Interfaces
Vol. 40, No. 2, March–April 2010, pp. 105–117
ISSN 0092-2102 | EISSN 1526-551X | 10 | 4002 | 0105

informs®
DOI 10.1287/inte.1090.0479
© 2010 INFORMS

Using Simulation Early in the Design of a Fuel Injector Production Line

Mustafa H. Tongarlak, Bruce Ankenman, Barry L. Nelson
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208
{mtongarlak@u.northwestern.edu, ankenman@northwestern.edu, nelsonb@northwestern.edu}

Laurent Borne
McKinsey & Company, Chicago, Illinois 60603, laurent_borne@mckinsey.com

Kyle Wolfe
Delphi Corporation, West Henrietta, New York 14586, kyle.s.wolfe@delphi.com

In this paper, we describe how Delphi Corporation used simulation in the concept-development phase of a new multimillion-dollar fuel injector production line. Delphi wanted to assess the financial viability of production targets and identify the critical features of the line on which it would focus its design-improvement efforts.

Key words: simulation: applications, design of experiments.
History: This paper was refereed. Published online in *Articles in Advance* February 16, 2010.

Delphi Corporation is a major supplier of fuel injectors to automobile manufacturers worldwide. The company was considering a proposal for a new line that would produce the next generation of fuel injectors. Its ambitious requirements on the line's throughput were essential for the project to be financially viable; in addition, Delphi had to address constraints on the space available for the line and on the cost to build it and staff it.

As envisioned, this production line will consist of multiple process segments, three of which will be placed in a clean room (Figure 1).

Each process segment may contain as many as 30 machines and robots that perform various tasks, along a series of conveyors that convey partially completed fuel injectors from machine to machine in pallets. The conveyors will also act as buffers; they will accumulate pallets in front of machines that are either down or unable to keep up with the flow of pallets from upstream machines. Transfer of assembled fuel injectors between process segments will be done via trays that are stacked in carts and moved manually by operators. Because contamination inside the clean room is a major concern, parts will be washed before they enter the clean room and operators will be assigned tasks that do not require them to move between rooms.

Delphi commissioned a simulation model and study at a very early stage of designing the new line. The simulation model's goal was to serve as a test bed for candidate production-line designs, including initial concept, fine-tuning, and examining process-improvement ideas after the line was operational. Simulating a line design must include fully specifying which machines to use and in what order, conveyor lengths, machine process rates and variability, operator responsibilities and priorities, failure and repair distributions for each machine, and scrap and rework rates on each machine. Such complete and detailed information is typically not available at the concept stage of a line design; however, this is precisely when simulation could be most valuable in helping Delphi to assess the project's financial viability and determine where in the evolving line design to expend the most effort. Using simulation at such an early stage requires frequent and significant changes to the simulation base model; these changes can be both time consuming and difficult to manage. However, the simulation recommendations are often inexpensive or free to implement because the equipment has not yet been built. After the equipment has been built, using simulation to analyze the process is less time consuming because the base model is no longer a moving target. The drawback is that much of the cost

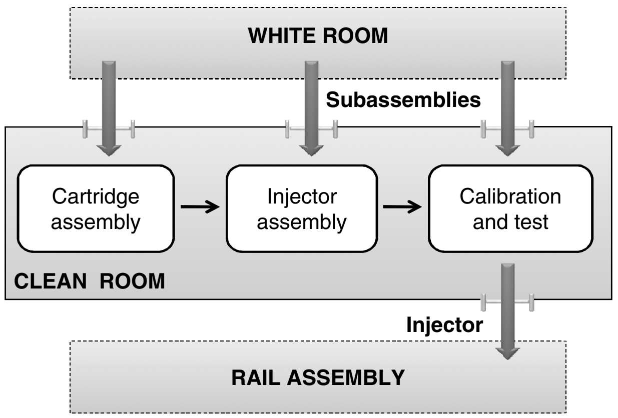

Figure 1: The three process segments inside the clean room are complex, highly automated, and costly.

has already been sunk and the study's recommendations may be more difficult to cost justify. This was our challenge.

Trying to optimize the design over the literally hundreds of changeable features early on is senseless because too little information is available and too much is in flux. Therefore, we used simulation to answer high-level questions about the line configuration, with the understanding that the model would later evolve into a representation capable of evaluating very specific questions. That is, the simulation's objective was first to guide the development of new designs by distinguishing critical features and factors from less-critical ones; later it would be updated as new designs emerged. Thus, the simulation would provide a perpetual road map for the next steps in the design process. This paper describes the first phase (about six months) of this iterative design process, a period in which answering high-level questions about line configuration, conveyor layout, and operator assignments was of primary concern. In the remainder of this paper, we will illustrate how simulation can have a profound impact at the concept stage of a complex project.

System Description and Data Sources

Clearly, machines (including robots) and conveyors must be included in the simulation model. Most machines will keep running until a failure occurs as long as they are not starved (because of a lack of material) or blocked. Because conveyors can handle the subassemblies within process segments, operators are needed only for transferring assembled fuel injectors between process segments, filling raw material buffers, repairing machines, and periodically performing other tasks, such as quality checks, rejected fuel injector handling, and preventative maintenance. Although the system is semiautomated and operators are only lightly involved with actual operations, their supporting role in the system is critical because machine failures will occur, and input buffers will be consumed, leading to lost production if operators are unable to fulfill their required tasks. Therefore, in addition to machines and conveyors, the model included operators, injector trays, and material carts.

