Statistical Validation of Shipyard Scheduling Software

Oliver Dain Matthew Ginsberg Erin Keenan Iain Pardoe John Pyle
Tristan Smith Andrew Stoneman*

December 10, 2005

Abstract

The ARGOS scheduling system is designed to reduce labor costs in large facilities such as shipyards. Theoretical results suggest that ARGOS is capable of reducing total labor costs for a large ship construction project by between 7 and 14 percent. SimYard is a stochastic shipyard simulation tool that can model a wide variety of shipyard production conditions, including problems that arise and how the shipyard reacts to those problems. It was built to evaluate ARGOS under actual production conditions to validate labor savings estimates. This article describes the major statistical hurdles involved in the design and subsequent validation of SimYard. The main challenge is to take performance data describing how a particular shipyard performs and tune SimYard to behave accordingly; this requires a model of SimYard itself. This problem can be framed as one of multivariate calibration, which can then be tackled using inverse regression methods. Predictive sampling from the resulting model provides an appropriate adjustment for statistical uncertainty, and shows that ARGOS schedules can be expected to save between 7 and 13 percent on a large ship construction project, confirming the theoretical results.

Key Words: Calibration; inverse regression; predictive sampling; stochastic simulation.

*Corresponding author is Iain Pardoe who is Assistant Professor, Department of Decision Sciences, Charles H. Lundquist College of Business, 1208 University of Oregon, Eugene, OR 97403–1208 (E-mail: ipardoe@lcbmail.uoregon.edu). Matthew Ginsberg is Chief Technical Officer, and Oliver Dain, Erin Keenan, John Pyle, Tristan Smith, and Andrew Stoneman are Research Scientists, On Time Systems, Inc., 1850 Millrace Drive, Suite 1, Eugene, OR 97403 (E-mail: {odain,ginsberg,erin,john,tsmith,stoneman}@otsys.com). The authors thank David Etherington, R. Dennis Cook, and Christopher Bingham for helpful discussions.

1 Introduction

The United States Navy spends billions of dollars annually on construction, refit, and repair of ships and submarines. New construction is primarily contracted out to commercial shipyards, while refit and repair are handled using a combination of commercial and naval shipyards. The problem of developing schedules that make efficient use of resources is critical, both to control costs and to meet Navy needs within the available resources, be they time, dry-dock space, or increasingly scarce skilled manpower. Both construction and maintenance are extremely complex processes, involving thousands of activities with thousands of constraints among them. This complexity makes it virtually impossible for people, even augmented by the commercial scheduling systems available today, to construct good schedules.

The first attempt to optimize shipyard schedules was based on technology for reducing aircraft assembly time. However, during this project it became apparent that, in shipyards, the fundamental goal is not to finish projects faster, but to reduce labor costs. As a result, the ARGOS scheduling system (Dain et al., 2005) was developed to target these costs directly. While an existing schedule might require no welders one week and ten the next, an ARGOS schedule is more likely to need five welders each week. By rearranging activities to smooth out the labor requirements in this way, ARGOS is able to significantly reduce overtime and undertime costs, as well as reduce costs associated with work-force acquisition and reduction. Based on an annual shipbuilding budget of $10.4B (2005) and Government Accountability Office estimates that 41% of this money is expended on labor (1992), theoretical analysis suggests that ARGOS savings can be expected to be on the order of $400M annually.

Because ARGOS may be licensed to the Navy on a percentage-of-savings basis, it became necessary to accurately determine the actual savings resulting from implementation of ARGOS schedules. However, it was not clear how those savings could be distinguished from savings attributable to other shipyard process changes, nor how they could be separated from fairly standard cost overruns due to unrelated problems. While personnel at various levels, both in the Navy and in the commercial shipyards, agree that use of ARGOS is likely to result in cost savings, there is concern that the theoretical savings might not be fully realized in practice due to the fact that construction never proceeds exactly as planned (for example, some activities take longer than expected, and others must be postponed when required parts are not available). There is also the possibility that the overall nature of ARGOS schedules tends to make them overly sensitive to unexpected shipyard events in practice, and thus less likely to produce the theoretical savings (for example, it is possible that existing shipyard schedules, which tend to have most of the work scheduled as early as possible, can adapt more easily to issues such as tasks that take longer than expected).

