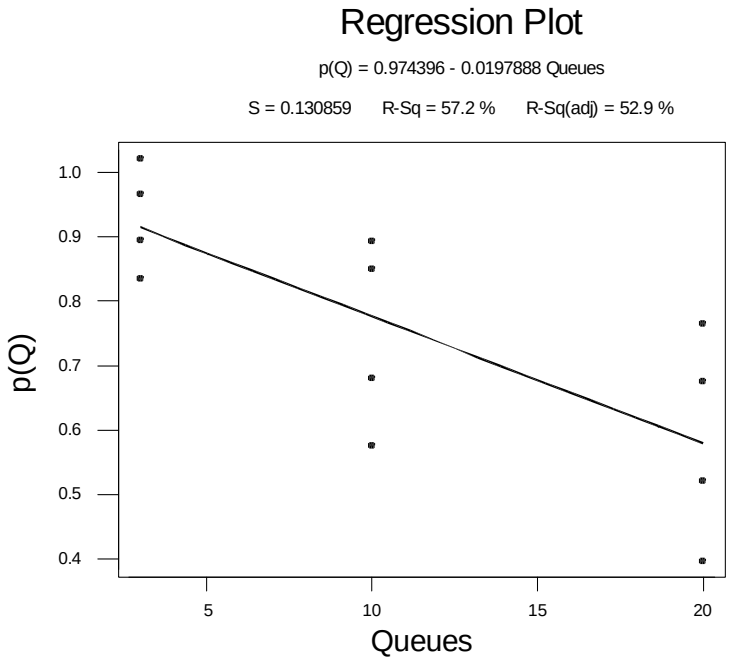

Figure 7: Effect of Queues on p

The regression analysis confirms that there could indeed be a marked effect on *p* with changes in the parameters Seg and Q. Increasing the length of the repeated segment appears to have a strong positive effect on *p* while increasing the number of queues in series seems to have a strong negative effect. The low values of R^2 shown above the charts is due to the fact that there are few data points; however, each point was generated

by a fit of tens of thousands of data points and precise to R^2 values of > 99%. K is shown to have only a marginal positive effect on *p* (see Appendix B Figure B1). (Some error in our initial experiment may have been due to limitations in Microsoft Excel forcing us to compute our across replication statistics for only the first ~70,000 entities or the first seven reps.) Also, our system of multiple queues in series has a considerable "warm-up" period, or a length of time that must elapse once entities start flowing into the system before the system can achieve a steady state. This initial-condition bias at the beginning of runs will generally have the effect of reducing the mean cycle time for the run and can encompass a large amount of data, particularly when the traffic intensity of the system is high.

Determination of Design Points, Run Length, and Number of Replicates

We set nine throughput levels as our experimental design points: [0.70, 0.76, 0.82, 0.88, 0.92, 0.95, 0.96, 0.97, 0.98]. We chose these points because they represent a consentration of throughput levels at the knee and the explosive part of the CT/TH curve while still covering a large range of throughputs. Whitt and others' work with the M/M/1 queue was generally concerned with throughputs only in the extreme high range whereas our fits include a lower range.

At each design point, a scenario with 20 M/M/1 queues in series and no re-entrant flow was exhaustively run for one replication of 100,000 entities. Plots of the raw data for the one replication were analyzed to estimate a warm-up period for the system to reach a steady state at each throughput design point. The arena model was then changed such that a set number of entities at the beginning of each run would be disposed of before their data was written to file. This, in effect, deleted their data before it could be included in the statistics for each replication. Figure 8 shows the number of data that we deleted in each replication for each of the nine design points, as well as the number of total data points that would be generated. According to Nelson, a good rule of thumb is to run enough data such that you run ten times the amount of data that is deleted. Since we only ran 10,000 entities in our trial replications, at the high throughputs, the vast majority of our data points were likely data that should have been deleted.

Design Point	Number Deleted	Number per Replication
0.7	300	3000
0.76	400	4000
0.82	500	5000
0.88	800	8000
0.92	3000	30000
0.95	4000	40000
0.96	6500	65000
0.97	7500	75000
0.98	8500	85000

Figure 8: Design Points and Data Deletion

Using our previous methods, we wouldn't have been able to see all the data for even the first replication (Excel will only show about 70,000 rows), so we revised our data gathering methods in our Arena model. Previously, we had printed each individual entity's time through the system and then averaged these in Excel across the replications. For our final experiment, however, we modified our Arena model to print out the intra-replication average and standard deviation rather than all the raw data points. (See Appendix A, Figure A1) From this small data sample, we were able to extract the across

replication averages as well as the variance of the means across all replications, which are useful in higher moment calculations.

The next task was to determine the number of replications to run at each design point in order to estimate the expected cycle times to a pre-specified precision. Since we had no idea before data was gathered how many replications would be required and how long the experiments would take to run, we did our calculations for the number of reps based on three levels of precision or percent relative error. We defined percent relative error as a percentage of the mean that the upper and lower bounds of a 95% confidence interval were allowed to vary from the across-replication mean. An initial mean and standard deviation were calculated to make this confidence interval and were taken from a run of R0 = 10 replications using the number of entities and warm-up periods that were noted previously. The relative error that we calculated from the largest data set (TH = 0.98) was used to calculate the necessary number of replications for all the other design points so that the final data points would have a uniform precision throughout. Figure 10 on the next page shows a sample for three design points of the spreadsheet used to calculate the number of replications (R1) that were used in the final experiment. The across replication mean for design point 0.98 was 90.83 minutes which resulted in acceptable relative errors of 9.08, 4.54, and 0.91 minutes for precisions of 10%, 5%, and 1%, respectively. From each of these values, we used the formula below to calculate the number of replications needed.

$$\text{Number of Reps} >= ((t(0.025,9)*\text{StdDev}(R0))/\varepsilon)^2$$

In the formula, ε is the relative error. Even at 1% relative error, the number of replications required at the 0.98 throughput point was a manageable 1113, still a much larger data sample than our trial runs (1113 reps * 85,000 points = 94,605,000 data points). Plugging the relative error of 0.91 into the same formula for the other design points gave us the final number of replications, which decreased as the level of throughput decreased down to a point where the four design points with the lowest throughput (0.70, 0.76, 0.82, 0.88) required less than 10 replications to achieve precisions of 0.9. We never computed statistics across less than 10 replications, however.

Throughput	**0.96**			**0.97**			**0.98**		
R0 Mean	44.920901			65.1083629			90.8398878		
R0 Standard Deviation	5.493600527			8.008385938			13.58713299		
% Relative Error	0.1	0.05	0.01	0.1	0.05	0.01	0.1	0.05	0.01
R0 Epsilon	4.49209	2.246045	0.449209	6.510836	3.255418	0.651084	9.083989	4.541994	**0.9083989**
Number of Reps (Min)	8	31	764	8	31	773	12	46	1113
R1 Mean		48.21532			64.32671			90.16123	
R1 Std Dev.		5.10013			8.255812			13.2081	
Number of Reps (Exp)			182			387			1113
h.l.			0.748533			0.83094			0.7799359
95% CI LB			47.46679			63.49577			89.381297
95% CI UB			48.96385			65.15765			90.941168

Figure 10: Determination of Number of Reps

Lastly, a run was performed at each design point with the same model using the calculated number of replications R1 and 95% confidence intervals calculated to determine if the relative error was indeed at most 0.91 in all cases. Three of those half widths are shown in the bottom row of Figure 10. For all the design points, the relative error proved to meet the 0.91 requirement. Because of the extraordinary size of the data being generated in Arena, running all nine of these experiments took several days, even on a 550 MHz Pentium processor machine with 512 MB of RAM. The duration of the final experiment, running all 12 scenarios varying K, Seg, and Queues over all the design points ended up being more than two weeks. The bulk of the time was spent running all 12 runs of 1113 replications at the 0.98 design point.

Once the design points, run length, and number of replications were determined, they became a model for future experiments that encompassed other relevant behaviors of semiconductor manufacturing systems, including batching, machine failures, part classes and priorities, and different queueing disciplines (other than FIFO).

After all the data was gathered, it was fit to three models using S-Plus, a statistical analysis software program. In addition to the model we used in our trial experiment, two other models were fit, one containing a linear term in the numerator and one containing both a linear and quadratic term in the numerator. Using higher order functions of the throughput in the numerator can account for variability as well as different shaped curves such as the 'U' shaped curve produced when a system uses batching. Our objective, however, is to keep the numerator as simple as possible and allow the exponent in the denominator to vary. The three models are shown below.

Model 1 (Quad): *Cycle Time = (C0 + C1*TH + C2*TH^2)/(1-TH)^p*

Model 2 (Lin): *Cycle Time = (C0 + C1*TH)/(1-TH)^p*

Model 3 (Const): *Cycle Time = C0/(1-TH)^p*

Re-Entrant Flow Data and Analysis

For each design point and each combination of K, Seg, and Q, we ran simulations and gathered the mean cycle times across all replications. Figure 11 shows a data sample for all scenarios with Q = 3.

TH	3Q_Seg1_K3	3Q_Seg1_K5	3Q_Seg3_K3	3Q_Seg3_K5
0.7	1.2472687	1.9061372	2.9988242	5.1131135
0.76	1.5043755	2.2739812	3.7337735	6.2660813
0.82	1.8905167	2.8780871	5.1687669	8.3027753
0.88	3.0007819	4.3552233	7.4174158	12.9282456
0.92	3.968445727	6.712389848	11.67160461	18.35836967
0.95	6.331076013	10.18506754	18.21118925	29.72959501
0.96	7.921448934	12.70238566	22.67479368	37.74253442
0.97	10.28420829	17.17452563	29.43059203	50.08778979
0.98	15.21267733	25.05895314	44.53989092	74.59431674

Figure 11: Data Sample

A CT/TH plot of the simulated mean data (Figure 12) shows groupings of curves by K*Seg as we had seen in preliminary trials. In this graph, however, there doesn't appear to be much difference in the curve shape.

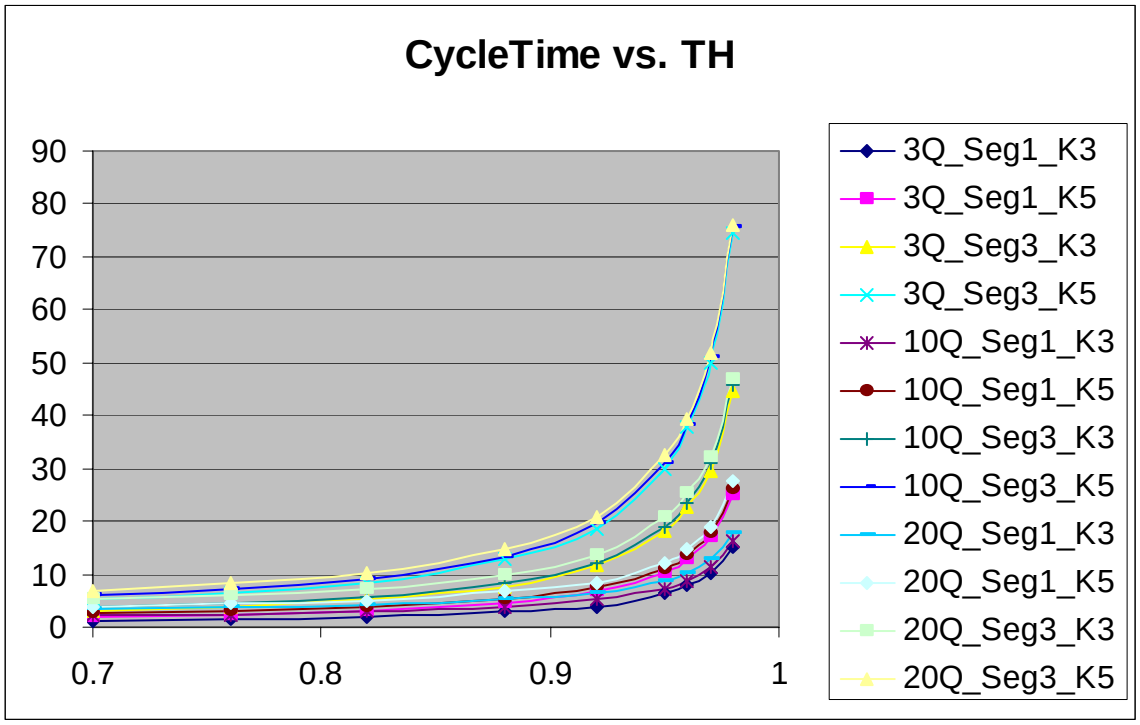

Figure 12: Final Simulated Data CT/TH curves

When fit to our three models, the mean data for each of the 12 scenarios fit with R^2 values exceeding 99% in almost every case. R^2 is an indicator of the percentage of the variation in the data that's accounted for by the model. Likely due to the exhaustive run length and number of replications in our simulations, these precise fits mean that the *p* values given for each model in Figure 13 are extremely accurate. Contrary to our initial hopes, however, it seems that in very few cases does *p* actually stray far from one, especially with Model 2 (linear term in the numerator). Model 1 (quadratic numerator) has a few values that are significantly higher than one and Model 3 (constant numerator) shows a few values that are significantly low.

K	Seg	Q	p (Model 1-Quad)	p (Model 2-Lin)	p (Model 3-Const)
3	1	3	0.982673	0.967076	0.942171
5	1	3	0.944810	0.971543	0.969974
3	3	3	0.997261	0.972476	0.982402
5	3	3	0.983012	0.995747	0.995573
3	1	10	1.060690	0.97643	0.812517
5	1	10	1.042790	1.00702	0.896176
3	3	10	0.998455	0.985962	0.947549
5	3	10	0.955875	0.985099	0.969298
3	1	20	1.179300	0.977417	0.664548
5	1	20	1.113160	1.03111	0.803138
3	3	20	0.925523	0.946443	0.86654
5	3	20	0.975076	0.975218	0.924118

Figure 13: Final fitted values of p

Factorial analysis of effects of K, Seg, and Q on *p* from Model 3 show main effects that are not as strong as we had found in early data (See Figure 14). With Model 2 and 3, these effects became less strong as *p* hovered more closely around one and were completely reversed in some cases with Q actually showing a positive effect on *p* in Model 1 (Figure 15 and 16).