Delphi represented line designs as AutoCAD drawings (that eventually became backgrounds for the simulation animation). More importantly, the company also maintained a single Excel spreadsheet with many worksheets, the manufacturing system design (MSD), which always contained the current state of knowledge about the line design. This included process information of all types from raw material buffer sizes and operator task assignments to anticipated machine reliability, processing time, and processing-time variability. Because this information ranged from firm values and commitments to educated guesses, and would evolve and expand throughout the project, we established a link between the MSD and the simulation model to ensure that the simulation would always be run using the most current data.

However, the MSD was more than just a data source; we also used it to balance the line to produce k fuel injectors each minute (k is a number that Delphi set and that we cannot reveal). In this semiautomated system, which comprises a series of connected processes, "balancing" simply means that if a particular operation is not able to produce the target k parts per minute, then Delphi adds enough parallel capacity to keep up with the production requirement. Because the line will operate for 24 hours per day, its production capacity would be $1{,}440k$ parts per day if we did not have to deal with machine downtime, scrap, or process variability, and if material buffers would never go empty. However, to be more realistic, the MSD also incorporated discount factors for the percentage of scrap and downtime by machine, leading to a still optimistic throughput target of 1,225*k* fuel injectors per day—approximately 22 percent less than the theoretical maximum. Achieving this target is necessary for the new fuel injector project to be financially viable.

The target 1,225*k* fuel injectors per day may be optimistic because a static analysis such as the MSD provides cannot account for the impact of process variability, starvation because of lack of material or blocking, operator response time to failures, conveyor congestion, etc. More-accurate analysis of the proposed system requires the fidelity of a detailed simulation model that considers the interactions among all parts of the system and the randomness inherent to the system. The critical question that the simulation had to help us answer at the concept stage was whether 1,225*k* fuel injectors per day was actually feasible and what Delphi would have to do to achieve it.

In the following subsections, we list some key system elements that were part of our analysis and mention any approximations we made.

Resources

Each machine is prone to failure. Mean time before failure (MTBF) represents the average time a machine runs until a failure occurs. When a machine fails, it cannot produce parts until an appropriate operator has arrived and fixed it, a period called downtime. The average repair time is denoted by mean time to repair (MTTR). The MTBF and MTTR are values derived from information in the MSD. We modeled the distributions of the time to failure and repair as exponential; we chose the exponential distribution because, in the absence of real data, it can be specified by a single parameter and reflects the high variability that Delphi observed in existing lines.

Operators

Three types of tasks need operators: machine-attention (MA), periodic (P), and material-handling (MH) tasks. Each requires a different set of operator skills. MA tasks have the highest variability and MH tasks are the most regular. Therefore, operators will not be assigned both MA and MH tasks. However, all operators will have some P tasks that they must perform, such as quality checks, reject handling, and maintenance.

P tasks must be performed regularly—every 2, 4, 8, 24, or 120 hours. MH tasks—much like P tasks—are performed periodically but more frequently than P tasks. When P tasks occur, they have priority over MH tasks. MA tasks are performed as needed; i.e., an operator who is responsible for fixing a specific machine must attend to this duty when that machine fails. This random nature of MA tasks makes allocating jobs to operators challenging, and poor operator assignments can hamper productivity. For example, suppose that a specific operator is responsible for repairs on both machines A and B. If machine A fails, the operator will respond to that machine as soon as possible. However, if machine B fails while machine A is still being repaired, the actual downtime of machine B will include not only its own repair time but also the remainder of the time it takes the operator to complete the repair of machine A and any travel time between the two machines. In general, because operators can be responsible for repairs on many machines, this effect can be multiplied many times and cause substantial downtime. Thus, to whatever extent possible, the MA and P tasks should be balanced among the qualified operators such that a single operator is not responsible for repair of more machines than necessary. To define a base case for operator assignments, we assigned each MA and P task to only one operator; we then grouped the tasks by their proximity to each other and approximately balanced them considering the relative reliability of each. We assigned each MH task to an operator who handled no MA or P tasks and developed an extensive point-to-point walk matrix to accurately account for operator travel times.

Figure 2 shows the operator-activity logic we used in the simulation model; it reflects Delphi's operator policies for its existing lines. Each operator is given responsibility for performing specific tasks; operators who complete a task seek another task unless their shift is over or it is time for lunch or a break. Lunches and breaks have priority over tasks. If a machine is down, filling buffers or handling rejects instead of fixing the machine is considered an inefficient use of operator time; therefore, MA tasks have the highest priority of the three task types. In addition, if two tasks of the same type become due or call for operator attention, they must be completed based on a fixed

