Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

SYSTEM CENTRIC MISSION CRITICAL SIMULATION MODEL FOR MES AUTOMATION

Amit Jindal

Software and Solutions Group
Intel Corporation
Chandler, AZ 85226, U.S.A.

Rajkumar Khandelwal

Component Automation Systems
Intel Corporation
Chandler, AZ 85226, U.S.A.

ABSTRACT

This paper describes the system centric simulation methodology used for stress testing of Manufacturing Execution System (MES) in Intel. System centric simulation involves testing such that the system components (infrastructure stack and software) are characterized for the load they would experience in production, irrespective of how that load is exerted. A new manufacturing execution system software is introduced in Intel's latest fabrication facility. Validation of the product under stress is vital to ensuring that the mission critical capability will be able to comply with Intel's reliability, availability, performance, and scalability needs. The System centric simulation model allows for accurate reproduction of real-world scenarios while not requiring the expensive setup and execution of the complete set of defined use cases.

1 INTRODUCTION

This paper illustrates a methodology used for stress validation of a new software product before initiating production use. The methodology described here accomplishes realistic system centric simulation of the factory scenarios.

The introduction of a mission critical software in a manufacturing facility requires extensive validation prior to exposing the users to its capabilities. A manufacturing facility has several different needs from any software deployed for its use: functional needs, reliability of the software, availability of the software, ability of the software to perform at required speed in order to meet user expectations, and ability of the software to scale as production ramps volume to meet the customers' demands. Functional needs are validated by lab testing with the right interfaces by simply exercising all the use-cases at normal and boundary conditions.

The validation of reliability, availability, performance, and scalability needs, however, are orders of magnitude more difficult to achieve. Reliability is defined as the inverse of the "number of times" a software fails over a defined time span. Availability is defined as the inverse of the "duration" of software failure over a defined time span. Performance needs for the software are defined by the users as the required throughput rate, typically measured by average amount of time taken by the software for each transaction category. Scalability needs for the software are defined as its ability to maintain its reliability, availability, and performance metrics within the required range, as usage increases to meet the peak volume demands.

Ability of the mission critical software to meet the reliability, availability, performance, and scalability needs, is a necessary condition for it to be successful in a manufacturing facility. Poor reliability of the software impacts the productivity and efficiency of the factory and its users. It is also not possible to accurately plan the factory outs, if its mission critical software keeps failing often and at an unpredictable rate. Long downtimes of the mission critical software can result in poor availability of the factory systems, thereby resulting in reduced output of the factory. Software not meeting the performance requirements can result in frustration amongst the users, not to mention the reduced productivity resulting in sub-optimal output of the factory. Lastly, if the mission critical software is not able to scale to the volume demands of the factory, it limits the factory's ability to deliver to the market demand, which can result in serious repercussions because for the corporation. Hence, it is imperative that any mission critical software for a manufacturing facility be subject to rigorous stress validation to ensure it meets not just the functional needs of the users, but also the system needs of the factory.

The de-facto approach for stress testing of a misson critical software would be to setup a lab with a sample number of users, setup the extensive end-user interactions with all the required systems interfacing with each other, exercise all the use-cases expected with the appropriate cardinality and frequency, and thereby simulate Production as would occur on the shop floor of the factory. This method, however, is far too expensive with respect to number of users required to do the simulation, and the number of systems required to mimic a true production facility. It is also error-prone owing to the enormous setup required, the significant lock-step interaction of all the users or use cases to achieve

synchronized testing, and the appropriate randomization of the test scenarios as determined by human interpretation of real-world usage. Some of these limitations could be eliminated by using software to simulate the end-user experience. However, this approach poses unique new issues, such as determining the mechanism for recovery from failure of a particular thread of simulation. More often than ever, the only approach possible is to scrap the entire simulation run in case of failure of any major lane of testing, because of the reliance of other testing threads on the failed one.

The System Centric simulation is a revolutionary mechanism for testing of mission critical software, because it eliminates the dependency on end-user interaction and also provides a fairly easy approach towards recovering from a failure. This methodology involves usage of a simulator to exercise the necessary transactions that result from the use-cases, by directly invoking the system APIs rather than through a user-interface. It thereby eliminates the need for the user-interface in the simulation suite. It employs a unique method for setup of the simulation in terms of the object cardinality and other data collaterals, and the transaction cardinality, with no overlaps between any major threads of the required testing. This ensures that no test run will have to be scrapped because of failure of any single component of the simulation. The approach uses a probabilistic distribution method to mimic the randomness of the timing of transactions which is a characteristic of factory usage. The System Centric simulation methodology delivers realistic simulation of the factory usage without incurring the overheads of the traditional simulation systems, thereby delivering to the factory needs at orders of magnitude of efficiency.