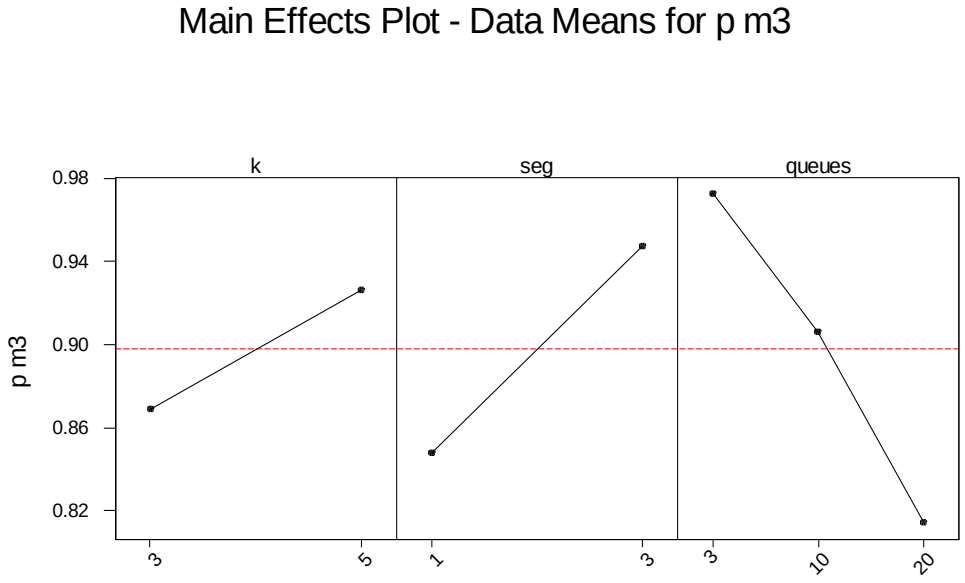

Figure 14: Main effects plot for *p* using Model 3

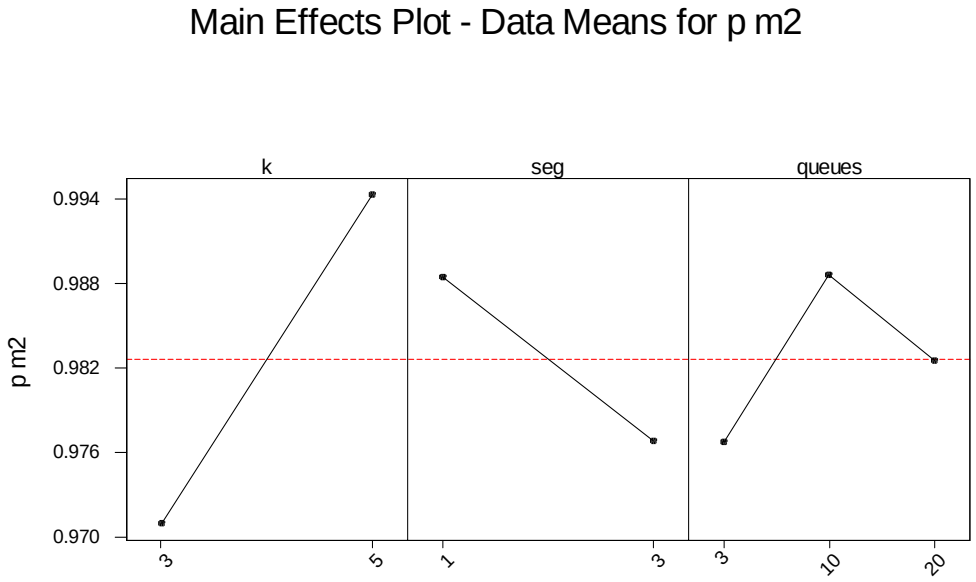

Figure 15: Main effects plot for *p* using Model 2

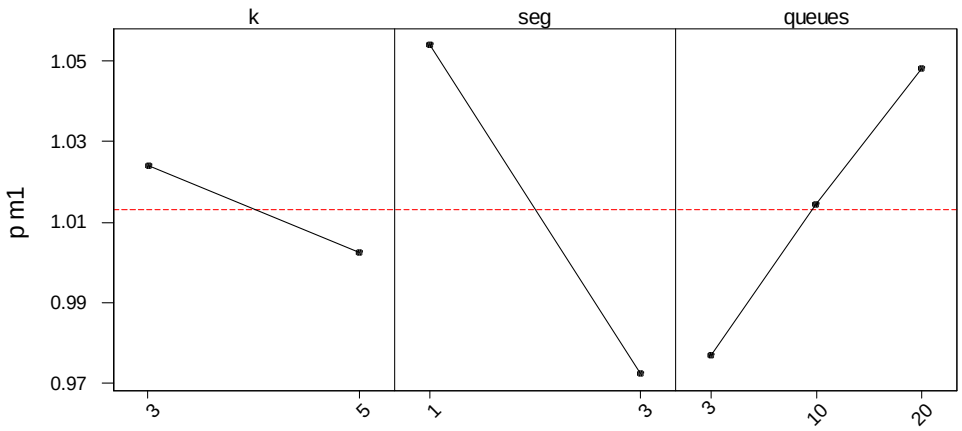

Figure 16: Main effects plot for *p* using Model 1

With Models 2 and 3, it appears that there is a slight interaction between Q and Seg, but the scale is too small for it to be significant. Even in the cases where the main effects look significant on the plots, the range of *p* values is closely centered around one. See Appendix B for the interaction plots for all three models.

Plotting the values of *p* for all three models across all the scenarios confirms that Model 1 tends to generate values for *p* that are slightly higher than one, Model 3 generates *p* values below one, and Model 2 stays consistently in the middle (Figure 17). Although the R^2 values for Model 1 are slightly better than the other two models, it is clear that with Model 2, we were able to accurately fit a CT/TH curve with *p* not straying far from one. The R^2 values we achieve with Model 2 are, in fact, better than R^2 values generated if the value of *p* is fixed at one in the model (see Appendix C), but the fitted values of *p* are close enough for one to make the assumption that this class of topologies need not require values of *p* different from one if the proper function of throughput is present in the numerator. Fixing *p* at one causes the model with the linear numerator to fit best. Further analysis needs to be done to examine some of the trends in *p* evident in models 1 and 3 seen in Figure 17 and also to delve into residual patterns evident in our data (Appendix C), but that is left for another study.

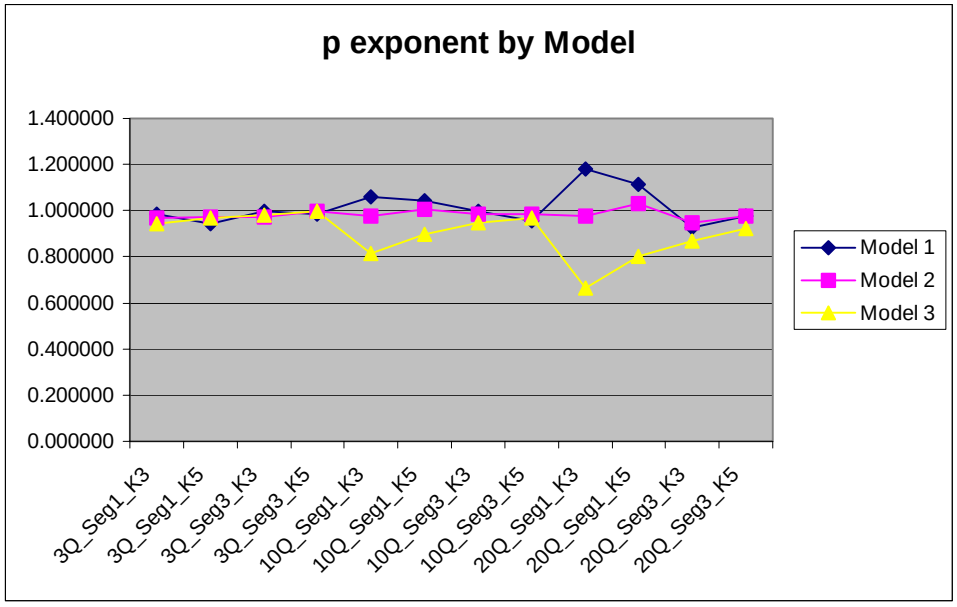

Figure 17: Plot of *p* by Model over all scenarios

Although we show here that it is possible to adequately fit a model to the mean curve where the exponent *p* is equal to one, there still remain questions as to the nature of the variance and higher moment curves for these topologies. We examined higher moments only for the two extreme cases in our re-entrant flow models: [Q = 3, Seg = 1, K = 3] and [Q = 20, Seg = 3, K = 5].

We fit models for the variance of the mean cycle time (multiplied by the number of replications at each design point, since they are all different), raw and centered 2nd moment of the cycle time, and the variance of the 2nd moment. Once the data was run, and only the replication mean and standard deviation recorded, these were the most measures we could extrapolate. Known results at high throughputs with the M/M/1 queue provide the following models:

$$1^{st}\ Moment\ Variance = 1/(1\text{-}TH)\text{^}4\ [exponent = q]$$

$$Raw\ and\ Centered\ 2^{nd}\ Moment = 1/(1\text{-}TH)\text{^}2\ [exponent = p2\ (raw),\ p2'\ (centered)]$$

$$2^{nd}\ Moment\ Variance = 1/(1\text{-}TH)\text{^}6\ [exponent = q2]$$

Very little work has been done analyzing CT/TH higher moment curves, especially fit to such low throughputs as 0.7. We speculate that just as with the mean curves, there will be topologies and situations for which the value of the exponents should not be fixed as in the M/M/1 case. We also fit each data set to models with numerators of varying complexity, ranging from only a constant term to a quadratic function of throughput. As was stated earlier, the exponent *p* in the mean CT/TH model can intuitively be thought of as representing the "steepness" of the incline at the knee of the curve. Although their effect is not quite as direct, the exponents *q1, q2, p2, p2'* can also be thought of as effecting this "steepness."

Figure 18 below shows a data sample from the 0.7 throughput design point of the 3QSeg1K3 re-entrant flow model. The columns labeled *Rep #, Mean, and Std Dev* are

printed to the output files after each run of the simulation. The *Variance* column is calculated by squaring the standard deviations from each rep. The *2nd Moment* column is generated by multiplying the variance from each replication times the ratio (m-1)/m, where m is the number of data points for that particular replication. This product is then added to the square of the mean of the replication to get the 2nd moment. Each throughput design point is then associated with the variance of its replication means multiplied by the number of points per replication (Var of Means*(rep length) or 1st Moment Variance), the average variance across all replications (Mean Variance or the Centered 2nd Moment), the mean of each replication's 2nd moment (Mean 2nd Moment or Raw 2nd Moment), and the variance of those 2nd moment measures (2nd Moment Variance).

Throughput	**0.7**
Mean	1.2472687
SD of Means	0.076735482
Rep Length (post delete)	2700
Var of Means*(rep length)	15.89850219
Mean Variance	0.775427679
Mean 2nd Moment	2.336119194
2nd Moment Variance	0.165536449

Rep #	Mean	Std Dev	Variance	2nd Moment
1	1.223589	0.792845	0.628603	2.125540419
2	1.318832	0.980681	0.961735	2.70069687
3	1.334216	0.996883	0.993776	2.773539985
4	1.157382	0.745361	0.555563	1.89489035
5	1.202565	0.884719	0.782728	2.228600389
6	1.175991	0.795883	0.63343	2.016149978
7	1.208202	0.838797	0.70358	2.163071895
8	1.396589	1.106963	1.225367	3.175374079
9	1.226463	0.821194	0.67436	2.178321313
10	1.228858	0.77145	0.595135	2.105006666

Figure 18: Sample Higher Moment Data Sample from 3QSeg1K3 scenario

The variance of the means curve was fit only to the model with a constant in the numerator. As shown in Figure 19, our data fit with an exponent *q* of very near four, which is the predicted value for a single M/M/1 queue. The next six columns on the graph represent the *p2* and *p2'* fits or the Raw and Centered 2nd moment exponents. There appear to be slight differences in the value of the exponent depending on the degree of the numerator, but all are very close to the predicted value of two, especially those exponents fit to the constant numerator model. With the fits for the 2nd moment variance (the last three columns in the graph), there are marked differences in the values of the exponent *q2*. It seems that as the numerator becomes less complex, the value of *q2* gets closer to its predicted M/M/1 value of six. None of the three models' results come as close to the M/M/1 prediction as the previous measures had; they are all, in fact, well short. This difference could be the result of many factors, the number of queues in series, re-entrant flow, fitting data associated with relatively low throughputs, or simply lack of data for a measurement that is highly variable.

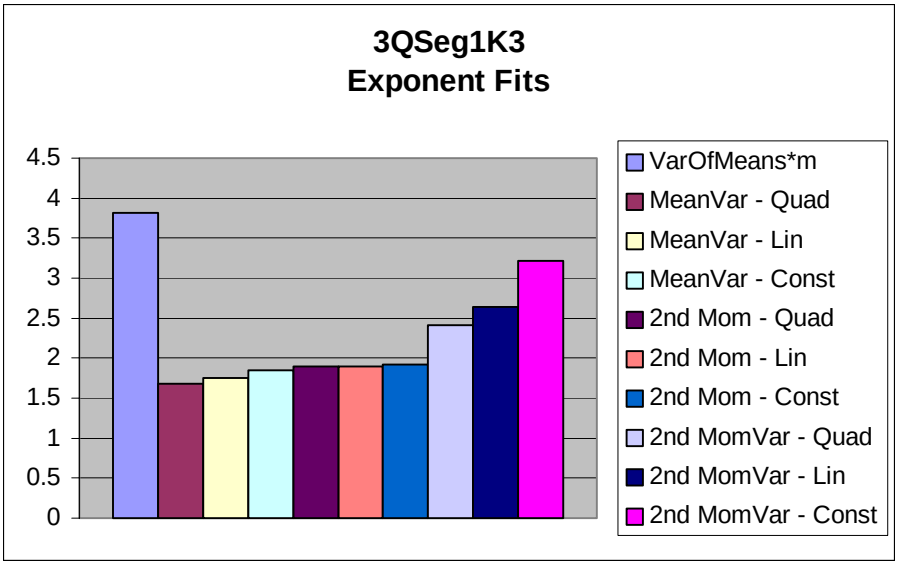

Figure 19: Higher moment exponent fits for 3Q Seg1 K3 re-entrant flow model

We did the same calculations and fits for the 20QSeg3K5 scenario and found some similar results (Figure 20). The exponent *q* in the 1st moment variance model fits almost perfectly to the predicted value of four, and the values of *p2* and *p2'* show much the same pattern as the 3QSeg1K3 model, getting closer to two as the order of the numerator decreases. The quadratic and linear numerator models fit to the raw 2nd moment data exhibit a value of *p2* closer to three, but again there exists at least one case (model with a constant numerator) where it's possible to get a good fit with a value of two as in the M/M/1 case. (Incidentally, all the exponent fits for the higher moment models in the 3Q and 20Q case achieved R^2 values well over 99%.)

S-Plus couldn't fit a quadratic or linear numerator model to the 2nd moment variance data for the 20Q case, but the fit of the constant model reveals a *q2* value much greater than six and clearly greater than the values generated with the 3Q case. This indicates that the true values of *q2* for these topologies could indeed be different than the ideal M/M/1 case and are a function of either the number of queues in series or re-entrant flow.

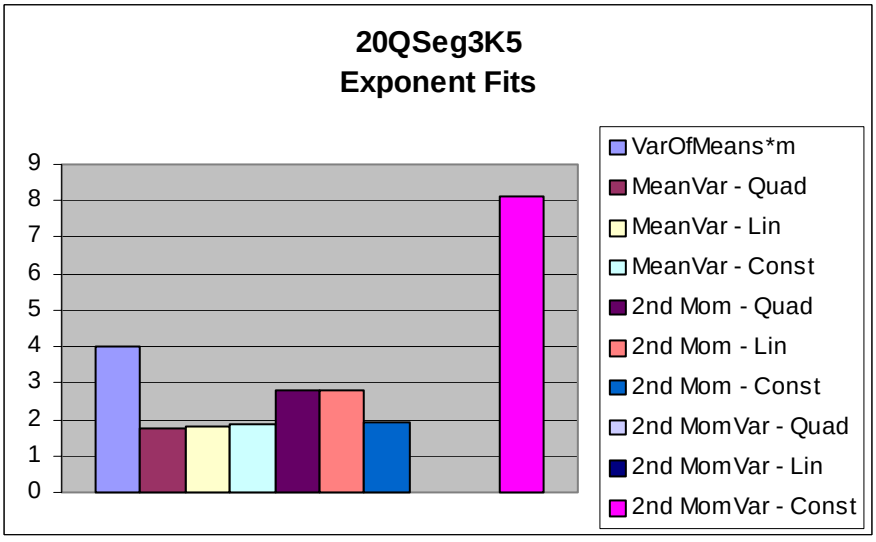

Figure 20: Higher moment exponent fits for 20Q Seg3 K5 re-entrant flow model

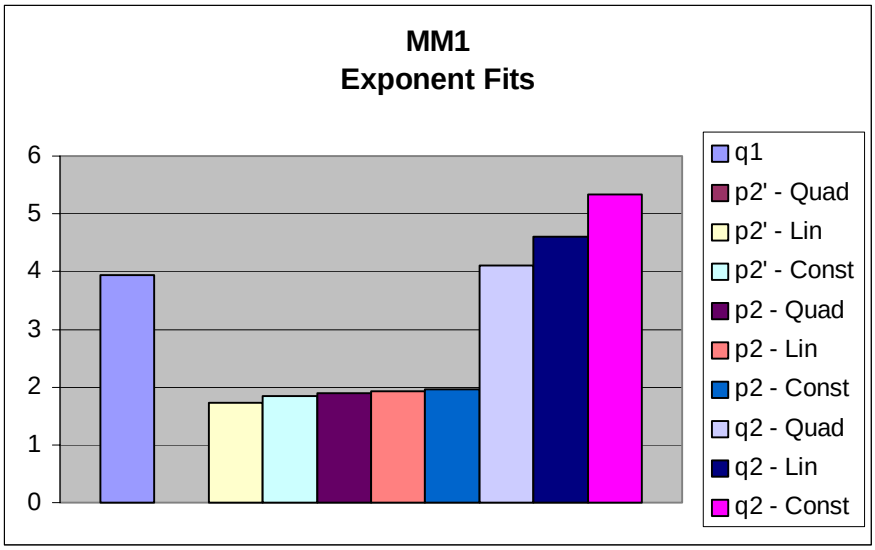

Figure 21: Higher moment exponent fits for a single M/M/1 case.