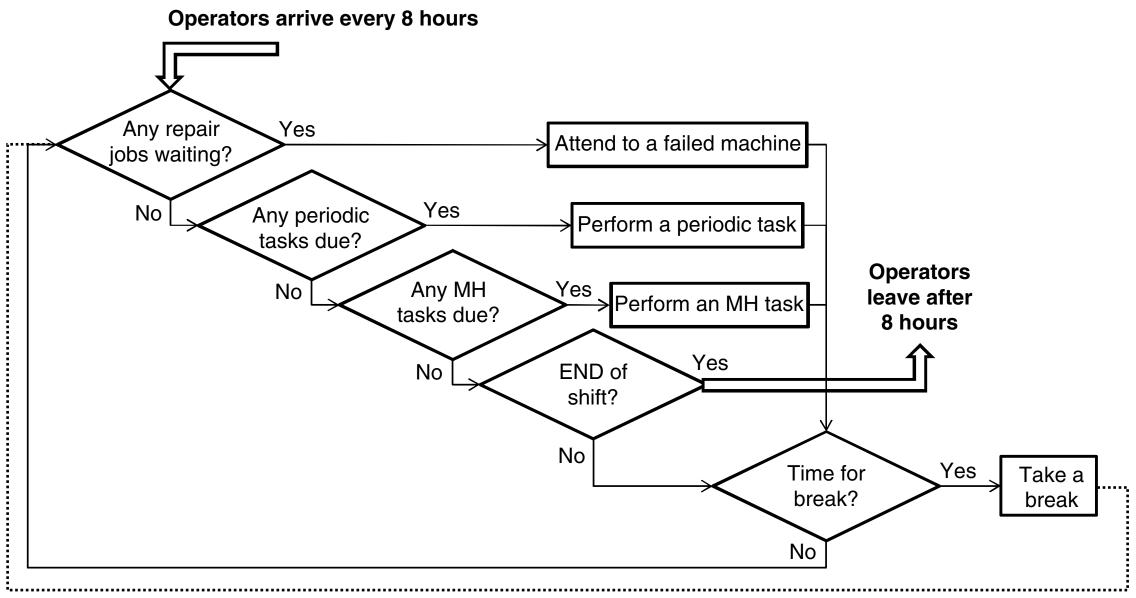

Figure 2: This flowchart of operator tasks shows the operator-activity logic used in the simulation program.

priority order. For example, if operator 4 is responsible for repairing resources 5 – 3 – 4 – 6 in that order, then 5 is repaired first if both 5 and 3 fail. Repairing a failed unique machine (i.e., the only machine that can handle a specific task or set of tasks) is given priority over repairing one of a group of parallel machines. Operator modeling, which we describe in more detail in Appendix A, was the most difficult aspect of building the simulation.

Process Variation

Each machine has an average processing time given in its specifications. Because some level of variation exists in processing rates, producing k fuel injectors every minute, even when no scrap is produced and all machines are operational, is not possible. Because process variation affects production, it must be embedded in the model.

With help from the Delphi production team, we classified the process variation of each machine into one of the following three categories based on its knowledge of the process: high, medium, or low variation. We assigned a coefficient of variation (CV), defined as the ratio of the standard deviation to the mean, for each of these categories. Because the mean process time was specified, we could then calculate the process standard deviations using these CV values. Using this mean and standard deviation, we specified a normal distribution to generate random processing times for each part that machine processed.

Simulation Model

We developed the simulation model using version 12.00 of Rockwell Automation's Arena simulation software (Kelton et al. 2006). It was a good choice for the project for the following reasons:

- Arena is well-suited to modeling conveyors, which are the major mode of material transfer in Delphi's production lines.
- It provides a user-friendly interface that makes understanding and modifying the model easier for people other than the modeler. In addition, it gives the user the ability to design templates that can be used to create similar code for the parts of the model that are likely to be repeated (although with different parameters). For example, we designed a template that handles the tasks performed when a part arrives for a machine; examples include picking the part from the conveyor, checking for raw material inventory, processing the part, and placing it back on the conveyor. When the template is available, a modeler only needs to define machine-specific parameters, such as

conveyor name, machine number, and buffer type; the template will automatically generate customized code.

• The model can be linked to a data source (e.g., the MSD, which is an Excel spreadsheet) to allow for making most minor updates (e.g., changing MTBF for some machines) directly using Excel rather than Arena. This also preserves data integrity because all Delphi analyses use a common data source.

• Arena provides animation capability that can be used for model validation and demonstration purposes.

• Arena collects and reports most of the necessary statistics by default and lets the modeler add other statistics.

Simulation Experiments and Results

The main performance measure in our simulation study was long-run average daily fuel injector throughput, i.e., the long-run average number of fuel injectors that can be produced in a 24-hour period. Therefore, we treated this as a steady-state simulation (Banks et al. 2005) requiring a "warm-up period" to get from the initial state (conveyors empty but all input buffers full and all machines operational), to long-run operating conditions. For the base case (described below), approximately one day's production was adequate (see Appendix B). Therefore, we made each replication 11 days long; we discarded the output data from the first day and retained the next 10 days (two weeks) of data. We then made enough replications to estimate long-run average throughput to within a 5 percent relative error; this required 20 replications. We empirically determined the number of replications so that (halfwidth of confidence interval)/(sample mean) was less than 0.05 for the estimated throughput. Running the scenario required approximately eight hours to obtain 20 replications of 11 days on a relatively fast PC. This is a function of both the size and complexity of the system and the very large number of fuel injector subassemblies in process at any one time. Because all the other scenarios are modifications of the base case, we used the same experimental effort (20 replications of 11 days) in each scenario tested. Although throughput drove our performance measurement, measures of operator utilization, machine downtime, congestion on conveyor segments, and raw material buffer states were also examined.

We compared the results of all experiments with the results of two benchmarks: the base case, which is a simulation of Delphi's line design as specified in the MSD, and the target of 1,125k fuel injectors per day. If we had not found a gap between these two benchmarks, then the proof-of-concept phase of the line design would have been completed. Instead, we found a very substantial gap; therefore, our objective became to find the key factors producing the gap so that they could become the focus of future design efforts. Because we did not have any specific design alternatives for comparison, we developed the experimental approach described below. Although the critical factors were unexpected, they were completely understandable after the fact.