A desire to address these concerns led to the development of SimYard, a simulated shipyard that stochastically models problems that are likely to occur and shipyard responses to those problems. In SimYard, single hulls (or sets of hulls) can be "built" many times, allowing savings to be evaluated accurately and in a context independent of changes to other shipyard practices. SimYard confirms that most, but not all, of the theoretical savings calculated by ARGOS should be realized in practice, with predicted labor savings of approximately 10% on the construction of a single large hull. Further, these savings appear to be nearly independent of a wide variety of shipyard conditions.

However, at some level, the use of SimYard to evaluate ARGOS only moves the software validation question, as opposed to addressing it completely. It is true that SimYard appears to show ARGOS to be valid, but what shows SimYard itself to be valid? The purpose of this article is to describe the major statistical challenges involved with assessing this validity. Section 2 outlines how SimYard simulates the operation of a shipyard given a particular schedule. Section 3 discusses how SimYard itself can be statistically modeled to produce estimated inputs for a particular schedule (based on observed outcomes from real shipyard data), while incorporating appropriate stochastic uncertainty. Results are presented in Section 4, while Section 5 contains a discussion.

2 Shipyard scheduling and simulation

The simulator in SimYard takes a given shipyard schedule and a set of inputs, X. It then steps through a project while simulating changes that arise and shipyard responses to those changes. For example, on the day an activity is started, it might be discovered that more workers are needed than were anticipated; this might result in other activities being delayed. Each simulation produces corresponding outputs, Y.

The inputs, $X_1, \ldots, X_{42}$, describe how the shipyard operates and consist of 42 variables such as how often tasks tend to get delayed. The outputs, $Y_1, \ldots, Y_{20}$, measure the outcome of a particular project and consist of 20 variables such as how often constraints were broken. Labor costs incurred during a simulated shipyard project can then be calculated from the simulated outputs—these costs include work-force acquisition and reduction costs, as well as overtime and undertime. Thus, the SimYard simulator can map any schedule and set of inputs to a final cost estimate. Running the simulator many times provides a range of costs that reflects the stochastic uncertainty of SimYard itself (and which mimics real uncertainties in shipyard projects).

The simulator consists of three main components:

- *Reality* models how things change under real-world, yard-dependent conditions. This component

makes changes to the project based on the input variables. For example, there are inputs that determine the chance that activities will take longer than expected or require more workers than expected. Other inputs influence unexpected delays and staffing changes (due to workers who are sick or who are assigned to an unexpected project).

- The virtual *Floor Manager* adapts the schedule in response to *Reality*, deciding which tasks to work on and which to put off. This component is intended to model the actual floor manager of a real shipyard. Therefore, the decisions made are based on the knowledge that a real manager would be expected to have (based on conversations with employees in various shipyards). When the *Floor Manager* sees the work grow, it considers several responses. First, it checks if there are already people on staff to handle the extra work; if so, it may decide no response is necessary. On the other hand, if there is a shortage, the *Floor Manager* considers rescheduling work and/or assigning overtime. It has similar choices when the amount of work shrinks or other surprises occur.
- The virtual *Personnel Manager* reacts to the overall changes from *Reality* and from the *Floor Manager*'s responses to those changes by revising staffing decisions (work-force acquisition and reduction).

The SimYard simulator differs from other recent shipyard simulation projects (for example, McLean and Shao, 2001; Williams et al., 2001; Asok and Aoyama, 2005) in three important respects. First, these other studies simulate at a more fine-grained level (including physical constraints such as crane alignment and shop-floor space), but at the cost of simulating entire shipyards for only very limited periods of time. Second, previous work has had only limited success in simulating behavior that matches observed real-world behavior; this is a major challenge that we address in this article. Finally, none of these other projects includes the ability to adapt to real-world changes, as we do with the virtual *Floor Manager* and *Personnel Manager*; we believe these components are crucial, given our observations of actual shipyards.