To see how a single M/M/1 queue will behave farther down the CT/TH curve where we were running our models, we generated the same amount of data at the same design points as we had previously, but with a single M/M/1 queue. Using the same procedures, we fit this data to our set of metamodels. Figure 21 above shows the results of those fits. The values of *q, p2*, and *p2*' look exactly as we suspected they would based on our results with the re-entrant flow models. The values of *q2*, however, do not quite reach the value of six that is predicted for a single M/M/1 queue at high throughputs. The last fit with the constant numerator model comes pretty close, but there is evidence enough in this plot to suggest that higher moment curves can posses exponent values different than what has been widely accepted when one looks only as far down the throughput curve as 0.7; this is true even in the simplest case of the single M/M/1 queue. All of our higher moment CT/TH curves can be found in Appendix E.

Batching and the Location of the Bottleneck Queue

In addition to the re-entrant flow model with K, Seg, and Q as parameters, we experimented with several other topologies to see if they might have a more drastic effect on the exponent *p* in the mean CT/TH curve in our family of models.

One other topology we tried back when we were first experimenting with re-entrant flow was batching. Batching involves waiting for groups of entities to accumulate up to a certain batch size before starting a process or group of processes. The group of entities, or batch, then goes through one or more processes before being broken down again into individual entities. We learned that batching was a common occurrence in the world of silicon wafer fabrication and wafers might be put into batches, un-batched, and re-batched several times during the course of manufacturing.

Using 20 processes in series as before, we produced a model where entities could be grouped into batches, run through a process as a batch, and then be separated at three locations in the series. In all cases, a string of 5 generic processes without batching were run through first followed by a batch process, a regular process, another batch process, another regular process, and finally a third batch process. This left 10 out of 20 total queues in series at the end that were either all on (the entities passed through them) or all

off (their processing time was set to zero). This factor was to see if having standard M/M/1 queues after batching would act as sort of a buffer, limiting the batching effect. Finally, we tested the system with batch sizes of 10, 50, and 100 entities. The Arena model we used for batching as well as the experimental design matrix can be found in Appendix D.

An additional experiment was run using our same batching model but we added a bottleneck queue at to different places within the system. The bottleneck queue was set at a fixed throughput level even as we varied the overall system throughput. We know from Whitt's work that if one queue in a system is a bottleneck, or is congested, then the overall system CT/TH behavior will be due to the congestion in that single queue. Just to verify that this principle isn't broken when the system employs batching, bottleneck queues were placed in three locations throughout our model: before the batch processes, in between two batch processes, and after the batch processes. In all these cases, the batch size was fixed at 100 and there were two batching processes turned on. That design matrix can also be seen in Appendix D.

With batching in a system, CT/TH curves will be 'U' shaped due to long cycle times at low throughputs where entities wait long periods of time for batches to build up. This curve shape was evident for data from runs of our Batching and Batching/Bottleneck models (Figure 22). The curves shown below were created from runs of 10,000 entities over 10 replications. As was articulated in our NSF proposal, we know that the family of models to which we're fitting data with a first order denominator can account for the 'U' shaped curves due to batching by adjusting the functional form of the numerator. At higher throughputs where estimation of the cycle time is most critical, however, this downward slope is not a factor so our models don't even necessarily have to account for them. Furthermore, we were unable to see any unexpected behaviors due to the presence or location of the bottleneck queue. Because of these facts, we decided not to expend any further experimental effort on the Batching/Bottleneck models, as we were not likely to need to adjust the exponent p to get good fits at high throughputs. More data has to be run in order to fit any higher moment models to the batching and bottleneck data, as well.

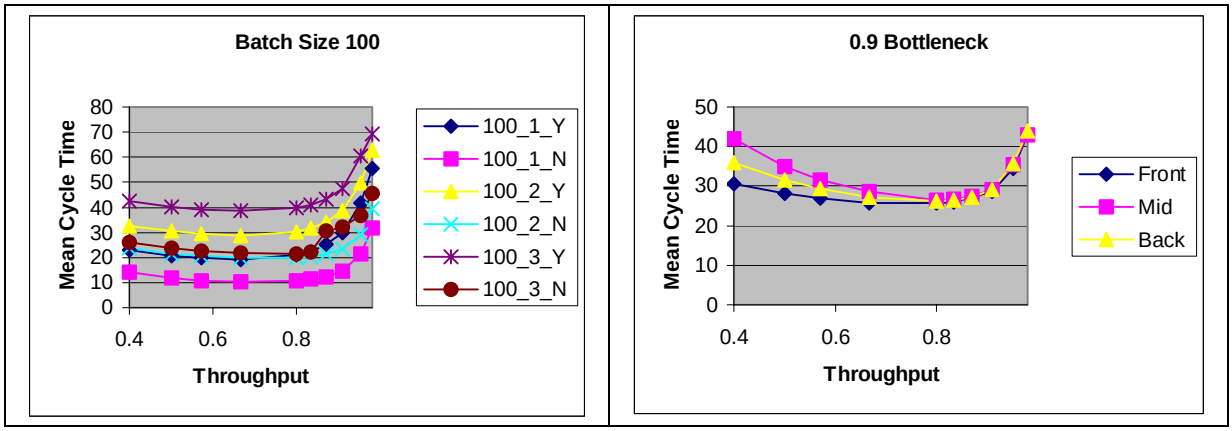

Figure 22: Data from Batching and Batching/Bottleneck Models

Alternate Queueing Disciplines and Failures

After drawing conclusions from our re-entrant flow model, we experimented with two queueing disciplines that are proven to fit values of p significantly different than one in the case of a single M/M/1 queue using a First-In-First-Out discipline: LIFO and SPTF. In Arena, these disciplines can be easily implemented as properties settings for

each process module's queues. SPTF is achieved by assigning the random processing time to each entity as an attribute before it arrives at each process. Each queue is then set to give priority to entities whose processing time attribute is the smallest.

To examine the effect of these different queueing disciplines on the CT/TH curves, we set a baseline scenario of 20 queues in series (as we did to determine the warm-up period and run length previously) and compared those results to scenarios where different queues within the series utilized the new disciplines. We collected data for the following scenarios: one queue in the middle using LIFO or SPTF (we used the 10^{th} queue), five queues in the middle (8^{th}, 9^{th}, 10^{th}, 11^{th}, and 12^{th} queues), five queues spread out evenly (1^{st}, 5^{th}, 10^{th}, 15^{th}, and 20^{th} queues), and all 20 queues using LIFO or SPTF as their queueing discipline. In the interest of time, 10 replications were run with our standard set of run lengths. The mean data from these runs is shown in Figure 23.

TH	0.70	0.76	0.82	0.88	0.92	0.95	0.96	0.97	0.98
None	6.72	8.19	10.99	17.66	24.17	41.48	44.92	65.11	90.84
LIFO.one	6.72	8.20	10.99	17.65	24.08	41.50	46.04	59.46	90.79
LIFO.fivemid	6.73	8.20	10.99	17.59	24.06	41.48	46.04	59.28	90.36
LIFO.fivespread	6.72	8.20	10.99	17.60	24.09	41.30	46.04	59.38	90.61
LIFO.all	6.73	8.19	11.02	17.53	24.04	40.98	45.82	59.08	88.28
SPTF.one	6.55	7.79	10.89	16.50	24.38	38.01	46.52	56.62	85.50
SPTF.fivemid	6.18	7.53	10.15	13.78	20.60	33.14	39.59	53.49	70.40
SPTF.fivespread	6.03	7.57	9.64	14.16	20.83	32.52	39.47	51.30	69.18
SPTF.all	4.47	4.97	5.69	6.86	8.32	10.69	11.96	14.04	17.71

Figure 23: LIFO and SPTF across replication mean data

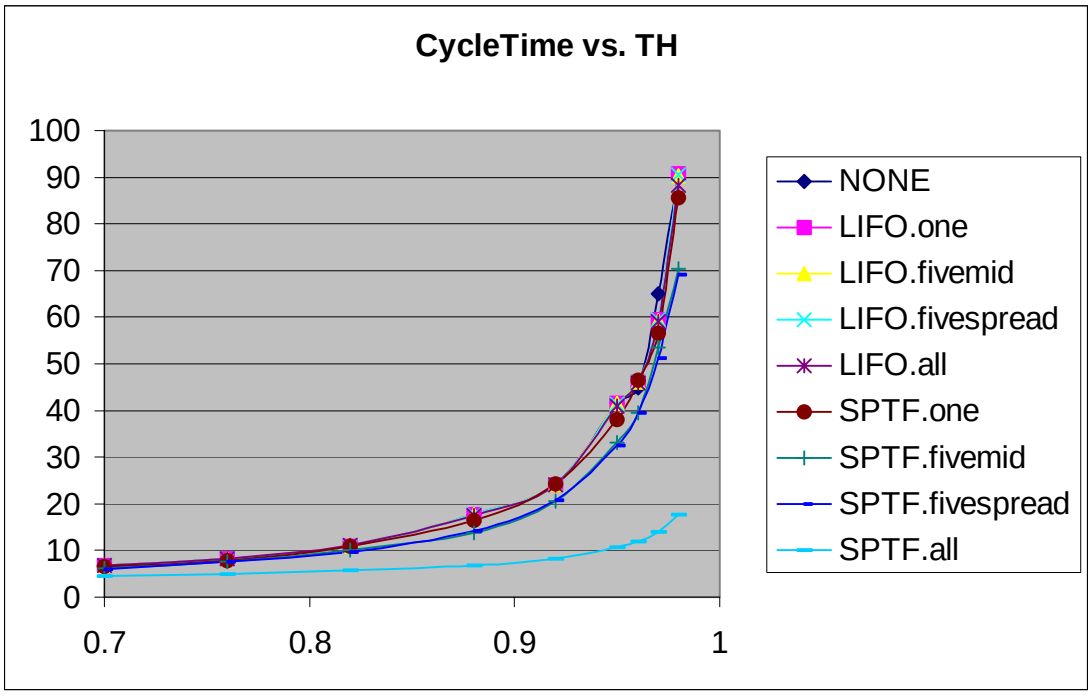

Figure 24: LIFO and SPTF CT/TH curves

One cannot tell much about the effect of LIFO by simply looking at the numbers, but it is clear from the SPTF data that SPTF drastically reduces the mean cycle time as

more of the queues in series utilize that priority system (see the SPTF.all CT/TH curve in Figure 24). In both cases, there also doesn't appear to be much evidence that using a group of queues in the middle or spread out over the whole series makes a difference.

From previous work with a single M/M/1 queue, we know that using LIFO will create CT/TH curves with values of *p* greater than one, and SPTF CT/TH curves will have values of *p* significantly lower than one. Our objective is to see if this effect is still achieved with 20 queues in series, and how the value of *p* changes depending on the functional form of the numerator in the model. In addition to the three models we fit in previous experiments, for LIFO and SPTF we tested a fourth model with a cubic throughput term in the numerator. That model is as follows:

$$Cycle\ Time = (C0 + C1*TH + C2*TH^2 + C3*TH^3)/(1-TH)^p$$

When our data is fit to the four models, it is evident that the values of *p* can be drastically different depending on the terms in the numerator. We figured to find values of *p* in the LIFO case that were much greater than one, but we only find that to be the case when the LIFO data was fit to the fourth model with the cubic term in the numerator. For all the models, it appears to not make a difference how many queues use LIFO, and in the case of the model with only a constant term in the numerator, *p* stays relatively constant even when comparing the case with all queues using FIFO. Figure 25 shows the values of *p* for all four models across all five scenarios. To verify that all the points in the plot below were relevant, we look at the R^2 values for all the model fits (See Figure 26).

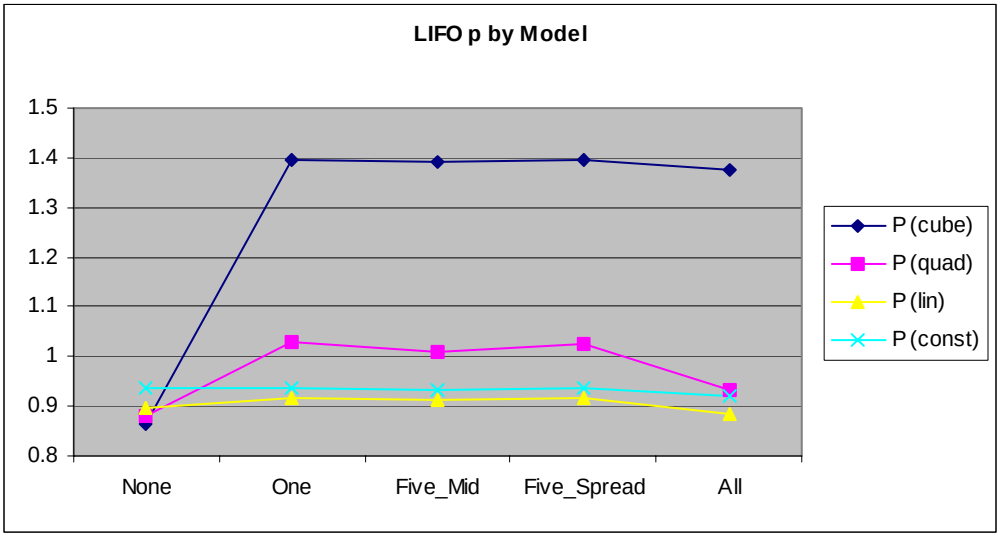

Figure 25: LIFO p-fits

Although the value of R^2 increases as more terms are added to the model (an indication of a higher percentage of the data's variability being explained by the model), the increase is small relative to the average magnitude of the measure. Even in the case of the model with only a constant in the numerator, the model accounts for over 99.76% of the data's variability. In this case, we shown once again that it's possible to fit a model to our data extremely well without changing the value of *p* far from one. It's also possible, however, to do this without overcomplicating the numerator.

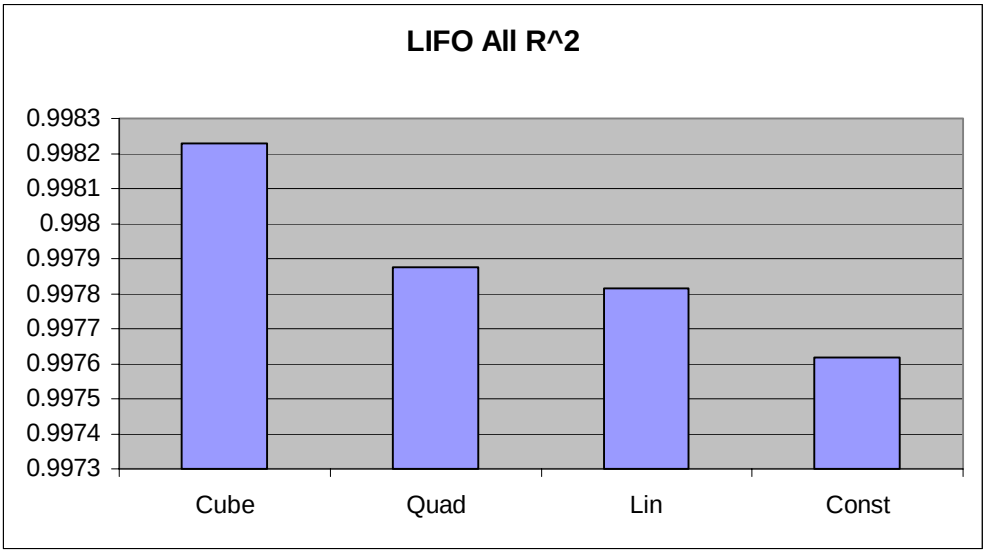

Figure 26: LIFO R^2 values

In the plot of our fits for the SPTF scenarios, there is less of a dependence on the model being used as to the value of *p*. For the scenarios tested with one and five of the queues using SPTF *p* fits are shown that vary inversely with the number of terms in the numerator. As before, however, for all points the fits are so precise that there isn't a clear best fit in these cases. It is evident from Figure 27, however, that the value of *p* in the model will be significantly less than one (approximately 0.6) when all 20 queues in the series employ SPTF as a queueing discipline no matter which model we fit.