First-Phase Scenarios

To design our preliminary experiments, called "first-phase scenarios," we listed all the factors that might contribute to the loss of production. Obviously, if no such factors existed, then we would be consistently producing 1,125k fuel injectors per day. We listed process variability, scrap rates, and downtime related to operator availability as potentially critical factors, and conveyor lengths and number of pallets on each conveyor were listed as secondary factors. We focused our initial experiments on the potentially critical factors; we felt that conclusions about conveyors would be premature at this early study stage because major changes to the line layout were very likely.

In the simulation, we could control downtime by manipulating the machine repair time. Because our study involved 25 machines, each with a specified scrap rate, process variability, and repair time, we could analyze 75 factors using classical design-of-experiments methods. However, even if we prioritized and selected only 30 of these 75 factors and examined only the main effects, we would require at least 32 runs. At eight hours per test plus the time necessary to set up scenarios and analyze results, this experiment would take about two weeks. At the concept stage of the line design, investing two weeks of time and computational effort to estimate the effects of each individual factor is not a worthwhile use of time; many details are still in flux and major changes to equipment and conveyor layouts are almost certain. Therefore, we used a group screening approach, grouping the factors into functional categories. By categorizing the impact or lack of impact of these functional groups, we could answer broad questions about which factors have the greatest impact using reasonable computing effort.

For example, instead of investigating each machine's process variability individually, we investigated the effect of eliminating *all* the process variability. This allowed us to quantify the maximum impact that process variability has on throughput and saves the effort required to assess the effect of process variability for each of the 25 machines if we find that process variability as a group is not very significant. We constructed analogous scenarios to examine the group effects of scrap rates and repair times. Thus, the first-phase scenarios included the following:

- Scenario 1: The base case is the model representation of the factory as reflected in the MSD file and the AutoCAD drawing of the layout.
- Scenario 2: All process variation is removed from the base case.
- Scenario 3: Process variation and all scrap are eliminated from the base case.
- Scenario 4: Repair times are set to zero; i.e., as soon as an operator attends to a failed machine, it starts running again and the operator can leave for the next assignment.

Scenario 2, which eliminates the process variation, was especially critical because the variation input to the model was based on expert opinion and estimated CVs. Therefore, if eliminating all the variation does not achieve a significant production gain, then putting effort into creating better CV estimates for the processing time of each machine is not essential at the conceptual design stage. However, if it makes a significant difference, then we should not make radical design changes before we carefully characterize this variation.

Scenario 3 eliminates both scrap and process variation. This scenario was intended to inform us of the sensitivity of throughput to scrap rates and to determine if improving the machine scrap-rate estimate before proceeding is critical. The purpose of eliminating both process variation and scrap rate together is to use this scenario as a "best-case scenario" for this design candidate.

Scenario 4 takes a different approach than Scenarios 2 and 3. In this experiment, we set repair times to zero but retained scrap and process variations. Our goal was to determine the potential of the production line when repairs are done infinitely fast and machines are down only for the time required for the designated operator to attend to the machine (i.e., response time, which is time to complete a task in progress and walking travel time). When we set repair times to zero, operators became more available and response times to failures decreased. Although we recognized that this level of service is impossible to achieve, this scenario showed us the benefits of improving machine-attention activities.

First-Phase Results and Analysis

For the base-case scenario, the average daily production value estimate was only 800*k* fuel injectors per day or 71 percent of the target throughput for the new line. This left a gap of nearly 30 percent between planned and realized production. The results of Scenarios 2–4 shed light on what caused this gap and what design changes would give the largest improvement.

In Scenario 2, when process variation was removed from all the machines, daily production relative to the goal increased to 75 percent. In Scenario 3, in which we eliminated both process variability and scrap, this percentage became 76 percent. Both of these drastic and unattainable changes to the system produced only slight increases in productivity. Therefore, we concluded that more-detailed analysis of machine process variation and scrap rates should be left to later stages of our study after we made more influential changes to the system.

In Scenario 4, in which we reduced repair times to zero, we achieved our goal of 1,125*k* fuel injectors per minute; the average daily production for this scenario was 102 percent of the target. This significant jump from the base case was a good indication that machine downtime is a primary cause of the low production in the base case. However, before we jumped to a conclusion too quickly, we returned to the base-case scenario and looked at statistics that we had collected to discover other reasons for lost production. We were interested in four types of analysis: operator availability, pallet congestion, inventory levels, and failure analysis.

Operator utilizations in the base case showed that all operators were heavily utilized. Some operators were so very heavily utilized that they often did not complete their periodic tasks, such as quality checks during their shift; thus, they pushed these tasks to the following shift. For specific tasks, the operators in subsequent shifts were also too busy to complete these tasks; therefore, these jobs would queue up and never be completed. An even more significant consequence of having heavily utilized operators was that many machine failures did not get immediate attention from operators; therefore, operator response time to machine failure began to overwhelm repair time as the primary cause of machine downtime. This raised the machine downtime for most machines far above the expected downtime and caused dramatic production losses.