2 SIMULATION SET COMPOSITION

A typical 300mm semiconductor manufacturing fabrication unit (fab) runs on several hundred categories of OLTP transactions. Simulation of these transactions enables proactive identification of systemic issues and validation of resolution to these issues. A purist approach would make it empirical that every simulation model accounts for 100% of systemic external forces (all the OLTP transactions in our case). While these external forces cannot be discounted for a life-threatening scenario like rocket launch or medical surgery; for manufacturing units, a smart cost-effective mechanism can be devised to discover and fix systemic issues by identifying and characterizing the transaction set that impose most significant load on the system. This can be achieved within short time durations and keeping the simulation costs low.

Taking the conventional 80-20 pareto rule as our starting point, we data mined the historical behavior of 300mm and 200 mm manufacturing facilities. Our selection criteria for including a particular OLTP transaction into the simulation model was based on five vectors (Figure 1).

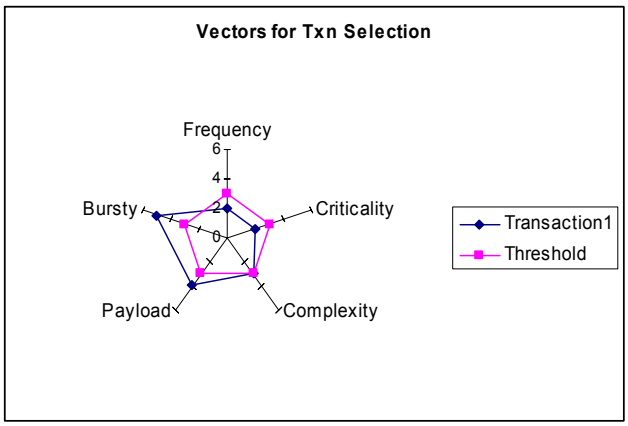

Figure 1: Vectors for Transaction Selection

2.1 Invocation Frequency

Assume that set t below denotes all the transactions in a fab:

$$t = \{t_1, t_2, t_3 \ldots t_n\}$$

And v_i is respective volume of transaction t_i

$$v = \{v_1, v_2, v_3 \ldots v_n\}$$

Declaring Sv as sorted set v in descending order ; St as equivalent sorted set t, we can derive Cv as cumulative volume set for Sv as follows:

$$Cv_r = \sum_{i=1}^{r-1} Cv_i + v_r$$

From a pure volume perspective, we drew a threshold to include only those transactions that contribute to top 95% of the total fab volume. This cutoff can be expressed as follows:

$$C_{cutoff} = 0.95 \times \sum_{i=0}^{n} Cv_i$$

Using the obtained cutoff above, we can derive the partitioning index i as follows:

index i : $Cv(i\text{-}1) < C_{cutoff}$ & $Cv(i) > C_{cutoff}$

Applying the partition index i in our transaction set S_t, the top 95% transactions filtered included $S_t(0)$ .. $S_t(i)$

2.2 Criticality of the transaction

Manufacturing units operate in an assembly-line like fashion. Consequently, presence of operational dependent transaction is almost guaranteed. For example, there could be one low volume transaction for every n high-volume transactions but slow execution of the low volume transaction would annul any benefits by speeding up the high-volume transactions. After series of discussions with domain experts and careful study of the use-cases, we arrived

at the list of critical transactions that were not included through transaction frequency filter – as they were not part of the top 95% transactions by volume.

2.3 Complexity of the transaction

OLTP queries to the database in the backend may involve complex queries. These queries could involve activities like multiple joins, obtaining table-wide locks, etc. Poor implementation of these queries may have cascading impact on other transactions impacting the database at the same time. After exhaustive database tracings, all such transactions were identified and included in simulation set.

2.4 Large Payload

Transactions that result in a large (from a memory perspective) intermediary or final reply from the backend should be included to ensure that memory characterization of the simulation is close representative to the system.

2.5 Transactions with Bursts

In Figure 2, s(t) is distribution of slopes for the transaction over time. A high value of standard deviation of s(t) signifies that the transaction has irregular bursts. Transaction with bursts needs to be included in simulation set – as their sporadic high volumes would impact the system temporarily. It is important to evaluate if the system would be able to auto-recover from these sporadic bursts.