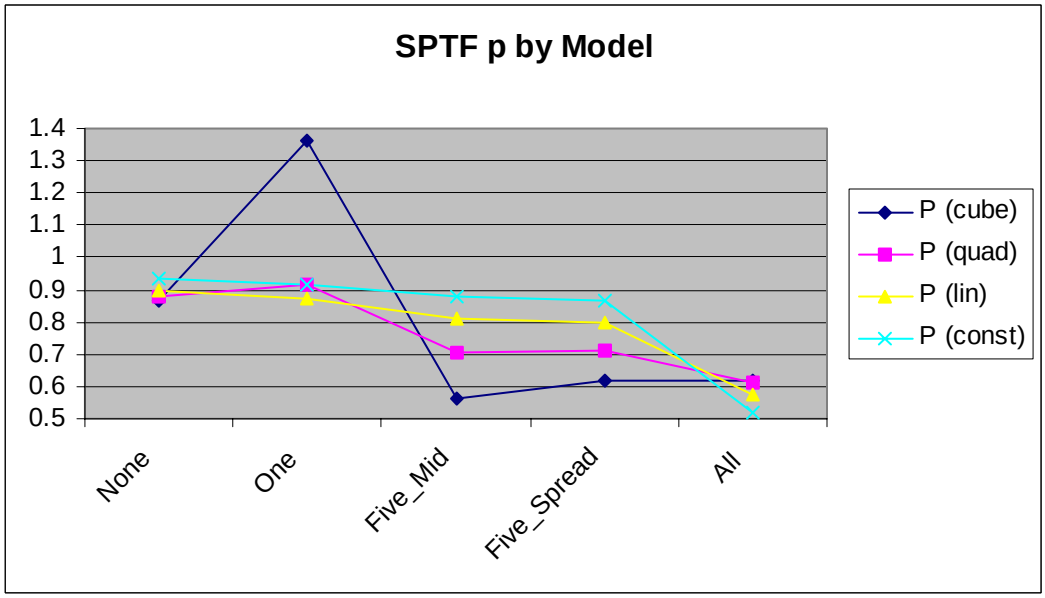

Figure 27: SPTF p-fits

LIFO and SPTF are both examples of queueing disciplines that are common in all applications of queueing, across industries. Even so, in industry it is not likely that either of these disciplines is executed in their purest forms. It is likely, however, that someone out there uses a similar sort of priority system based on the order of part arrivals or the expected processing time for a particular class of parts. Oftentimes, methods of

prioritizing within queues will have nothing to do with when an item arrives or how long it will take to process. Priority is often given based on approaching deadlines for a product, reducing machine setup time between part classes, or systems as arbitrary as personal preference. To account for the most complicated prioritization process possible in a queue, we decided to examine "Service in Random Order" or SIRO as a sort of queueing discipline. This is a discipline that we contrived where the next item to be processed is randomly selected from the current items residing in the queue. We tried two methods of implementing these phenomena in our Arena models, generated data for both, and fit the data to our four models. Both methods depend on the use of an Arena module called "Remove" that's capable of extracting an entity from any queue referenced by its rank or its position in the queue. To randomly pick an entity from a queue, an assign module generates a random number between zero and the number of entities currently in the queue then sends that value on to the remove module telling it which rank to pull. The methods differ in how entities are placed back into the queue from which they are removed and whether the prompt to remove an entity from a queue comes before an entity gets to that queue or after an entity finishes the process. Method number one is shown in Figure 28.

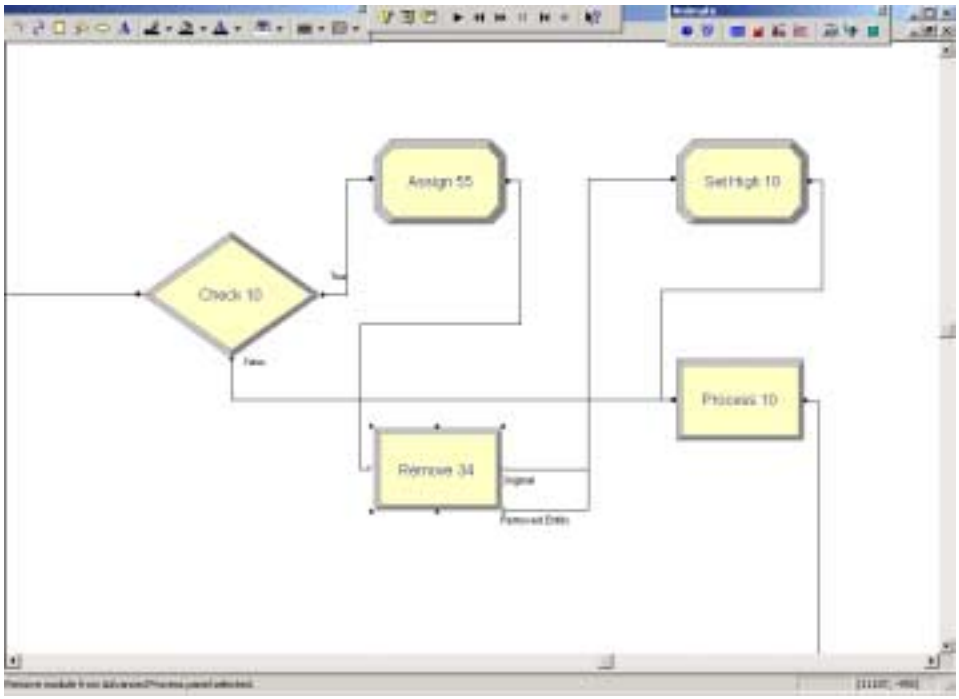

Figure 28: First SIRO model

In the first method, a decide module (labeled "Check 10") checks to see if there are currently any entities in the queue for Process 10. If there are no entities currently in the queue, the entity moves ahead into the queue. If there are entities in the Process 10 queue, the entity goes through an assign module ("Assign 55") where it is assigned randomly a rank of one of the entities already in the Process 10 queue. As is proceeds through the remove module, an entity is removed from Process 10's queue. The original entity proceeds to the rear of the Process 10 queue while the entity that was removed goes through another assign module ("Set High 10"). This final assignment module increments a global variable and assigns it as an attribute X to the entity passing through.

The queueing discipline in each process is set to sort by highest X value and since time the global variable is incremented each time an entity is removed, there is no other entity in the system with a higher X value at that moment. The randomly removed entity, therefore, is processed next.

Even more so than in the case of LIFO, the values of *p* that we fit for this first SIRO method show great dependency on the number of terms in the numerator of the model. Across all scenarios for SIRO, increasing the number of terms results in a lower value of *p*. The scenarios with five queues using SIRO showed a spread of nearly 0.3 in the value of *p* from the model with a constant numerator to the model with a cubic throughput term. All four models fit with R^2 values over 99%, and the constant numerator model kept its *p* value very near one across all scenarios (See Figure 29).

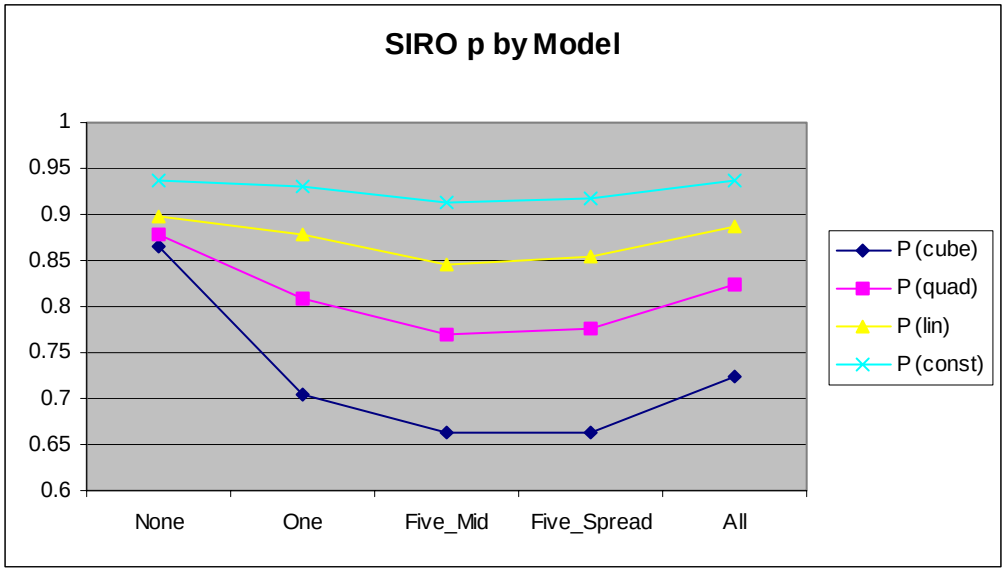

Figure 29: SIRO (Method 1) p-fits

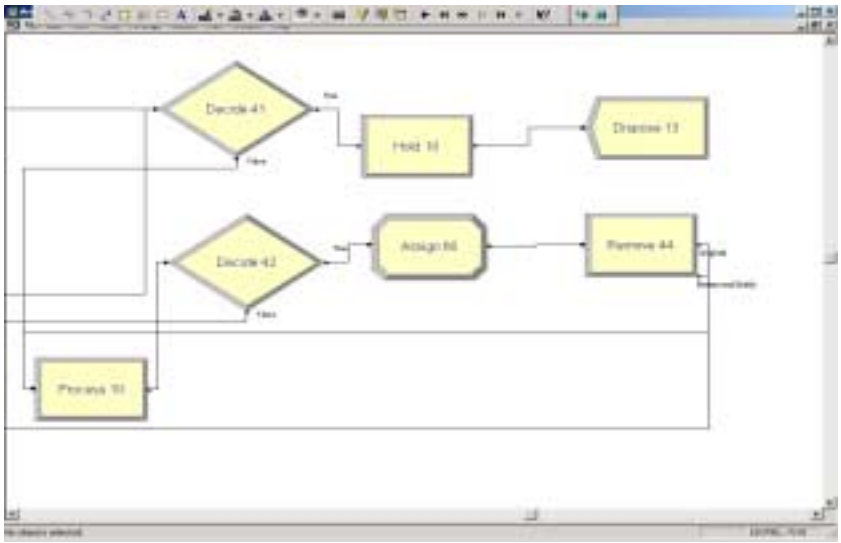

Figure 30: Second SIRO model

Figure 30 shows our second attempt at creating true SIRO behavior (developed by Dr. Nelson). In this method, a decide module ("Decide 41") checks the Work In Process

(WIP) of the upcoming process module ("Process 10") to see if there is an entity actually being processed (not just one residing in the queue as before). If it is determined that the process is currently busy, instead of being put in the queue, the entity goes to a hold module ("Hold 10") that basically acts as a free-standing queue independent of the process it's holding entities for. If the process WIP is zero, the entity goes into the process where it will enter without having seen any time in a queue. When an entity finishes going through the process, it goes through another decide module ("Decide 42") that counts to see if there are entities waiting in the hold module (equivalently Process 10's surrogate queue). If there are none, the entity proceeds to the next process in the series. As in the previous method, if there are entities waiting, a random rank is assigned and an entity of that rank is removed from the queue. The removed entity then goes directly into the process (there is no queue) and the entity that had passed through the remove module to trigger the event continues on through.

TH	0.70	0.76	0.82	0.88	0.92	0.95	0.96	0.97	0.98
SIRO.one	6.61	8.57	11.48	15.93	25.25	39.10	48.86	64.64	90.37
SIRO.fivemid	6.61	8.56	11.46	15.80	25.13	38.83	48.63	62.95	87.46
SIRO.fivespread	6.61	8.57	11.49	15.89	25.22	38.95	48.54	64.03	88.31
SIRO.all	6.61	8.57	11.49	15.93	25.25	39.15	48.87	64.78	91.07
SIRO(2).one	6.96	8.74	11.57	16.54	25.03	40.67	50.07	64.79	93.33
SIRO(2).fivemid	7.86	9.68	12.84	19.23	28.22	44.50	56.57	77.33	100.55
SIRO(2).fivespread	7.79	9.69	13.04	18.03	29.07	44.74	54.47	73.62	101.69
SIRO(2).all	11.04	14.00	18.28	25.63	38.30	59.03	72.26	99.49	134.40

Figure 31: SIRO raw data (both Methods 1 and 2)

As compared to the first SIRO method, the second attempt shows more of an increase in the mean cycle time values as we would expect due to the random priorities given to the entities. When fit to our four models, however, the data from the second SIRO method produces values of p that are similar to those generated by the first method. The values are fairly close across all the models, and once again we saw a trend in the values of p decreasing as the number of terms in the numerator increased. Also, as in the case of the first SIRO method, the model with only a constant in the numerator produces a value of p that stays close to one even as more and more processes in the series use SIRO.

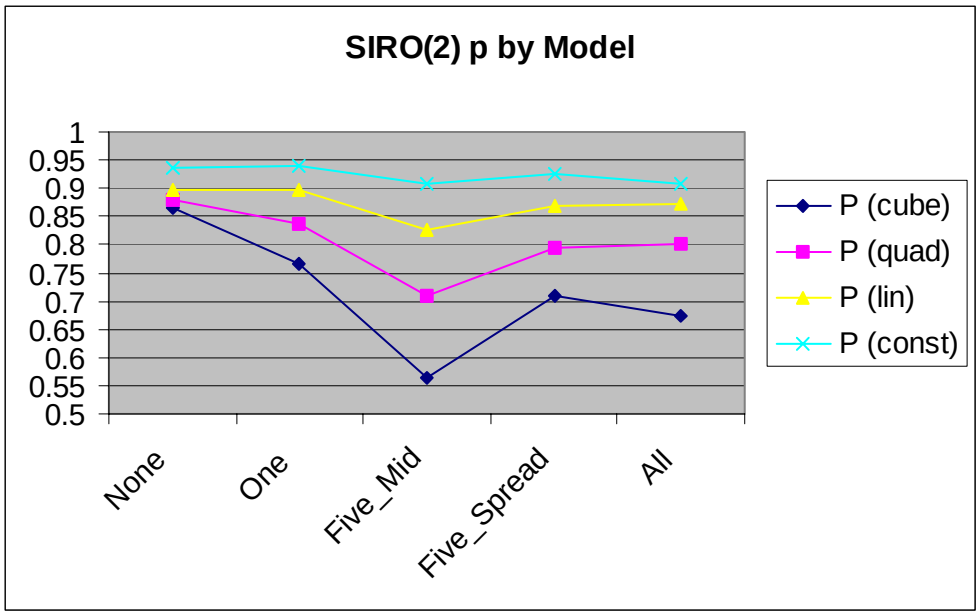

Figure 32: SIRO (Method 2) p-fits

One phenomenon that can be explored in conjunction with those previously tested and likely one of the most common issues dealt with in all of manufacturing is machine failure. When machines fail and need to be repaired, production is brought to a halt and parts can stockpile in idle queues. If this machine failure and/or maintenance occurs often in a system, not modeling some kind of mean time until machine failure and mean repair time can lead to representations of the CT/TH curve that are drastically different than the real behavior of the system and can have costly results. We modeled one scenario where failures occurred periodically and randomly at each of the 20 processes in the series. The time between failures was modeled as exponentially distributed with a mean of 10 minutes. Once down, the time to bring a process back up was exponentially distributed with a mean of 1 minute. In our Arena model, the arrival rates were adjusted by a factor so that the introduction of failures won't push the system beyond capacity. The formula for stability in a system with failures is:

$$\rho \text{ (stability condition)} \rightarrow \lambda/\mu^{*}(1 + \delta/\gamma) < 1$$

In the formula, δ/γ is the ratio of the "arrival rate" of failures to the "repair rate," or 1/10 in our case. Hence, to achieve our desired throughputs in the failure models, our arrival rates were multiplied by 1 + 1/10 or 11/10. For example, a throughput of 0.98 in the model with failures would be a throughput of 0.89 with no failures taken into account.