Congestion on conveyors is also a system measure. A very straightforward way of determining which conveyor segments experience higher congestion is to look at which segments are heavily utilized for a long time. Before running any scenario, we created four utilization states: fully, highly, moderately, and lightly utilized. Throughout the 10 days of each replication, we monitored the time that each conveyor segment spent in each utilization state. Conveyor segments that spent a high percentage of time in fully or highly utilized states were marked and compared to the machine downtimes in the vicinity of these conveyors. We concluded that for the base case a very strong relationship existed between operator inability to respond quickly to machine failures and heavy utilization of the conveyor segments just upstream of those machines.

Raw material buffer inventory is also relevant. We looked at whether any machines went idle because of lack of material. All our scenarios included one operator who was dedicated to raw material handling with no machine-repair responsibilities. Thus, this function was not affected by the excessive machine downtime. From analyzing inventory buffers, we concluded that the material-handling assignments for the base case were good and needed only minor adjustments.

Finally, we performed some detailed failure analysis because overutilized operators and highly congested conveyor segments might be symptoms of another problem: frequently failed machines. As with the conveyors, we collected statistics on percentages of the time when each machine was idle, busy, or failed. For each machine in the system, we compared resulting downtime percentages with the expected downtime from the MSD model. For the majority of machines in the base case, the downtime percentages were two to three times higher than expected (Figure 3). Thus, we concluded that excessive machine downtime was the primary cause of the 30 percent gap between the target and the base-case scenarios.

The three direct contributors to the average machine downtime are MTBF, MTTR, and mean operator response time (MORT). We can calculate percentage downtime (PD) as

$$\mathrm{PD} = 100(\mathrm{MORT}+\mathrm{MTTR}) / (\mathrm{MTBF}+\mathrm{MORT}+\mathrm{MTTR}). \qquad (1)$$

We estimated machine PD for the MSD model by observing current machines that are similar to those planned for the new production line. Because it directly affects the production, PD for each machine is often recorded. MTBF for the MSD model is also easily collected because machines often include run-time monitors. However, separating the operator response time from the repair time is not typically done; doing so would be very difficult unless we do a lengthy

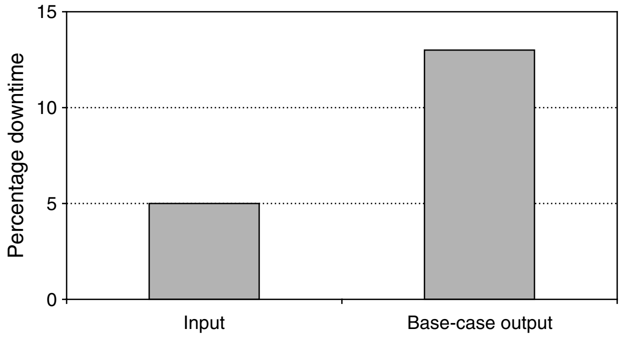

Figure 3: The graph shows a comparison of input and output downtime percentages for a typical machine in the base case.

time study on each machine. Although we can easily estimate the sum of MORT and MTTR, we cannot estimate either value individually. When setting up the base-case scenario, we discovered this problem because each machine must have a distribution specified for both time before failure and time to repair. To simplify the problem, we assumed that if operator assignments were designed well MORT would be substantially less than MTTR and thus negligible. Therefore, solving Equation (1) for MTTR, we get

$$\widehat{\text{MTTR}} \approx \widehat{\text{PD}} \times \widehat{\text{MTBF}}/(100 - \widehat{\text{PD}}), \qquad (2)$$

where $\widehat{\text{PD}}$ and $\widehat{\text{MTBF}}$ are the estimates of machine PD and MTBF from the MSD. Given the results of Scenario 4, our assumption that MORT is negligible was incorrect for the base case and the approximation of MTTR in Equation (2) is poor.

Our first response to this information was that we should make all reasonable efforts to determine a good estimate of MTTR for each machine to simulate the base case correctly. However, this would have required substantial time to collect data at a current production facility with similar machines—if we could find such a facility. Another option, more in the spirit of using simulation early in the design process, was to redesign the base case to reduce the operator load and thus reduce the impact of MORT. We developed additional scenarios to help us to understand the problem of operator loading on the base case and potentially suggest new design directions.

Second-Phase Scenarios

In light of the results of the first-phase scenarios, the second-phase experiments focused on operator availability for machine failures. The operator assignments for the base-case scenario involved some critical assumptions, some of which might have a substantial impact on daily production numbers. For example, in the base-case design, we assigned each task (quality check, machine repair, or material-handling job) to a specific operator; no other operator on that shift could respond to that task. As a result, when a particular operator was heavily utilized, failed machines would often wait for long periods of time before the assigned operator would arrive to start repairs. Similarly, when an operator was at lunch or on a break, repair of any failed machine assigned to that operator would have to wait until that operator's lunch or break was over. In this highly connected production system, any deviation of a machine from its production target is enough to slow down finished fuel injector throughput.

Another base-case assumption was that operators leave for lunch or a break on schedule as soon as they have completed the task at hand, potentially leaving one or more machines failed through an entire lunch or break period. It also assumes that all operators go to lunch or break at the same time, thus complicating the downtime problem. Clearly, the throughput gap might be reduced by redesigning the lunch and break times of operators, possibly cross-training some operators to provide backup for others, or by resetting the priorities for lunches, breaks, or specific machines.