3 Validation challenges

The biggest challenge in developing SimYard is "tuning" it by selecting appropriate sets of inputs so that its behavior matches that of an actual production shipyard. The problem is that many input parameters cannot be obtained from existing data. For example, if a task requires more work hours than expected, the *Floor Manager* could assign overtime, rearrange other tasks, or postpone the task in question. The likelihood of each decision depends in turn on other factors; postponing the task, for example, may depend on how willing the *Floor Manager* is to break deadlines or constraints with other tasks. Quantifying, *a priori*, the input

parameters that determine this behavior (for example, the cost of missed deadlines) is almost impossible.

Therefore, given performance data for a real project that has been completed, the goal is to create sets of inputs for which SimYard's behavior on the project would be similar to that observed in the performance data. One way to approach this is to try to model the relationship between inputs and outputs in SimYard. However, estimating such a model and then finding appropriate inputs is not straightforward for several reasons:

- Since the SimYard simulations are complex and stochastic, it is impossible to calculate exact relationships between the inputs and outputs from a randomly generated sample of SimYard runs.
- The relatively large number of inputs (42) and outputs (20) suggests that, practically speaking, any model relating inputs and outputs will need to have a linear structure.
- In a linear system with more inputs than outputs, there is a range of input values that leads to identical measured outputs. Further, costs may vary with the inputs even when other observables remain constant. Thus, it is necessary to accurately characterize the portion of the input space that yields the desired outputs, so that SimYard can be run on many input values in this space to produce accurate cost estimates with appropriate measures of uncertainty.

These restrictions suggest a statistical modeling approach. In particular, the challenge lies in building a suitable statistical model that relates inputs and outputs, so that an appropriate set of inputs can be generated (sampled) for any given set of outputs (i.e., real ship-yard data). The model needs to be stochastic rather than deterministic to account for modeling uncertainty (the model can only hope to approximate SimYard relationships, not replicate them exactly).

A multivariate regression model with outputs as the dependent variables and inputs as the independent variables might seem a natural way to approach the problem, but runs into difficulties because there are many more inputs than outputs. For example, solving 20 linear equations (one for each output) for 42 unknowns (the inputs) leads to a high-dimensional space of possible solutions. Sampling correctly from this solution space is not straightforward. Further, based on initial work using this approach, the solution space often leads to implausible ranges for the input variables. An alternative approach seems prudent.

3.1 Calibration and inverse regression

If, in tuning SimYard, both inputs and outputs can be considered stochastic (a reasonable assumption here), then inference about the joint probability distribution of inputs and outputs can also be based on multivariate "inverse regressions" of inputs (dependent variables) on outputs (independent variables). This approach is particularly advantageous in this context, because it is straightforward to calculate expected input values

from observed output values using the estimated (inverse) regression results. Uncertainty is accounted for using standard formulas for prediction (confidence) regions. Also, having the number of inputs exceed the number of outputs does not lead to problems with this approach since the prediction region formulas automatically handle this.

Thus, we used an inverse regression approach to tune SimYard. In particular, we first sampled a large number (2000) of sets of inputs from appropriate distributions. To allay concerns over biasing the simulations, these were selected to be uniform over reasonable, meaningful ranges for each input variable (some inputs were also transformed if, for example, they were constrained in reality to be positive). We also experimented with Latin hypercube sampling (McKay et al., 1979) to sample the inputs, but this made little difference to our results or computing times. SimYard then produced a set of outputs for each set of inputs based on multiple stochastic simulations. The entire collection of simulated inputs and outputs was then modeled using inverse regressions of the input values on the output values.