TH	0.70	0.76	0.82	0.88	0.92	0.95	0.96	0.97	0.98
Failures	16.94	22.23	28.45	41.58	61.41	90.01	106.44	129.98	163.68

Figure 33: Failures raw data

The mean data shown in Figure 33 was generated from "full" runs of our model where we ran the number of replications calculated for our 20 M/M/1 queues such that the relative error at each point in the mean curve would be sufficiently small. Even when we add this seemingly small percentage of failure time to each process, it is evident that the effect on the mean cycle time is large. In terms of the value of the parameter *p*,

introducing failures into the system proves to show a large contrast in *p* depending on the model to which the data is fit. This contrast is markedly large compared to results we experience in the other experiments (*p* ranging from near 0.9 down to below 0.5), but R^2 values still show that all the fits are good. Figure 34 below shows a plot of the different *p* values for each model and Figure 35 shows the R^2 values. All fits show R^2 values over 99%; 80% is generally considered a good fit. In this case, there is no way to further simplify the numerator in the model in order to achieve *p* values closer to one.

Interestingly enough, in the models currently used by our research counterparts in Arizona, machine failure is one of the most common features and also happens to be the only real stochastic factor in those models. This makes it all the more encouraging that even in the mean curve we find instances where *p* cannot be one.

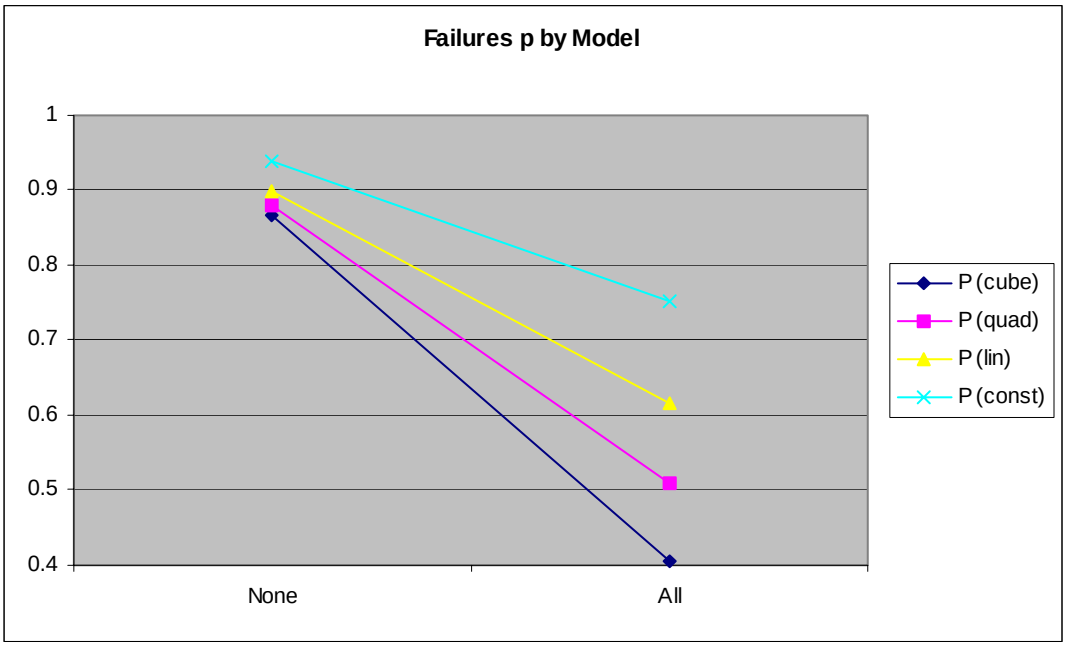

Figure 34: Failures p-fits

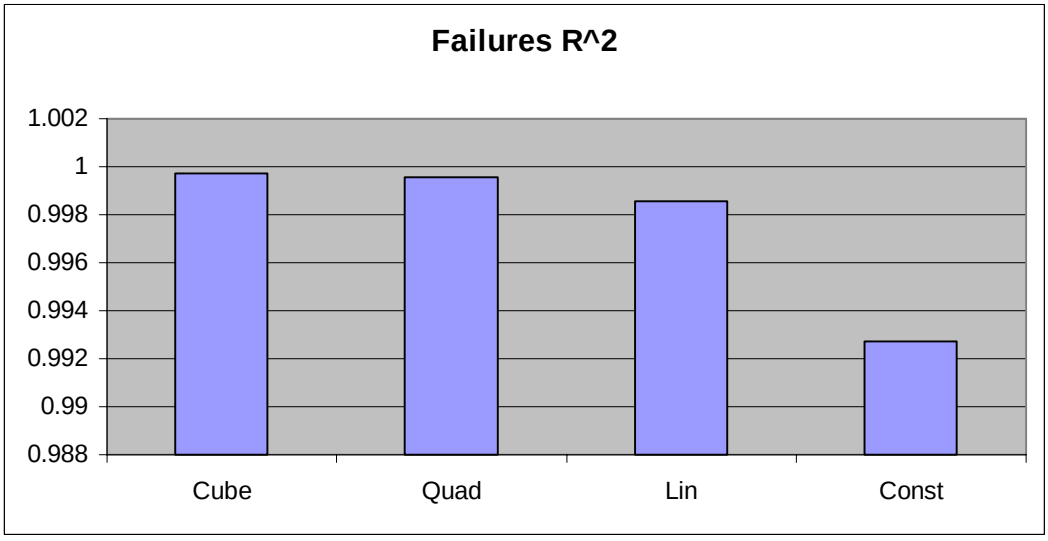

Figure 35: Failures R^2 values

The fits of the 1st moment variance, 2nd moment, and 2nd moment variance curves for the model incorporating failure show drastically different exponent fits than what is predicted for an M/M/1 queue without stochastic resource failure. While the value of *q* in the 1st moment variance model is about half of what is predicted (slightly greater than two), the raw and centered 2nd moments' *p2* values range from half to about 75% of their predicted value of two. The centered 2nd moment's fits are consistent no matter the order of the model's numerator while the raw second moment's exponent value is inversely correlated with the level of complexity in the numerator. This is also the case for the 2nd moment variance *q2* fits, none of which come close to six, as is the case for a standard M/M/1.

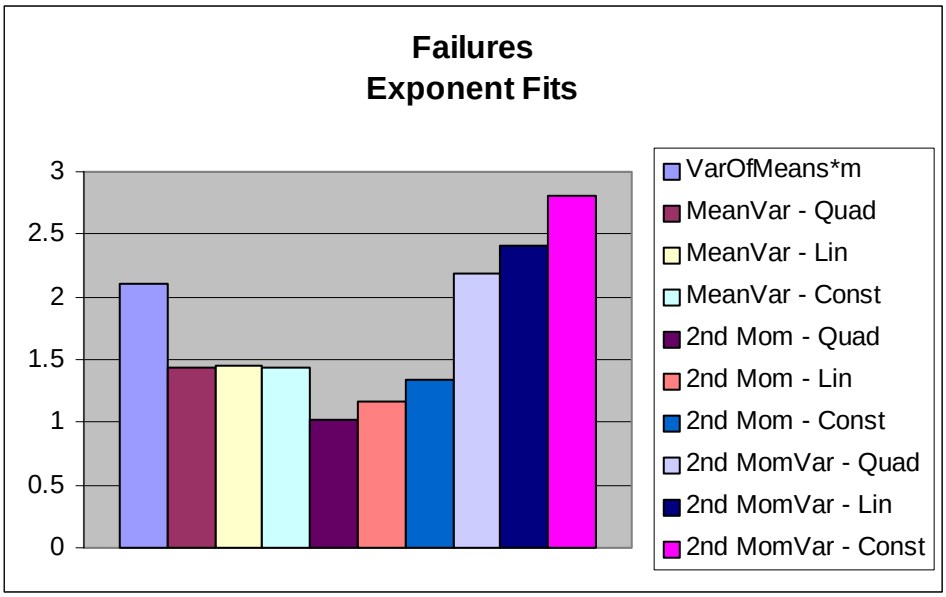

Figure 36: Higher moment exponent fits for a single M/M/1 case.

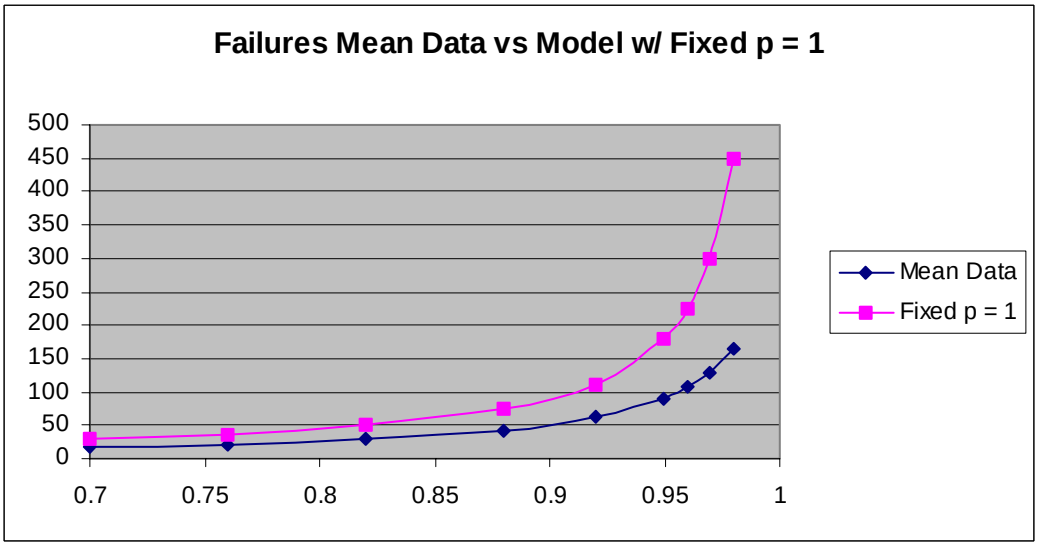

Figure 37: Plot of Failures mean data vs. a model with *p* fixed at 1

To illustrate the drastic effect changing the exponent can have on the CT/TH curve, Figure 37 shows a plot of the mean failure data along with a model with the same

numerator value as was fit to the experimental data, but with p constrained to one. The steepness of the knee is evidently curtailed when p is allowed to be less than one. If using a curve with a fixed p to estimate cycle time, one would surely overestimate the time needed to produce a certain number of units (or wafers) and understate their true production capabilities. It is harder to see such drastic differences in the plots of the higher moment curves, but the differences there lie in the precision to which one believes the cycle time can be predicted. If confidence intervals were drawn over each point in the above plot, smaller exponents at the high moments would translate to smaller intervals and thus more confidence in the estimator.

Conclusions

For a manufacturer of silicon wafers, or a manufacturer in any field, accurate estimation of the cycle time – throughput curve is critical to avoid costly errors in production. Examples are overproduction when the true cycle time is low leading to excess inventory or waste, or underproduction when the actual cycle time is greater than predicted. Costly errors in estimation are often overlooked simply because the tools and resources are not there to generate more accurate models using present methods. Our research strove to find models that robustly fit any manufacturing scenario and reduced instances of error. The simulation models we built do not come close to the vast array of topological possibilities that exist in the world of silicon wafer fabrication; they merely touch the surface of the methods and associated permutations that are used. With only a small sampling of topologies and other stochastic factors in our simulation models, however, we were able to find instances where the accepted theoretical models of Whitt and Cheng and Kleijnen fall short.

The first set of simulations we tried did not show exactly what we wanted in terms of varying values of p nor in the values of the exponents in the higher moment models. The factors we looked at were re-entrant flow and a number of queues in series. Specifically, we hoped that the re-entrant flow aspect would cause a significant adjustment in p. The aspect of queues in series was thought to maybe have a buffering effect on the traffic induced by the re-entrant flow. We found that a model with a linear throughput term in the numerator could more than adequately fit any combination of re-entrant flow and series topologies with an exponent p fixed at one. After running a small number of models using batching and bottleneck queues at different locations in the network, we came to a similar conclusion; p was not affected. Despite these discouraging realizations, we were able to infer that other such common nuances in topology would likely reveal the same and were thus not worth further exploration.

Some of the most common phenomena found in wafer fabrication networks and in the models currently used to simulate them are queueing prioritization and machine failure. We simulated prioritization situations via variations in queueing disciplines in our queues in series. We tried LIFO, SPTF, as well as SIRO (service in random order) as variations from FIFO, which we used in all other simulations. These pretty well covered the spectrum from simple prioritization based on arrivals or processing time, to the most complicated possible prioritization scheme: random selection from the queue. Our LIFO and SIRO results showed an interesting dependency on the value of p depending on the order of the numerator in our formula. We found cases (depending on how many of the

queues in the series utilized the experimental discipline) where p varied proportionally and inversely with the order of the numerator. Even so, neither LIFO nor SIRO definitively showed any consistent variation in p. For both, there was at least one model where $p = 1$ was a good fit. It also happened to be that it was possible in these cases to keep a small order numerator. This was a surprising result for LIFO, which has been shown to have values of p higher than one in the instance of a single M/M/1 queue. Only when fit to the cubic model was the LIFO mean data consistent with previous findings. We knew from previous work that SPTF has the opposite effect on p for a single M/M/1 queue; it tends to fit values of p significantly less than one. We confirmed this fact with our simulation model of 20 queues in series utilizing SPTF. Furthermore, the SPTF model showed almost no dependency on the degree of the function in the numerator.

Arguably, our most important finding comes from the model exhibiting machine failure. Surely, failures are unavoidable and will occur in all manufacturing systems no matter what the layout. A stochastic model of machine failures is as near as one can get to the unpredictable nature of real life. Even measures like preventative maintenance might help alleviate randomness in failure occurrence, but it still causes machine downtime, which can wreak havoc on a seemingly well-balanced queueing system at high throughputs. Our mean CT/TH curve generated by the failures model fit to vastly different values of p (all less than one) that were, as in previous cases, strongly dependent on the order of the functional model's numerator. Here, however, there were no cases where fixing p close to one would be a good fit. The model with a constant numerator came closest ($p = 0.75$), and adding more throughput terms to the numerator of the model (as suggested by Cheng and Kleijnen) only decreased p further. There is no way to fit a model with this type of machine failure without adjusting p. The 1^{st} moment variance, 2^{nd} moment, and 2^{nd} moment variance fits to the failure model data showed similarly convincing results. Traditional theoretical models for a single M/M/1 queue cannot account for the higher moment behavior exhibited by our network.

There likely exist more network topologies and manufacturing phenomena that exhibit just the type of behavior we sought to show here. Only by continuing to try more can we come up with the most generalized formula possible that remains simple enough to be practical. It was this team's objective to minimize experimental effort and unnecessary loss of revenue for the wafer fabrication community as well as the manufacturing world as a whole. Through this small sampling of scenarios, we show that better models do indeed exist and finding them is possible via low-cost simulation and minimal experimental effort.