To quantify the operator-redesign potential, Scenario 5 used the base case but eliminated lunches and breaks from operator schedules. We designed this experiment to determine an upper bound on the production gain that could be achieved by optimizing the rules on operator lunches and breaks; the base case had implemented a worst-case scenario in which lunch and breaks have priority and all occur at the same time. Of course, requiring operators to work without any breaks is unrealistic, as is scheduling all operators to take lunch at the same time. However, the objective of our experiments was to guide the design effort—not to make a final suggestion for implementation. Furthermore, practical ways could be implemented for lunch and break coverage, such as adding support operators to relieve operators when they leave for breaks, thus ensuring continuous coverage for all tasks.

The selection of Scenario 6 was intended to determine how much the approximation of MTTR in Equation (2) affected the daily production rate in the base case. Recall that MTTR for each machine is a direct input to the simulation model. More specifically, each time machine j failed, a random repair time was drawn from an exponential distribution with mean $\widehat{\text{MTTR}}_j$, which was calculated from Equation (2) using estimates of the downtime and MTBF for that machine or comparable machines from current production lines. Conversely, operator response time is a random output of the simulation model that depends on the random machine failures and the operator utilization levels. The approximation in Equation (2) used for the base case assumes that all downtime is repair time, an assumption that we found to be inaccurate. Scenario 4 went to the other extreme and assumed that all downtime was operator response time, an assumption that we also knew was untrue because repairs take time. Scenario 6 attempts to compromise between these two scenarios; it assumes that the repair will account for about half of the downtime and operator response time will be the other half. Thus, in Scenario 6, we halved the MTTR for each machine. Therefore, we ran the following two scenarios as second-phase scenarios:

• Scenario 5: Lunches and breaks are fully staffed with no reduced efficiency.

• Scenario 6: MTTR is halved. This scenario anticipates that about half of the machine downtime is operator response time.

In both scenarios, the base-case assumptions are kept the same except for the changes mentioned. Thus, they include scrap and process variability rates at the same levels as in the base case.

Second-Phase Results

In Scenario 5, where lunches and breaks were fully covered by relief operators, the expected daily production value relative to the production target was 85 percent. Despite substantial improvement, this showed that even if we redesigned the base case to provide full operator coverage during lunches and breaks, a gap between the achieved production and the target of 1,125*k* fuel injectors per day would still exist. However, this scenario still used the assumption that operator response time is negligible and investigation of the downtime for this scenario still did not match the expected downtime from current production lines. However, in Scenario 6, which includes operator lunches and breaks but cuts MTTR in half, the throughput is 92 percent of the target. This indicates that if the MTTR rates were indeed half of the originally proposed MTTR rates, then production rates would almost meet expectations (Figure 4).

Interpreting Scenario 6 requires some caution. The operator response time in the simulation model is based on the operator staffing level and assignments. Recall that the downtime estimates from current production machines are also based on some level of operator staffing and an unrecorded system of task assignments (that are not necessarily relevant to the new production line). Thus, the conclusion of this second phase is that it is vitally important to determine an estimate of the MTTR for each machine that is accurate and independent of the operator response time. We can only trust estimates of the daily throughput if we have accurate estimates of the MTTR. Once accurate estimates of the MTTR for each machine are collected, new design candidates can be accurately tested. In particular, Scenario 5 indicated that substantial increases in daily throughput could be gained by setting up an operator schedule that provides task coverage for operators when they take lunch or breaks. Table 1 summarizes the results of all six scenarios.

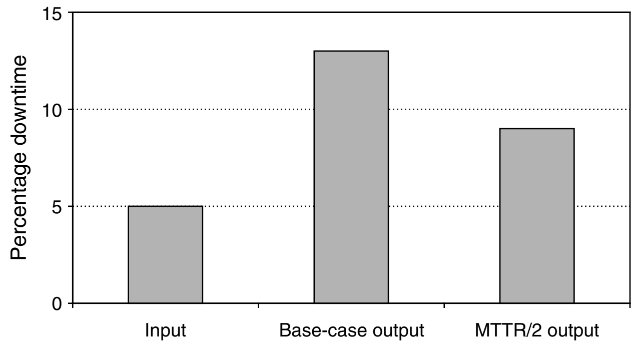

Figure 4: The graph illustrates downtime percentages in the base case and in the MTTR/2 scenario compared to input value.

Scenario	Daily production (relative to target) (%)
1: Base case	71
2: No process variability	75
3: No process variation and no scrap	76
4: No repair time	102
5: Full relief during lunch and break	85
6: MTTR/2	92

Table 1: The data show a comparison of daily production values.

Conclusions

Many real-world projects provide moving targets as the business climate, physical and financial constraints, and product concepts evolve. This might suggest that simulation, which is a detail-oriented methodology, is not suitable for system design until quite late in the development of a manufacturing system. However, Delphi's ongoing use of simulation to design and evaluate its next-generation fuel injector production line illustrates the value of simulation, even very early in the design process. The key is using simulation to answer questions at the right level, including questions about project viability and appropriate concentration of effort during system design. A simulation can address these needs before detailed specifications are available. Furthermore, if the simulation is constructed in a way that makes it reasonable to update and refine, then it can continue to contribute throughout the project life cycle.

We learned the following lessons about using simulation early in a system-design project:

- The simulation model should be constructed so that it can be easily changed. Ideally, it should be linked to the same data sources that are being used for the overall project planning.
- Use group screening to assess which design features or data sources, taken together, impact performance; thus, as the project evolves, scarce engineering effort can be concentrated where it is most useful.
- Avoid the temptation to create or define detailed production-control plans; instead, use easy-to-define, extreme scenarios that bound potential improvements or effectiveness.
- Focus not only on the output performance measures of most interest (e.g., throughput) but also record measures that provide insight into where problems might be occurring.