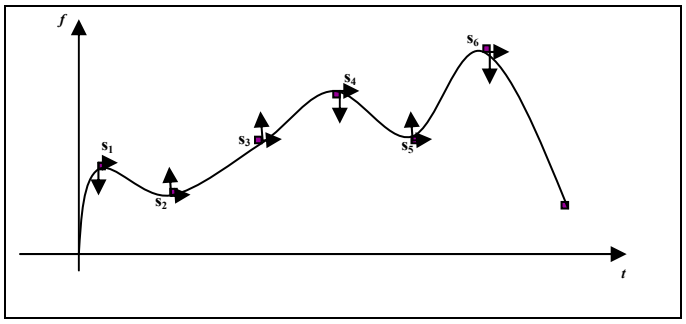

Figure 2: Determining Transactions with Bursts

3 DETERMINING TRANSACTION DISTRIBUTION

Historical data from existing 300mm Intel factories was used to estimate behavior of external forces on the system. While obtaining data is a mechanical process with some procedural and management related challenges, the bigger challenge is assimilating and interpreting the data. In order to gain some intelligence, we used the Anderson Darling test (Stephens 1974-1979).

The Anderson-Darling test is used to test if a sample of data came from a population with a specific distribution. It is a modification of the Kolmogorov-Smirnov (K-S) test and gives more weight to the tails than does the K-S test. Anderson-Darling test can be summarized in Fig ure 3.

H0:	The data follow a specified distribution.
Ha:	The data do not follow the specified distribution
Test Statistic:	The Anderson-Darling test statistic is defined as $A^2 = -N - S$ where $S = \sum_{i=1}^{N} \frac{(2i-1)}{N} [\ln F(Y_i) + \ln(1 - F(Y_{N+1-i}))]$ F is the cumulative distribution function of the specified distribution. Note that the Yi are the *ordered* data.

Figure 3: Anderson-Darling's Test

Commonly used simulation models often assume Gaussian distribution, apply the central limit theorem and then obtain normal distributions to obtain statistical aggregates becoming the basis for the model. However, this assumption can usually be 'dangerous' as the frequency tails could result in systemic chaos severely impacting the environment. Anderson-Darling test provides significant focus on these tails – making it a more appropriate choice for us.

Intelligent processing of historical data is critical in development of a high-quality simulation model. Anderson-Darling test enables in efficient translation of historical events into cumulative distribution functions. We extracted occurrence of all transactions that occurred in a fab over one month, aggregated their frequencies over 1 minute interval periods, and applied Anderson-Darling test. The outcome was probability distribution functions for each transaction, which were ultimately used to define the events for driving simulation.

A user on the fab floor could initiate events that lead to data being read from the database. These events do not cause any changes to the 'state' of the system – and hence are termed as 'state-less' transactions. On the other end of spectrum are transactions that lead to changes in the state of the system – termed as 'stateful' transactions. Stateless transactions are trivial to simulate. As long as the database of the system is correctly maintained, these transactions could be simulated based on their transaction frequencies. Stateful transactions, on the contrary, require preparatory and reset transactions to ensure that system remains in defined state. Preparatory transactions are events that set the system to the state required and desired for the test transaction to be simulated. Reset transactions are events that bring back the system to its original state after the test transaction has been executed. This approach of prep-reset

guarantees that the system would be in a defined state between any two atomic transactions.

4 SIMULATION MODEL DEFINITION

Figure 4 is a pictorial representation of a discrete event simulation by Ball (1996). So far, we have talked about transactions and their distributions. This section delves into defining entities used for simulation and their relationships; and simulation executive.

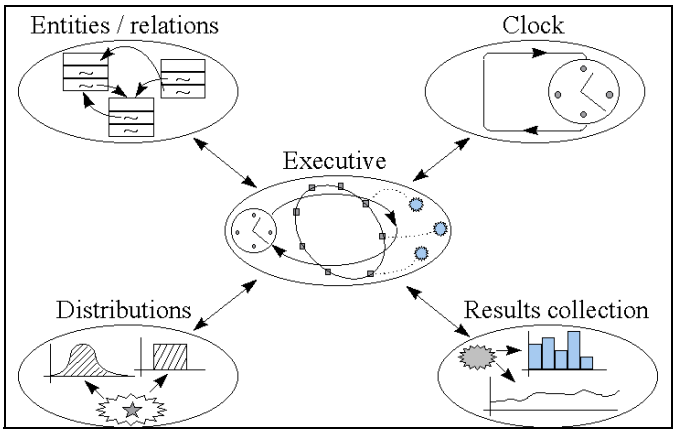

Figure 4: Defining Simulation Model

4.1 System Centric Simulation

The Fab facility can be very simplistically described as a system where a raw wafer is taken and is processed to yield a microprocessor. An operation-centric simulation approach would focus at lifetime of a wafer, thereby simulating various events in the fab. While, arguably, this might be the most effective simulation approach, it warrants the simulation executive to persist the system state throughout the simulation cycle. Persistence in the simulation executive is a fairly expensive task as it requires significant intelligence to be available within. Since, the objective of simulation was systems engineering, i.e. to ensure Reliability, Availability, and Performance of the system, we had a system-centric focus during the simulation.