Appendix A

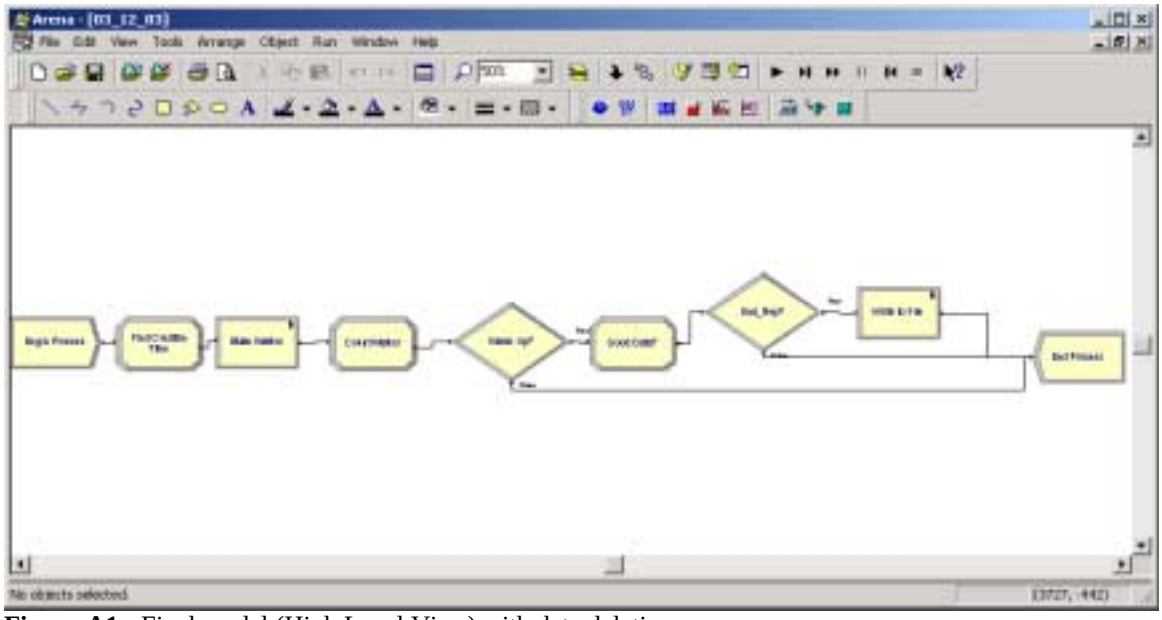

Figure A1: Final model (High Level View) with data deletion

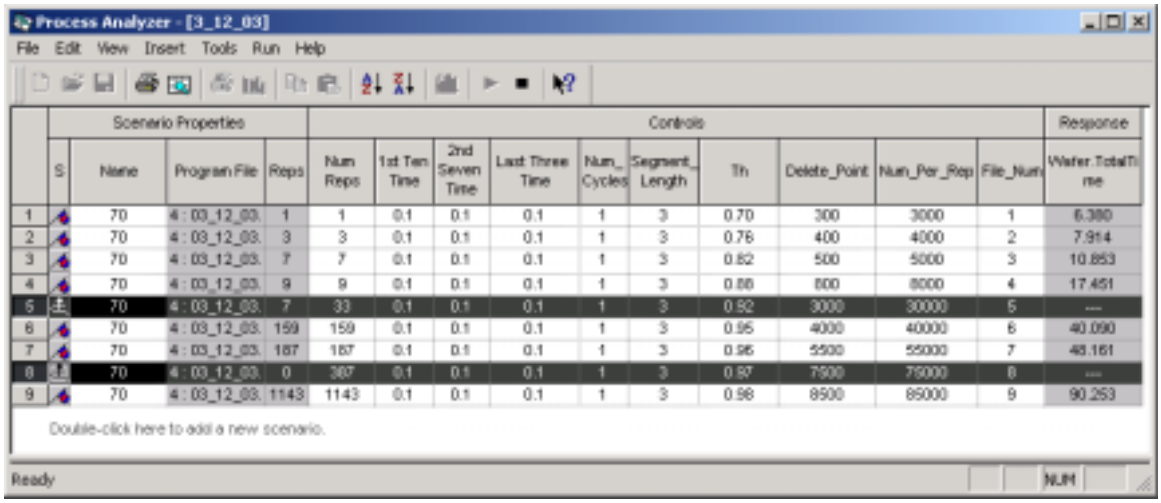

Figure A2: PAN setup for re-entrant flow model

Supplemental Data

Throughput	0.7			0.76		
R0 Mean		6.7189391			8.1910763	
R0 Standard Deviation		0.292162791			0.64925465	
% Relative Error	0.1	0.05	0.01	0.1	0.05	0.01
R0 Epsilon	0.671894	0.335947	0.067189	0.8191076	0.409554	0.081911
Number of Reps (Min)	1	4	97	4	13	321
R1 Mean		6.378985			7.913759	
R1 Std Dev.		0			0.579683	
Number of Reps (Exp)			1			3
h.l.			0.20603			0.457847
95% CI LB			6.512909			7.73323
95% CI UB			6.924969			8.648923

0.82			0.88			0.92		
10.9925381			17.6572315			24.1713063		
1.001890599			1.14683403			2.317766977		
0.1	0.05	0.01	0.1	0.05	0.01	0.1	0.05	0.01
1.0992545	0.5496271	0.1099254	1.7657233	0.8828629	0.1765722	2.4171315	1.2085651	0.2417134
	10.85281			17.45046			25.02591	
	1.142187			0.999312			2.484577	
		7			8			33
		0.706521			0.808734			0.880156
		10.28602			16.8485			24.14575
		11.69906			18.46597			25.90607

0.95			0.96			0.97		
41.4817842			44.920901			65.1083629		
5.060012524			5.493600527			8.008385938		
0.1	0.05	0.01	0.1	0.05	0.01	0.1	0.05	0.01
4.1481788	2.0740893	0.4148176	4.492098	2.2460453	0.4492097	6.5108368	3.2554183	0.6510847
	40.11617			48.21532			64.32671	
	5.063782			5.10013			8.255812	
		155			182			387
		0.805331			0.748533			0.83094
		39.31084			47.46679			63.49577
		40.9215			48.96385			65.15765

0.98		
90.8398878		
13.58713299		
0.1	0.05	0.01
9.0839891	4.5419946	**0.9083989**
12	46	1113
	90.16123	
	13.2081	
		1113
		0.7799359
		89.381297
		90.941168

Figure A3: Final Replication Determination

Model	k	seg	queues	p	c0	c1	c2	R^2
1	3	1	3	0.982673	0.196066	0.577393	-0.454297	0.999667
1	5	1	3	0.944810	1.609440	-2.503250	1.527160	0.999745
1	3	3	3	0.997261	-0.591007	3.565990	-2.088210	0.999777
1	5	3	3	0.983012	2.819820	-3.123200	1.912740	0.999883
1	3	1	10	1.060690	0.419709	1.283660	-1.477980	0.999908
1	5	1	10	1.042790	0.734792	0.911653	-1.234970	0.999915
1	3	3	10	0.998455	0.970247	0.948553	-1.017690	0.999974
1	5	3	10	0.955875	5.299700	-8.231040	4.749080	0.999961
1	3	1	20	1.179300	1.806160	0.673020	-1.010090	0.999888
1	5	1	20	1.113160	1.460980	0.530137	-1.694810	0.999913
1	3	3	20	0.925523	4.326460	-5.210040	2.118440	0.999916
1	5	3	20	0.975076	3.294660	-1.668310	0.020454	0.999945
2	3	1	3	0.967076	0.504422	0.161744	0.000000	0.99966
2	5	1	3	0.971543	0.577819	-0.016536	0.000000	0.999722
2	3	3	3	0.972476	0.809498	0.182893	0.000000	0.999762
2	5	3	3	0.995747	1.52408	-0.005024	0.000000	0.999878
2	3	1	10	0.97643	1.50046	-1.16536	0.000000	0.999788
2	5	1	10	1.00702	1.60519	-1.1195	0.000000	0.99989
2	3	3	10	0.985962	1.66537	-0.711336	0.000000	0.99997
2	5	3	10	0.985099	2.07458	-0.482151	0.000000	0.999934
2	3	1	20	0.977417	2.93599	-2.59967	0.000000	0.999536
2	5	1	20	1.03111	2.78723	-2.34845	0.000000	0.99982
2	3	3	20	0.946443	2.86704	-1.74459	0.000000	0.999902
2	5	3	20	0.975218	3.28063	-1.63488	0.000000	0.999945
3	3	1	3	0.942171	0.380245	0.000000	0.000000	0.999571
3	5	1	3	0.969974	0.564976	0.000000	0.000000	0.999721
3	3	3	3	0.982402	0.952109	0.000000	0.000000	0.999748
3	5	3	3	0.995573	1.52016	0.000000	0.000000	0.999878
3	3	1	10	0.812517	0.667232	0.000000	0.000000	0.996231
3	5	1	10	0.896176	0.774332	0.000000	0.000000	0.998349
3	3	3	10	0.947549	1.12054	0.000000	0.000000	0.999769
3	5	3	10	0.969298	1.70134	0.000000	0.000000	0.999899
3	3	1	20	0.664548	1.2687	0.000000	0.000000	0.987563
3	5	1	20	0.803138	1.15498	0.000000	0.000000	0.994002
3	3	3	20	0.86654	1.56692	0.000000	0.000000	0.998927
3	5	3	20	0.924118	2.03811	0.000000	0.000000	0.999578

Figure A4: Final Q, Seg, and K data, p-fits, and R^2 calculations

Appendix B

Remaining Factorial Analysis of Q, Seg, and K's Effect on p

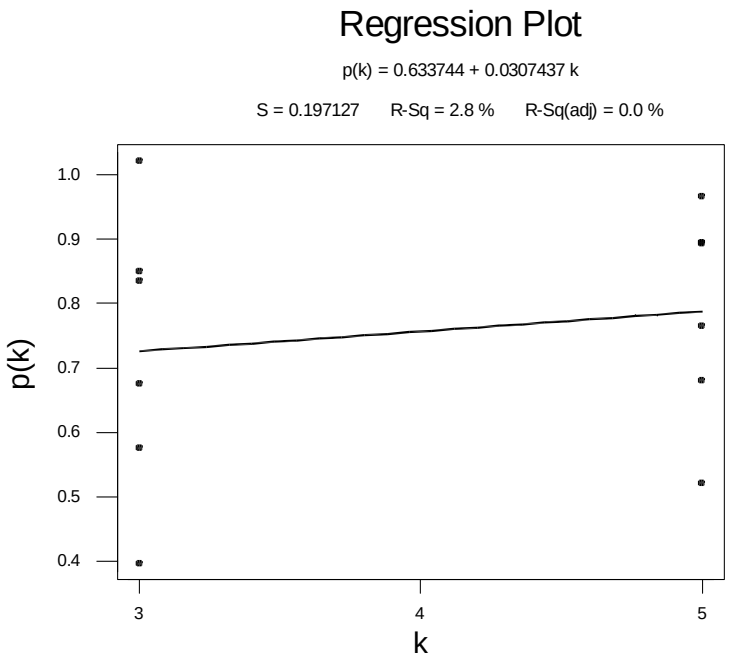

Figure B1: Main Effect of K on p (Trial Data)