Epilogue

Upon completing the first six months of the concept-development phase, we presented our results and conclusions to Delphi engineers and managers, including a visiting European manager. This was the first major manufacturing simulation project for the Delphi Powertrain division in 10 years. Our presentation generated a lot of feedback and sparked interest in carrying forward the simulation study for another three months. Taking the feedback into account, we designed the next phase as a refinement of the concept-development phase.

Our first task was to carefully calibrate downtimes to set MTTRs, paying particular attention to machines with the largest downtimes. Then we experimented with three alternative coverage options for operators: (1) full relief (operators continuously work without any breaks), (2) no relief (operators take periodic breaks during which their responsibilities await their arrival), and (3) staggered breaks with coverage (operators working in the same area never overlap their breaks to allow them to cover for each other). Only option 1 adds employees to cover for breaks. Options 2 and 3 require no additional resources, and our results favored option 3 as the base case moving forward.

After setting the base case, we then moved on to improving conveyor layouts. Based on a congestion analysis of conveyors, we listed segments that were highly congested and could benefit from lengthening. We added to this list those segments that are very long and might add unnecessary travel time and take up space. We defined three levels (long, medium, and short length) for all the segments in the list and ran a fractional factorial design to find the optimal level for each segment. This analysis showed which segments could significantly enhance throughput by being lengthened or shortened, and it identified segments for which throughput was insensitive to their lengths.

We constructed new conveyor layouts based on the suggested sizing of the conveyor segments. One conveyor layout resulted in a 7 percent improvement in throughput compared with the current layout and a 10 percent reduction in required floor space for no additional investment. The Delphi team embraced this layout as a viable alternative to its original proposal. Finally, a pallet study showed that throughput was insensitive to the number of pallets within a very wide range.

Insights gained by this simulation study convinced Delphi to continue to update the model as its equipment design evolved. Its management was impressed by the positive results of this project and plans to use simulation on future programs.

Appendix A. Modeling Operator "Bus Routes"

Within Delphi, operator work assignments (Figure 2) are referred to as "bus routes." The name was derived from the observation that operators have sequences of repetitive tasks that they perform on specific cycles. However, unlike the typical city bus route, deviations from the fixed route occur because of, for example, machine failures that demand immediate attention before the route can resume. This appendix describes how we used specific Arena features to simulate the complex work assignments of operators and, in particular, their interactions with failed machines.

In a typical Arena simulation, operators and machines are represented by Arena resources. An Arena resource is, effectively, a variable that keeps track of the status of the operator or the machine that operator represents. When represented as a resource, a machine becomes unavailable when it is processing a job or has failed; similarly, an operator who is performing a task or is on a break is considered unavailable. This logic for resources is usually sufficient for modeling machines with dedicated repair staff members who are ready to start fixing problems as soon as they occur, or for modeling a pool of operators whose only responsibility is to respond to failures of a group of machines. Often the machine and operator resource correspond one to one.

However, this approach is inadequate for modeling the more complex scenario we faced in which operators must perform a variety of duties and machines that have failed must stay unavailable as they wait for operator attention. The machines we modeled are semiautomatic; therefore, they do not have dedicated operator support. The operators in the system have duties of various types, ranging from machine-attention tasks to material handling and quality checks that make representing these tasks using the standard Arena resources inappropriate. Therefore, we used Arena's flexibility to implement a custom approach.

To represent the operators in the model, we created an Arena entity for each operator at the beginning of each shift and disposed of it at the end of the shift. As Figure 2 shows, operator entities follow a routine in which they perform specific tasks in the priority order assigned to them. As opposed to the passive involvement of operators as resources that are used by entities, operators are actively involved in choosing their next task as soon as they become available. Repair jobs, periodic tasks, and material-handling tasks are created in other parts of the model, and operators respond to them at their earliest opportunity according to the task priorities.

Although we did represent machines as resources, we chose not to use the built-in failure logic of the Arena resources because it does not allow the time a machine spends waiting for an operator to depend on the current state of that operator. Instead, we established three states for each resource—busy, idle, and failed; idle is the only state in which the machine is able to serve a new part. In the busy state, the resource is processing a part; when it completes its processing, it becomes idle and ready to receive the next part waiting to be processed. The failed state is active when the machine is down and unavailable until an operator attends to it. To model this, we created a special failure entity for each resource. This entity takes hold of the resource when it is time for the resource to go down because of a failure. The failure entities have priority over entities representing parts waiting to be processed; they can seize and therefore fail the machine as soon as they appear. Figure A.1 displays the logic of this simple approach in a stylized way.

1. First, a failure entity is created for each resource; that entity starts in the leftmost block, where it is delayed for t_1 seconds before moving to the second

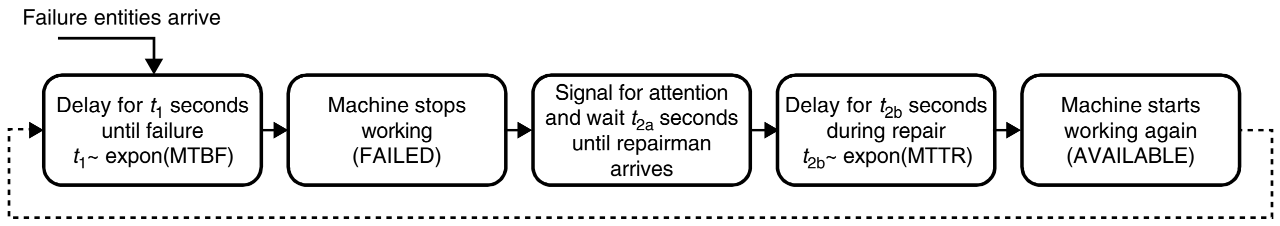
Failure entities arrive
Delay for t_1 seconds until failure t_1~ expon(MTBF)
Machine stops working (FAILED)
Signal for attention and wait t_{2a} seconds until repairman arrives
Delay for t_{2b} seconds during repair t_{2b}~ expon(MTTR)
Machine starts working again (AVAILABLE)

Figure A.1: The flowchart illustrates failure modeling in the Delphi simulation.