This approach was a revolutionary mechanism for testing of mission critical software, eliminating the dependency on end-user interaction - thereby also eliminating the need for the user-interface in the simulation suite. This ensured that no test run will have to be scrapped because of failure of any single component of the simulation. The System-centric approach of simulation provided us with the same quality as operational-centric simulation approach, while providing much higher level of control on simulation. We focused on simulation of each transaction which was determined to be impacting the system significantly (see Section 2), and ensured that all the required collaterals (entities, their state, inter-relationships, structure, and historical data) were appropriately configured into the system state. We defined two categories of system 'cardinalities' to create a system that could be used for system-centric simulation – object cardinality, and transaction cardinalities. Simulation employed a unique method for setup of in terms of the object cardinality and other data collaterals, and the transaction cardinality, with no overlaps between any major threads of the required testing.

4.1.1 Object Cardinality

An unrealistically lightly loaded system would perform better than the real world giving us incorrect results, and vice versa. To ensure proper 'loading' of the system, we defined object cardinality as the "number of system objects that exist impacting any system transaction." Example, if there are *m* types of machines in a factory, and a total of *n* machines with a ratio of *1:x* between the machine type and number of machines, this ratio needs to be preserved in the simulation model.

4.1.2 Transaction Cardinality

For realistic simulation of an event, it is critical to ensure that the magnitude of change of system state is representative of the real system. Example, if a 'stateful' transaction results in changing state of *n* machines, it is important that the simulation model should also reflect changing state of *n* machines for the same transaction. Since, the simulation is system-centric, it is also critical that wherever applicable, different transactions use mutually-exclusive system objects.

4.2 Simulation Executive

Simulation executive was an in-house developed software that worked in a plug 'n play mode – allowing user to include additional transactions, change transaction behavior, volumes, and level of statistical collections on the fly. We invested into development of a simulation executive to enable us in quick turnaround for deployment of new simulation scenarios and effective simulation data collection.

5 RESULTS AND CONCLUSIONS

Approximately 250 system engineering issues, and 450 OLTP performance issues were uncovered during 18 months testing, using the simulation approach described in this paper. Out of these, about 40 were showstopper issues – which could have caused factory down situations.

Employing different frequency distribution function for different transactions with a system-centric simulation was a novel approach in our simulation model helping us deploy a sturdier and more stable system in our factories. We have expanded this approach beyond MES to other

mission-critical automation systems. This simulation model continues to be used for simulating fab at higher-volumes and identify any potential issues inhibiting factory ramp-ups,

REFERENCES

Ball, P.D. 1996. Introduction to discrete event simulation for the *Proceedings of the 2nd DYCOMANS workshop on "Management and Control : Tools in Action",* 367-376.

Stephens, M.A. 1974. EDF Statistics for Goodness of Fit and Some Comparisons for the Journal of the *American Statistical Association*,. 69: 730-737.

Stephens, M.A. 1976. Asymptotic Results for Goodness-of-Fit Statistics with Unknown Parameters for the *Annals of Statistics,* 4: 357-369.

Stephens, M.A. 1977. Goodness of Fit for the Extreme Value Distribution for the *Biometrika,* 64: 583-588

Stephens, M.A. 1977. Goodness of Fit with Special Reference to Tests for Exponentiality for the *Technical Report No. 262, Department of Statistics, Stanford University, Stanford, CA*.

Stephens, M.A. 1979. Tests of Fit for the Logistic Distribution Based on the Empirical Distribution Function for the *Biometrika,* 66: 591-595.

AUTHOR BIOGRAPHIES

AMIT JINDAL is a software engineer in Software & Solutions Group with Intel Corp. He was simulation strategist for the 1264 program, leading development of simulation strategy, and model for the1264 automation He is currently focusing on SAP Engineering with the latest high-end microprocessor architecture to yield industry-highest performance benchmarks. He holds an MBA degree in Global Management from University of Phoenix, and MS degree in Computer Science with focus on distributed computing from Arizona State University.

RAJKUMAR KHANDELWAL is a software engineer with 14years of industry experience. He holds a Masters degree in computer science from Arizona State University. He has held a multitude of jobs in his career including application development, systems programming, database and systems administration, and management of varying Projects. In his last 7 years at Intel, he has managed Projects related to delivering automation capabilities for manufacturing, implementing changes for infrastructure technologies to meet Intel's needs, and recently in performing systems engineering for the automation for 1264 which is Intel's latest manufacturing process. He is currently leading the stress validation of the Intel's automation suite to meet the ramp needs.