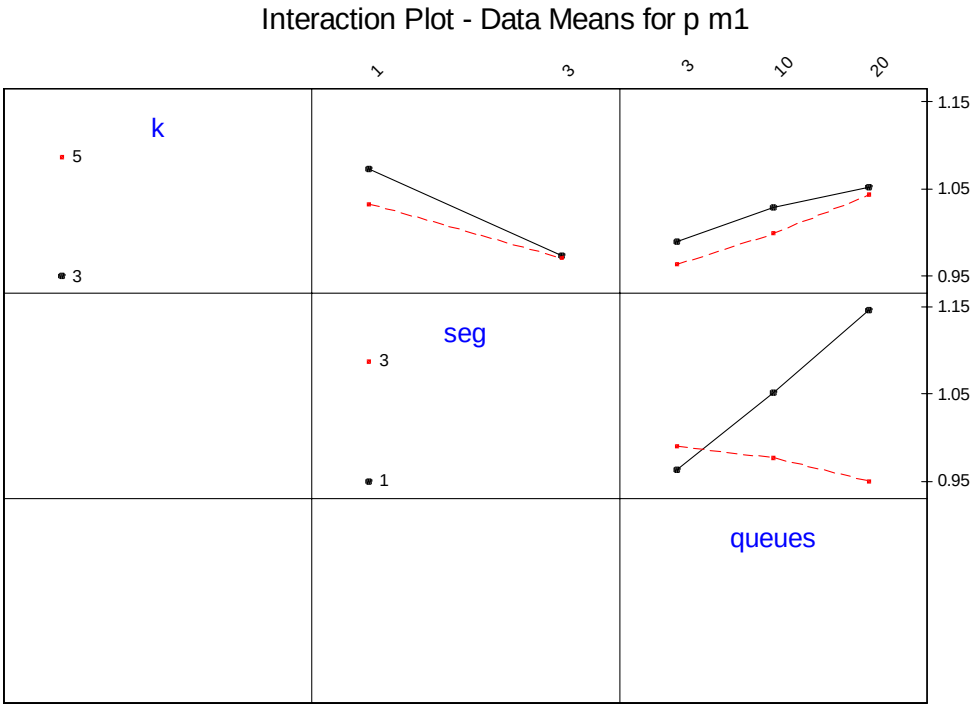

Figure B2: Model 1 Interactions

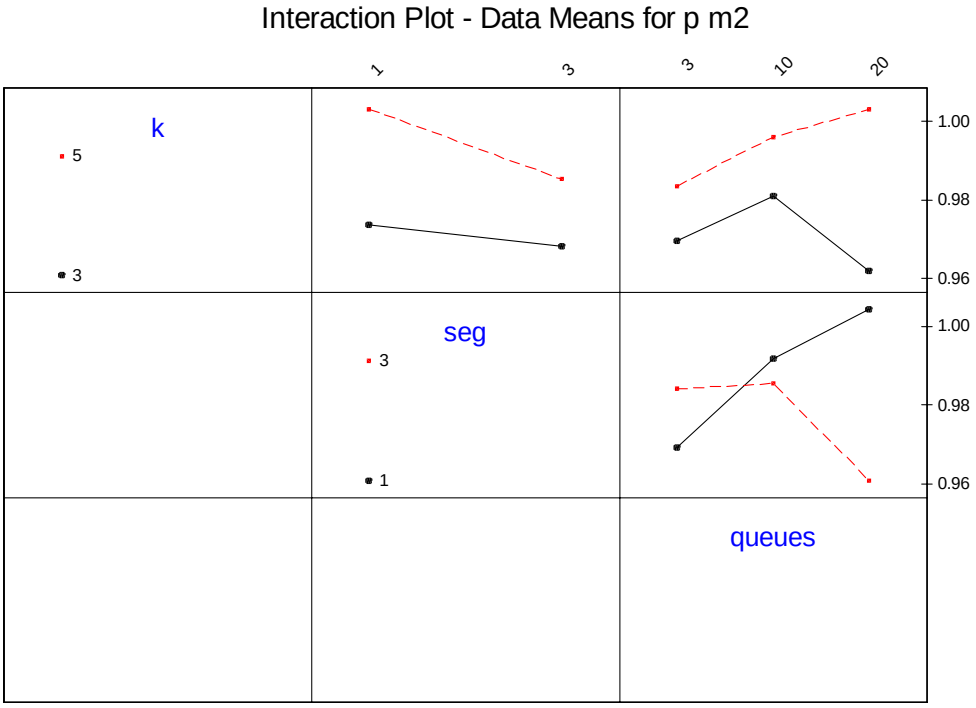

Figure B3: Model 2 Interactions

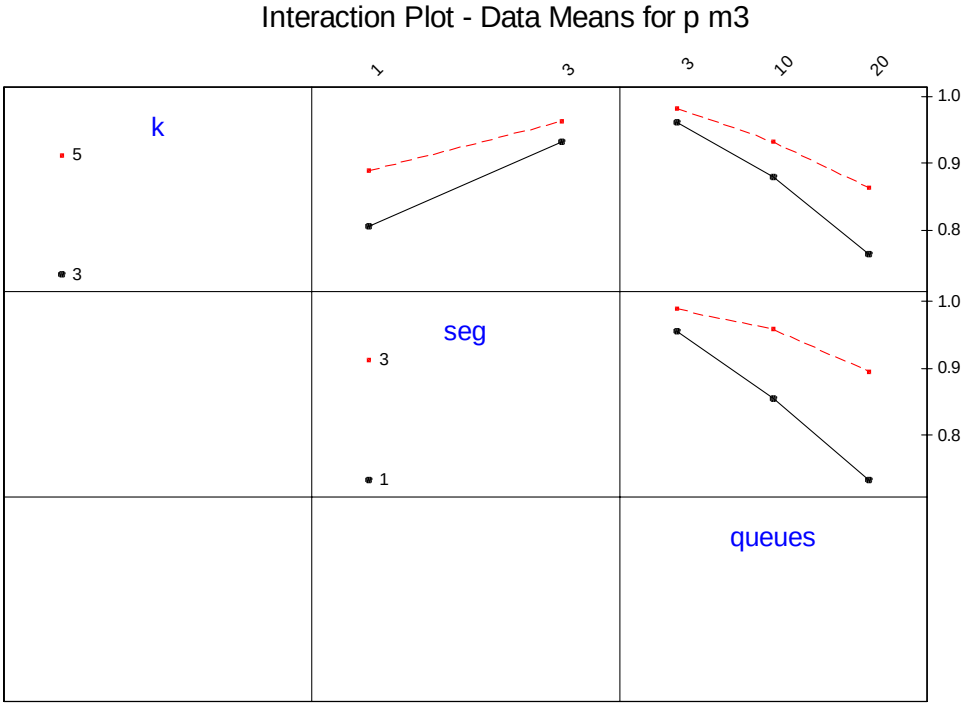

Figure B4: Model 3 Interactions

Appendix C

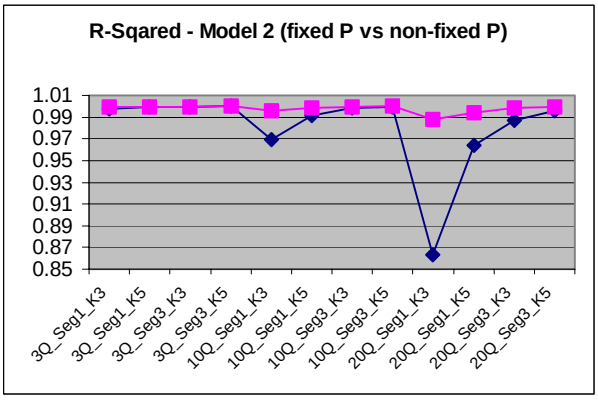

Figure C1: Justification for not fixing p at one. (Able to achieve better fits by adjusting p)

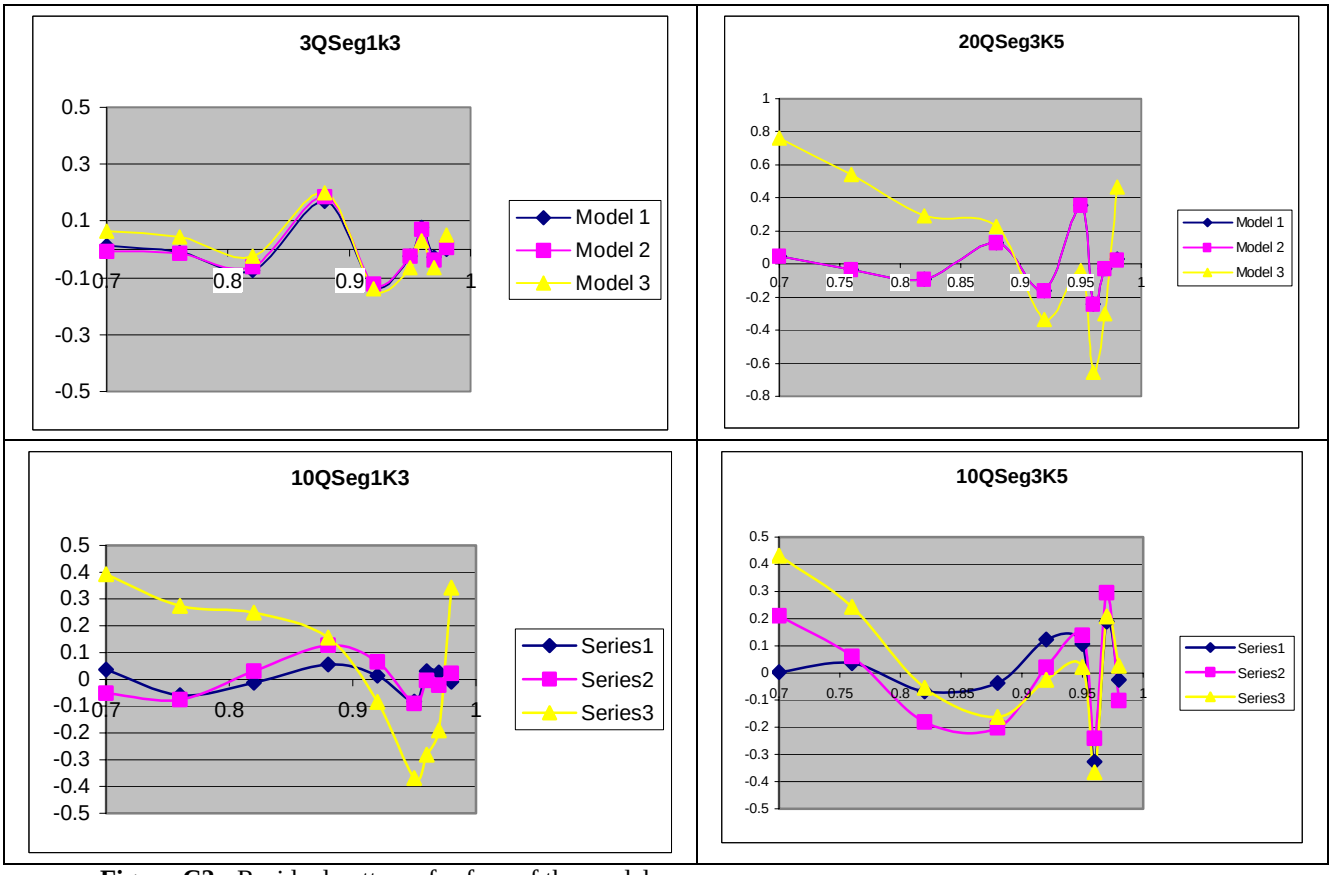

Figure C2: Residual patterns for four of the models.

Appendix D

Batch/Bottleneck Experiment

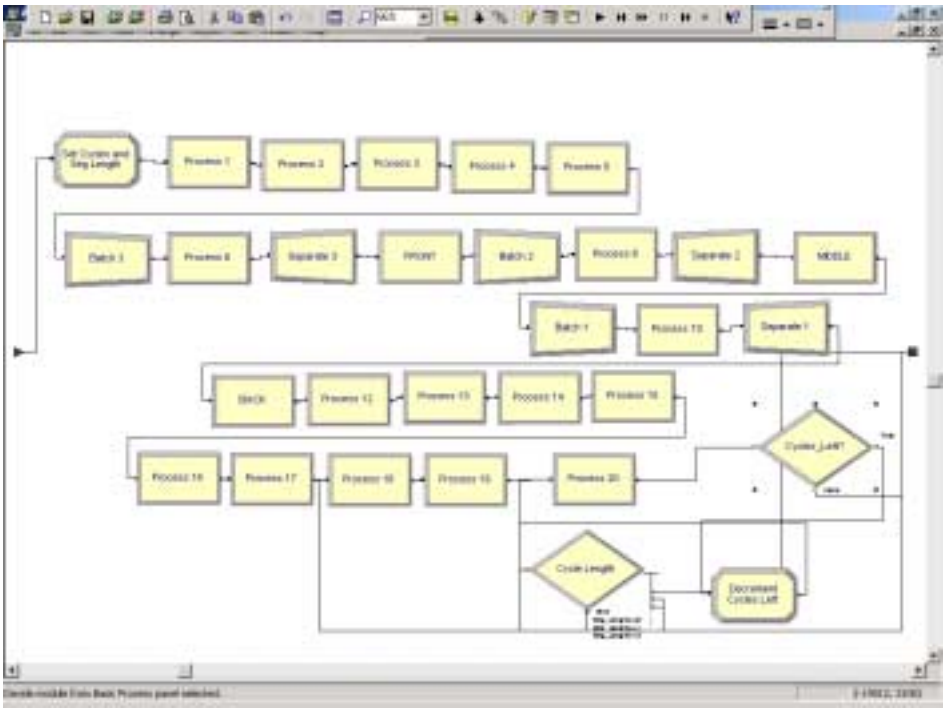

Figure D1: Batch/Bottlenect submodel.

Buffer Queues? (Y/N)	# of Batch Processes	Batch Size
Y	1	10
N	1	10
Y	2	10
N	2	10
Y	3	10
N	3	10
Y	1	50
N	1	50
Y	2	50
N	2	50
Y	3	50
N	3	50
Y	1	100
N	1	100
Y	2	100
N	2	100
Y	3	100
N	3	100

Figure D2: Batch experimental design.

Bottleneck TH	Bottleneck Location
0.9	Front
0.95	Front
0.98	Front
0.9	Middle
0.95	Middle
0.98	Middle
0.9	Back
0.95	Back
0.98	Back

Figure D3: Bottleneck experimental design.

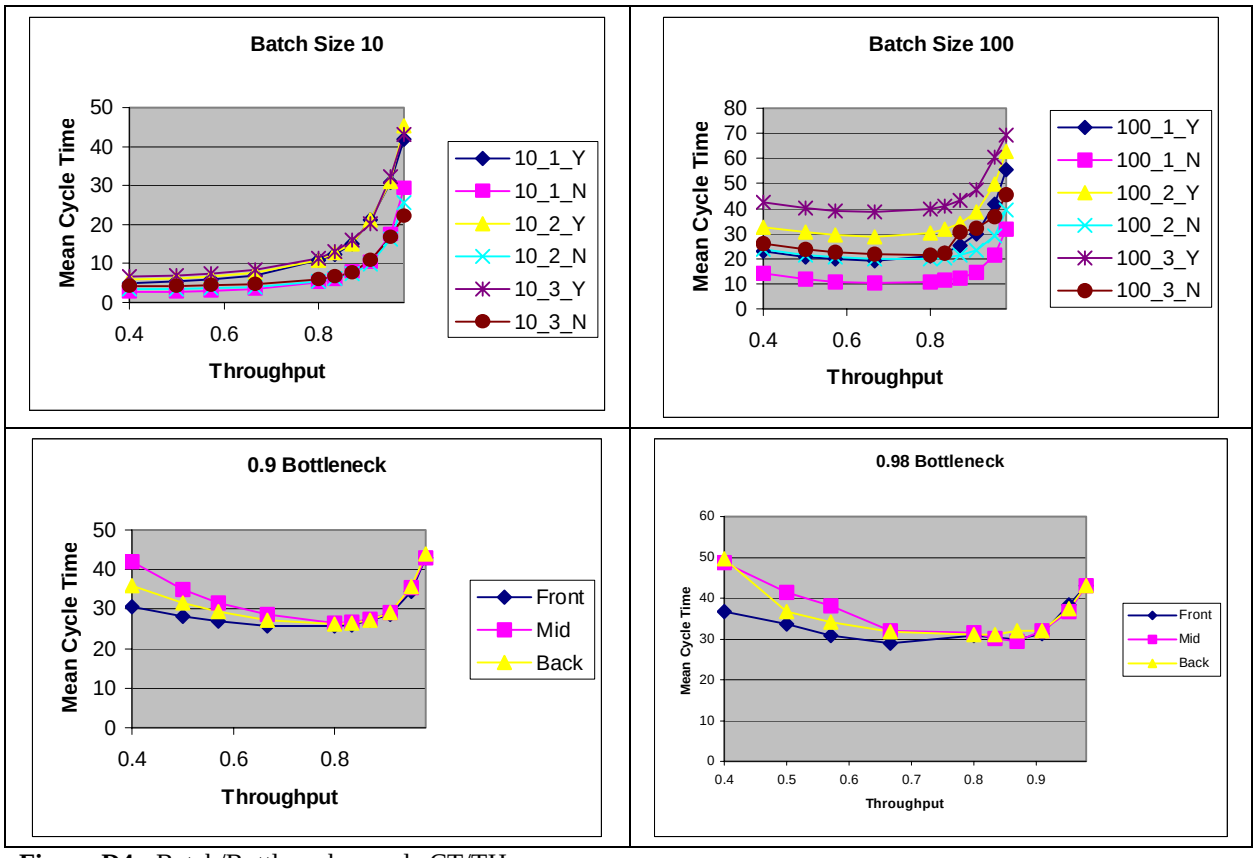

Figure D4: Batch/Bottleneck sample CT/TH curves.