Once these models were estimated, we generated a large number (1000) of sets of "likely inputs" by sampling from predictive distributions for the inputs based on observed outputs from the real shipyard in question (see Section 3.2 below). Finally, these sets of likely inputs were then run through SimYard itself to generate "likely outputs" for a particular schedule, and hence "likely costs." It is then straightforward to summarize the distribution of cost estimates with density estimates or confidence curves—see Section 4. The numbers of tuning samples (2000) and predictive distribution samples (1000) were selected to balance minimizing of sampling errors within a reasonable computing time (which still ended up being a few days, even with a small cluster of fairly high-powered computers).

This set-up—and the goal of identifying which inputs are most likely to have produced some actual observed outputs—bears some similarities to multivariate calibration problems. There, the "inputs" might be "gold standard" (hence expensive) measurements of some kind and the "outputs" are inexpensive alternative measurements. The goal is then to build inverse regression models relating the gold standard measurements to the inexpensive alternative measurements, so that future alternative measurements can be accurately recalibrated. Brown (1982) reviews the use of inverse regression in this context.

To understand this approach, consider a hypothetical example where there are six inputs, $(X_1, \ldots, X_6)$, and three outputs, (Y_1, Y_2, Y_3), related as follows:

$$\begin{aligned} Y_1 &= X_1 + X_2 + e \\ Y_2 &= X_3 + e \\ Y_3 &= X_4 + e \end{aligned}$$

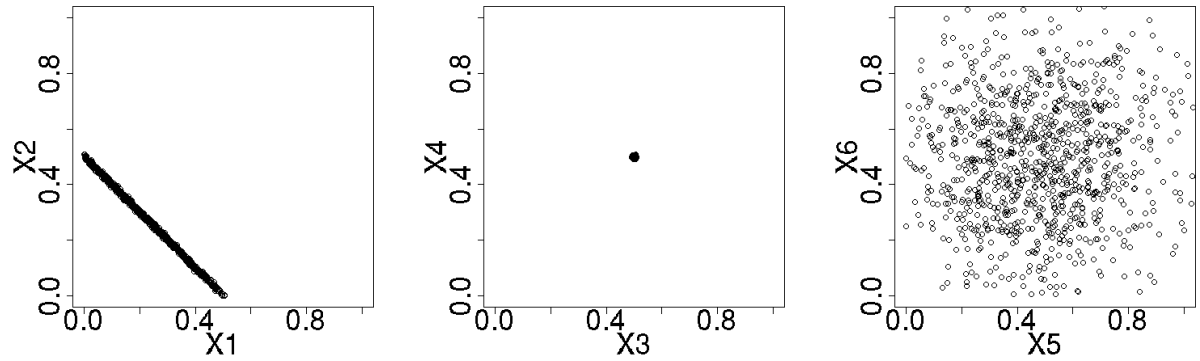

Figure 1: Scatterplots for the hypothetical six input/three output experiment showing X_1 vs. X_2, X_3 vs. X_4, and X_5 vs. X_6, respectively.

Here, e represents 1% random error. Note that two inputs, X_5 and X_6, do not affect any outputs.

It is straightforward to generate a large number of observations from this set-up, with the inputs ranging uniformly over [0,1]. The resulting input and output values can then be used to estimate inverse regression models of X on Y. Finally, given a particular set of observed output values, say $Y_1 = Y_2 = Y_3 = 0.5$, the estimated inverse regression models can be used to generate a large number of appropriate "likely" input values (draws from the predictive distribution—see Section 3.2).

We ran this experiment in R statistical software (R Development Core Team, 2005) to produce the scatterplots in Figure 1. The generated values of X_1 and X_2 in the left-hand plot show that they satisfy the first linear equation, $X_1 + X_2 = 0.5$ (with a small amount of variation due to error). The values of X_3 and X_4 in the center plot show that they correctly cluster tightly around 0.5 (which is required to satisfy the second and third linear equations). Finally, since X_5 and X_6 do not affect the outputs, the right-hand plot correctly shows sampling approximately uniformly over the range [0,1].