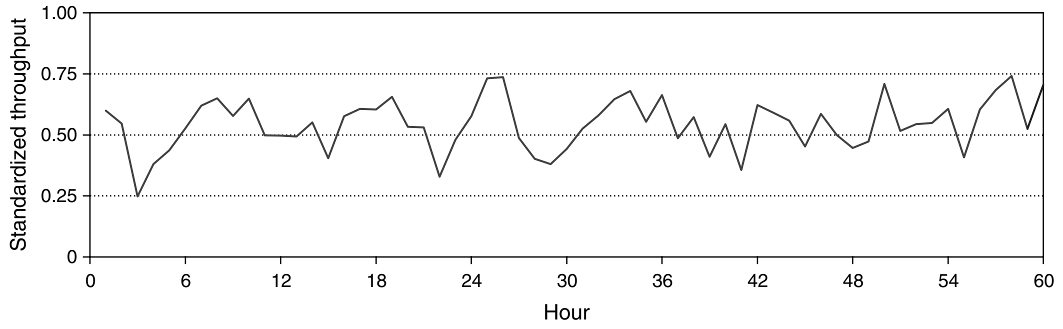

Figure B.1: The graph shows average hourly throughput for the first 60 hours of the simulation.

block; t_1 represents the time before the next failure and is drawn from an exponential distribution with a parameter of MTBF.

2. In the second block, the failure entity occupies the resource and puts it into the failed state.

3. In the next block, the failure signal is turned on for the resource just failed. The time that passes between the failure signal and the operator response is represented in Figure A.1 as t_{2a} seconds. However, t_{2a} is not drawn from any distribution because it is a logical delay that depends on the state of the whole system at that moment. If the operator responsible for repairing the machine that just failed is busy repairing another machine and other machines are waiting to be repaired by the same operator, then this time might be quite long. For t_{2a} seconds, the machine stays in the failed state because repair has not started. The failure entity that controls this routine's logic stays in this block during this time.

4. When the operator arrives, repair starts and the failure entity passes to the next block, in which it is delayed for t_{2b} seconds until the repair is completed.

5. After completion of the repair, the failure entity moves to the final block in the series; this puts the machine into the idle state. Finally, the entity goes back to the beginning of the routine to wait another t_1 seconds before failing the resource again.

Appendix B. Warm-Up Analysis

A practical and widely used method for finding a good warm-up period is mean plot analysis (Banks et al. 2005). The idea is simple: estimate the transient mean of the process by averaging across a number of replications; then choose as the deletion point a time after which the plot varies consistently around a fixed value. We explain how we used this method in our fuel injector simulation below.

First, we ran 20 replications of our base-case model, each replication of 10 simulated days, and recorded the number of fuel injectors produced in each one-hour interval for each replication. This gave us production numbers for 240 time intervals per replication. Then, we averaged the 20 values from each time interval separately and plotted them as a function of time interval. We choose as the deletion point the interval at which this plot no longer has a trend. There is only a weak upward trend between 3 to 9 hours, so deleting 24 hours is more than adequate.

Figure B.1 shows the first 60 hours of this plot (with the throughput normalized to an arbitrary constant unrelated to the actual value). Because we preload each process segment with injectors and all machines are up and raw material buffers full, the throughput quickly reaches steady state; thus, a 24-hour warm-up period is certainly sufficient.

Acknowledgments
This paper was adapted from Tongarlak et al. (2008). We thank the editors and referees for their helpful guidance.

References

Banks, J., J. S. Carson, B. L. Nelson, D. Nicol. 2005. *Discrete-Event System Simulation*, 4th ed. Prentice Hall, Upper Saddle River, NJ.

Kelton, W. D., R. P. Sadowski, D. T. Sturrock. 2006. *Simulation with Arena*, 4th ed. McGraw-Hill, New York.

Tongarlak, M. H., B. Ankenman, B. Nelson, L. Borne, K. Wolfe. 2008. Using simulation early in the design of a fuel injector production line. S. J. Mason, R. R. Hill, L. Moench, O. Rose, eds. *Proc. 2008 Winter Simulation Conf., Miami*, 471–478.

John Wray, Delphi Communications, 3000 University Drive, Auburn Hills, MI 48326, writes: "On behalf of Delphi Corporation I am pleased to verify our use of the simulation analysis presented in 'Using Simulation Early in the Design of a Fuel Injector Production Line' by Tongarlak, Ankenman, Nelson, Borne, and Wolfe.

"The project described in the paper is an ongoing effort to design a production line for a new product at Delphi. The simulation analysis has provided and continues to provide valuable guidance for the layout, loading and staffing of this line, and in particular has allowed us to assess design concepts early in the design process."