Appendix E

Additional Higher Moment Plots, Data

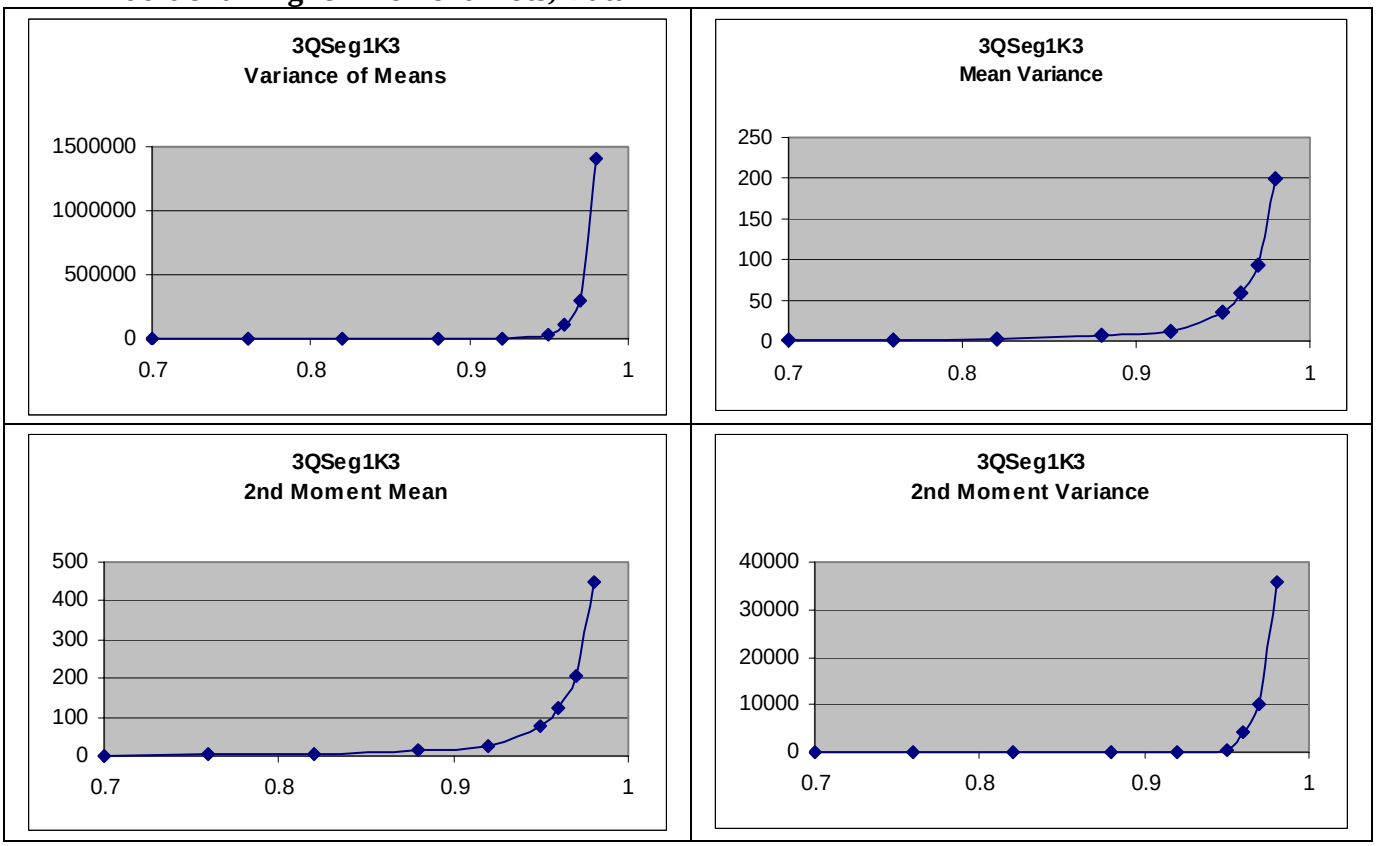

Figure E1: 3Qseg1K3 Higher moment CT/TH curves

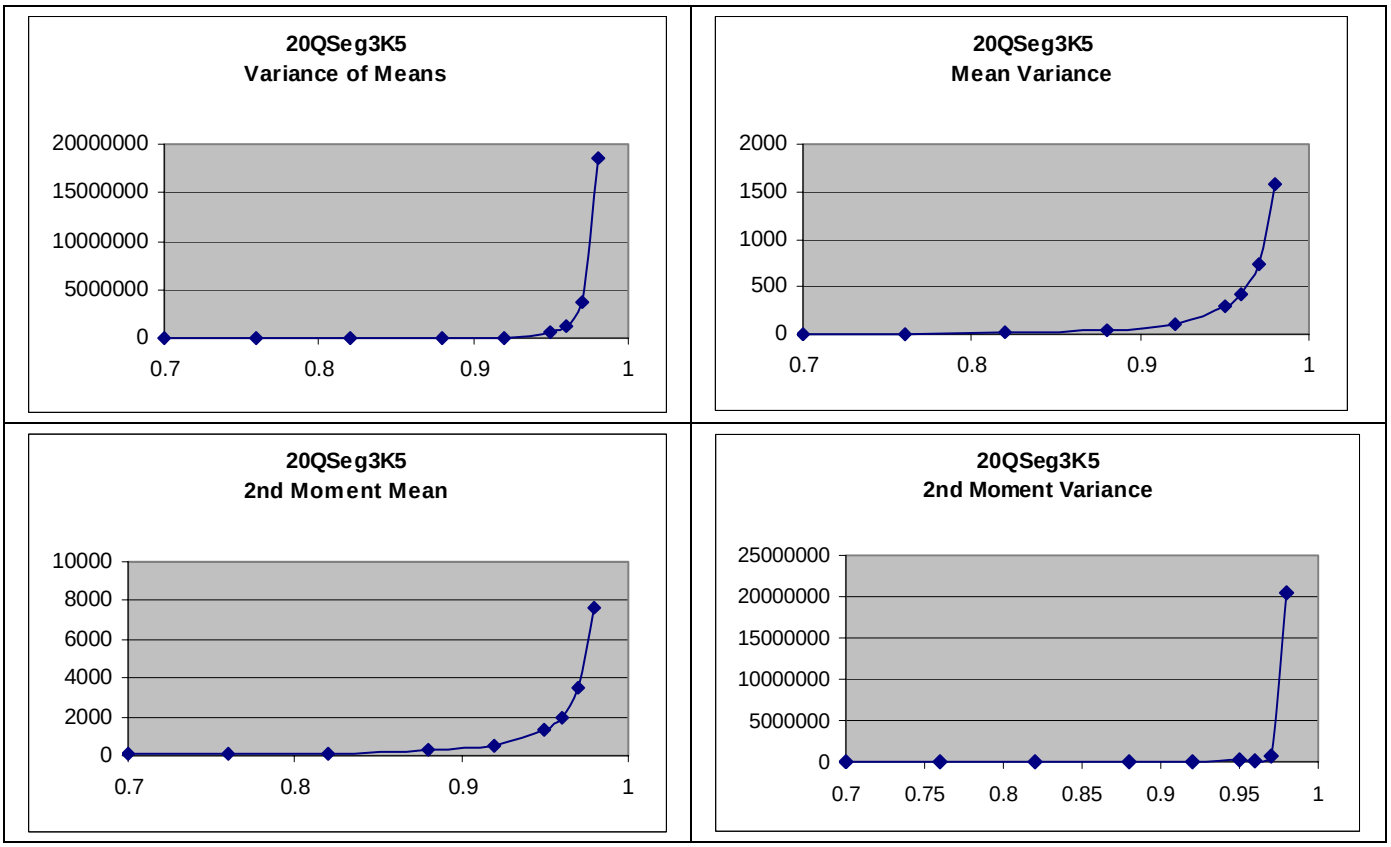

Figure E2: 20Qseg3K5 Higher moment CT/TH curves

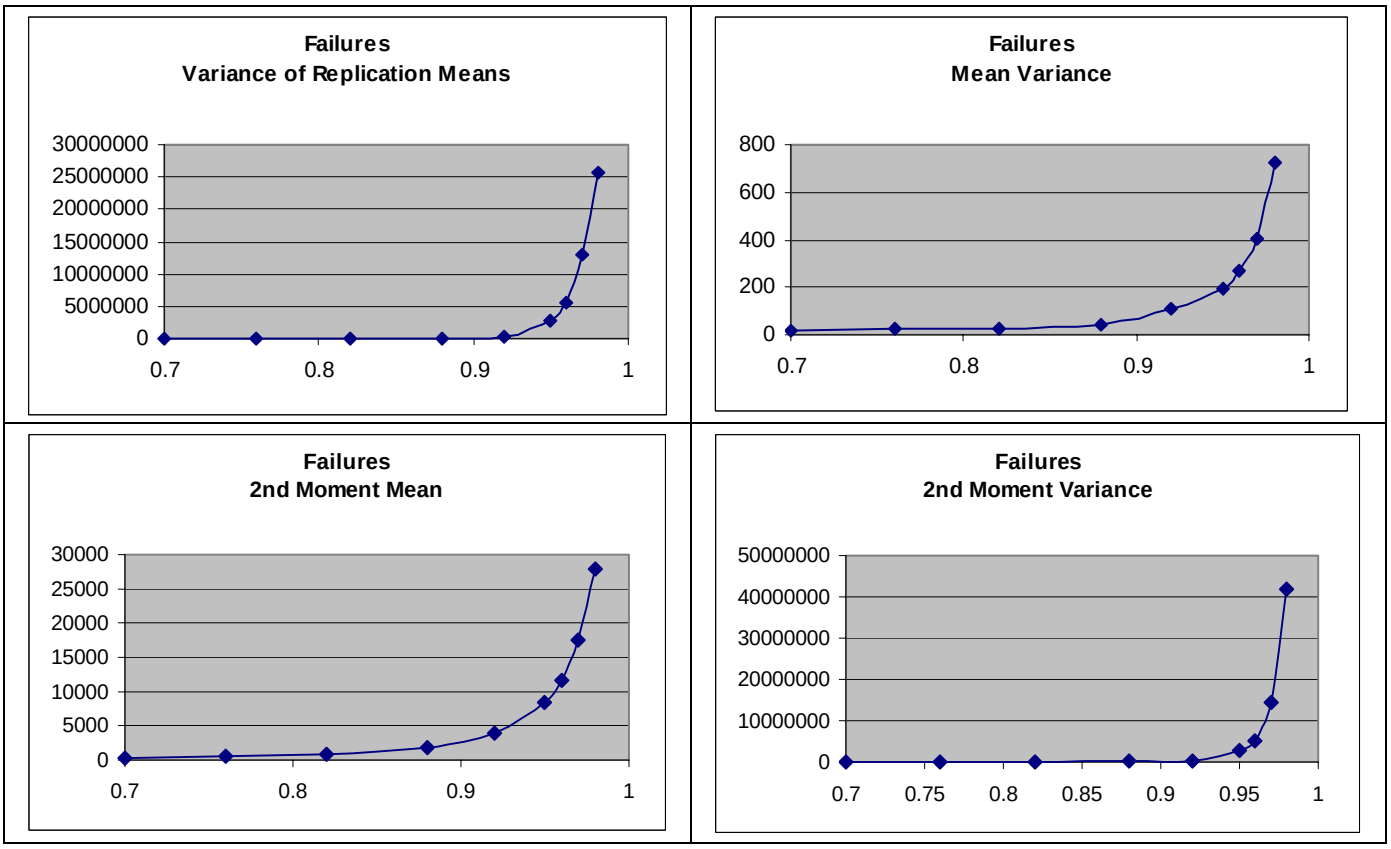

Figure E3: Failures Higher moment CT/TH curves

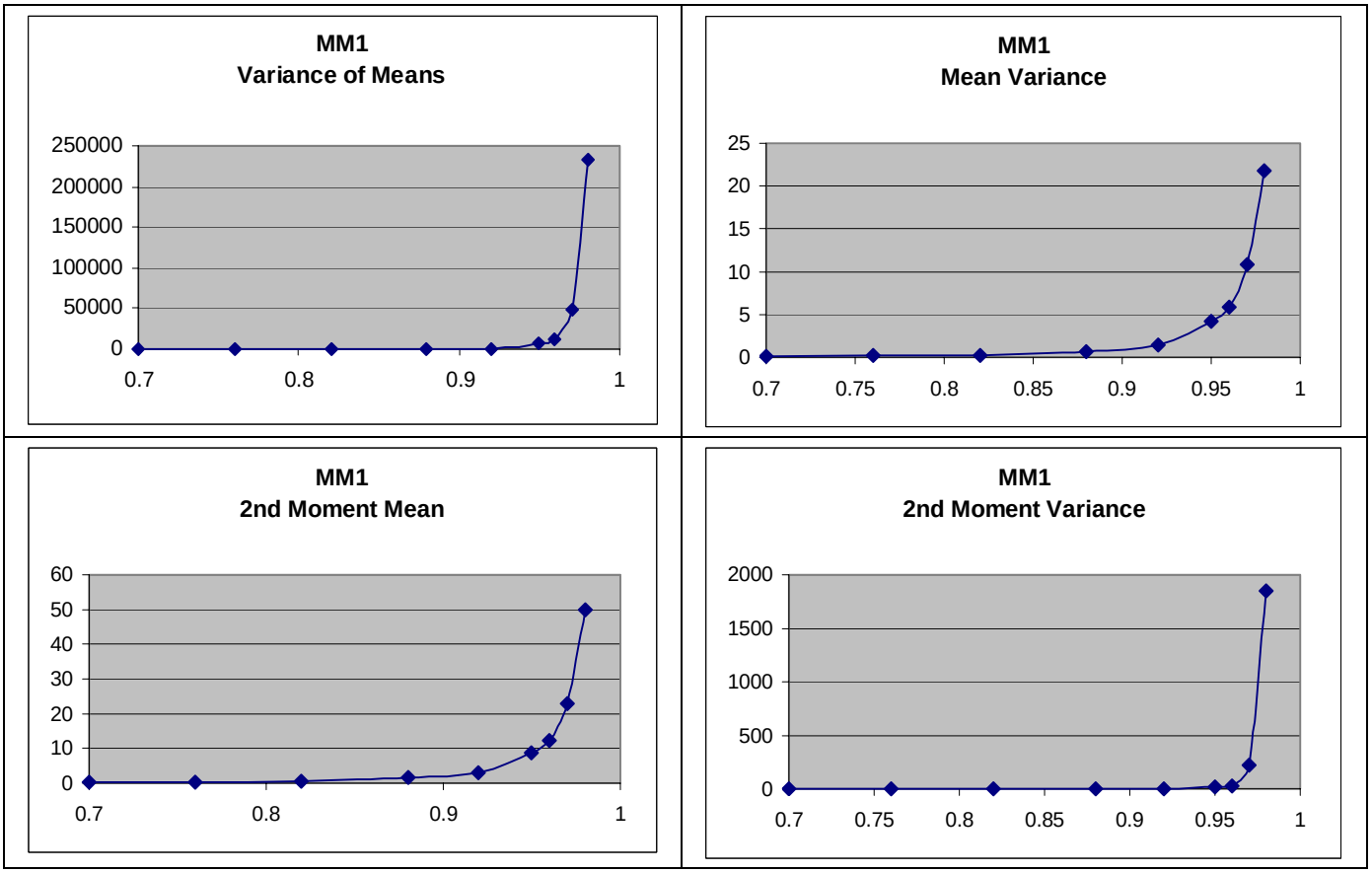

Figure E4: MM1 Higher moment CT/TH curves

		Model		Exponent	A	B	C	R^2
3QSeg1K3	Variance of Means	Const	q	3.81428	0.466684			0.999925
		Quad		1.67943	2.12849	-5.26786	3.449245	0.999366
	Mean Variance	Linear	p2?	1.751078	-0.39177	0.614741		0.999332
		Const		1.851661	0.142415			0.999139
	2nd Moment Mean	Quad		1.89801	0.246157	-0.12078	0.14531	0.999728
		Linear	p2	1.901242	0.136402	0.130199		0.999728
		Const		1.917205	0.248076			0.999725
	2nd Moment Variance	Quad		2.406776	247.4186	-573.458	330.5777	0.998789
		Linear	q2	2.642008	-19.3259	20.90684		0.998592
		Const		3.220357	0.121186			0.997772
20QSeg3K5	Variance of Means	Const	q	3.989933	3.094546			0.999927
		Quad		1.744132	9.975301	-24.6122	16.5132	0.999917
	Mean Variance	Linear	p2?	1.798769	-2.17471	3.632629		0.999902
		Const		1.887591	0.980161			0.999773
	2nd Moment Mean	Quad		2.793876	5.577347	-5.44876	-0.10512	0.999954
		Linear	p2	2.795825	5.64937	-5.62633		0.999954
		Const		1.919834	4.168475			0.999918
	2nd Moment Variance	Quad						
		Linear	q2					
		Const		8.120907	3.26E-07			0.99977
Failures	Variance of Means	Const	q	2.101601	7024.175			0.990037
		Quad		1.438089	6.575685	-8.28429	4.315811	0.998995
	Mean Variance	Linear	p2?	1.45214	3.478392	-1.03577		0.998994
		Const		1.43597	2.622434			0.998983
	2nd Moment Mean	Quad		1.017968	5971.699	-15253.1	9889.732	0.999533
		Linear	p2	1.172013	-748.744	1056.986		0.998558
		Const		1.342825	148.6929			0.99546
	2nd Moment Variance	Quad		2.183202	500854.7	-1171162	682104.8	0.999382
		Linear	q2	2.407155	-39774	44076.53		9.99E-01
		Const		2.814993	694.7619			9.98E-01
MM1	Variance of Means	Const	q	3.936789	0.047628			0.999724
		Quad						1
	Mean Variance	Linear	p2?	1.725571	-0.05476	0.081986		0.999352
		Const		1.836662	0.016604			0.999084
	2nd Moment Mean	Quad		1.894545	0.102342	-0.2144	0.143622	0.999688
		Linear	p2	1.920898	-0.00624	0.034136		0.999686
		Const		1.961444	0.023234			0.999668
	2nd Moment Variance	Quad		4.108366	0.092658	-0.19715	0.1049	0.999935
		Linear	q2	4.605739	-0.00064	0.000679		0.999921
		Const		5.3372	1.59E-06			0.999879

Figure E5: Higher Moment Model fit data

References

Cheng and Kleijnen, 1999. "*Improved Design of Queueing Simulation Experiments with Highly Heteroscedastic Responses," Operations Research* Vol. 47, No. 5, 762-777

L. Kleinrock. 1976. Queueing Systems, Vol. II: Computer Applications, Wiley, NY.

Nelson, Banks, Carson, and Nicol, 2001. *Discrete-Event System Simulation,* Third Edition, Upper Saddle River, New Jersey, 398-441

Whitt, W., 1989 "Planning Queueing Simulation," *Management Science* **35,** 1341-1366