3.2 Predictive sampling

While it is straightforward to calculate expected input values from observed output values using the estimated (inverse) regression results, it is not immediately obvious how to generate a set of possible input values from an appropriate prediction region. The multivariate probability distribution required for drawing these predictive samples, a matrix-t distribution (see Keyes and Levey, 1996; Raiffa and Schlaifer, 2000, pp. 256), is non-standard and reasonably difficult and computer-intensive to sample from. Due to the relatively large size of the simulations, we therefore used a multivariate normal distribution (which is easy to sample from) as an approximation here. Since the number of input/output observations used to tune Sim-

Yard and estimate the inverse regression models is 2000, this would seem to be a reasonable approximation. Further investigation suggested that generated values from a matrix-t distribution using a sample size of 2000 do indeed appear to be "normal" when graphed on QQ-plots (normal probability plots).

However, the QQ-plot is just a univariate graphical diagnostic for normality. Multivariate normality is a little more difficult to show (since, as is well known, a set of variables may each be distributed as univariate normals, but their multivariate distribution may not be multivariate normal). We therefore conducted a stronger test for whether the multivariate normal approximation to the matrix-t distribution is valid here. In particular, we ran simulations in R (outside of SimYard) to check that the nominal coverage of the multivariate prediction region based on predictive samples using the multivariate normal approximation is what it is supposed to be. The relevant prediction region equation can be found in any standard multivariate analysis textbook (for example, equation 7-48 on p. 396 of Johnson and Wichern 2002). We then used the set of input values generated using the multivariate normal approximation, and calculated the proportion of values inside the 95% region (say) using this equation. If the multivariate normal distribution does provide an adequate approximation to the matrix-t distribution (in this context), then this proportion should be close to 95% (subject to sampling variation). We repeated this process a few hundred times to confirm that the proportion does indeed remain reasonably close to 95% (it ranged from 94.8% to 97.6% for a simulation with 42 inputs and 20 outputs, 2000 tuning samples, and 1000 predictive distribution samples).

3.3 Further validation

We considered a further confirmatory method to validate that the inverse regression models were correctly producing appropriate input values. We randomly selected SimYard output values, and checked if SimYard simulations based on input values generated from the corresponding inverse regression models actually produced output values similar to those initially selected. In particular:

1. We ran 1000 SimYard simulations using the default uniform input ranges to generate corresponding outputs, and then estimated the inverse regression models (based on the 1000 sets of inputs/outputs).
2. We selected one of the 1000 runs at random and used the output values produced by that run as assumed "real" performance data.
3. We used the predictive sampling (of Section 3.2) to produce 100 sets of "likely" input values that could have produced the selected set of "real" output values.
4. We ran a SimYard simulation for each of these 100 sets of input values to produce 100 corresponding sets of output values.

5. Finally, we looked at these 100 sets of output values and compared them to the original randomly selected set of "real" output values. In particular, for each output, we calculated the percentile of the SimYard simulation values in which the randomly chosen "real" value was found.
6. We repeated steps 1–5 100 times to obtain an estimate of the distribution of percentiles (from step 5) for each output.

Since SimYard is stochastic, any particular output value will differ from its predicted value based on the inverse regression models. If it is close to its predicted value, then the percentile in step 5 should be close to 0.5. If it is far from its predicted value (i.e., it is somewhat of an outlier), then the percentile might be closer to 0 or 1. However, on average, repeating this experiment should—if SimYard is not producing biased output values—result in percentiles centered around 0.5 (although, as just noted, individual values might be as extreme as 0 or 1).

Since there are 20 outputs, there is a good chance that with any randomly chosen set of output values, a few could be relative outliers. We observed this in the above experiment; each experimental run gave rise to different outputs with extreme percentiles, but with some runs having no outputs with extreme percentiles. Overall however, no individual outputs exhibited a tendency to consistently produce extreme percentiles. For example, Figure 2 shows a histogram of the percentiles produced for one particular output. This exhibits the desired behavior, being centered at 0.5 with no particular tendency to always produce extreme percentiles close to 0 or 1 (such a tendency would correspond to a bias on the part of SimYard for the corresponding output). Other outputs produced broadly similar results, with none showing signs of severe bias.

4 Results—estimated cost savings

The overall goal of this project was to predict the likely cost savings of using ARGOS in a commercial shipyard production environment. In this section, we compare the simulated costs of two schedules for a single large hull, consisting of over 7000 activities spanning 7 years:

1. A standard non-ARGOS schedule for a commercial shipyard, produced by the shipyard's project management system in concert with a considerable amount of human tuning.
2. An ARGOS schedule, optimized to minimize cost.

For each schedule, we generated 1000 sets of outputs and associated costs, as detailed in Section 3.1. Figure 3 shows the cost savings of the ARGOS schedule relative to the standard schedule. Over all the simulations done, the ARGOS schedule saves between 4% and 17% of the labor costs, with most runs saving between 7% and 13%. Figure 4 shows a savings confidence plot for the ARGOS schedule relative to the

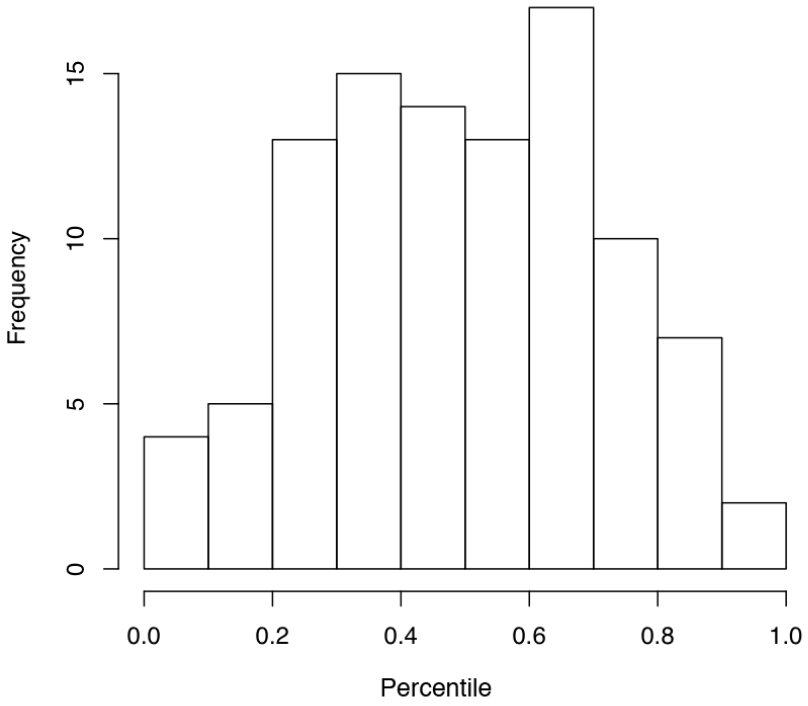

Figure 2: Histogram of percentiles for a randomly selected output value with respect to SimYard simulations of that particular output.

standard schedule. The ARGOS schedule saves around 10% or more half the time and saves 7% or more 90% of the time.

In addition to comparing the savings of one schedule to another, the simulations described in this article enable consideration of how those savings vary as input values change. For example, Figure 5 shows the savings sensitivity of the ARGOS schedule relative to the standard schedule based on the input that measures the average change in manpower requirements from unexpected shipyard events. As tasks tend to require more work than expected, the savings of the ARGOS schedule go down, presumably because the resulting uncertainty is forcing SimYard to depend more on the floor manager and less on the details of the original schedule. Figure 6 shows a similar trend for the input that measures the variability of task durations. As task durations get more and more unpredictable, the ARGOS schedule again tends to save less money. However, overall, the savings achieved by the ARGOS schedule appear to be relatively robust to changes in input values. For example, Figure 7 shows that ARGOS savings depend little on the fraction of tasks that get unexpectedly delayed.

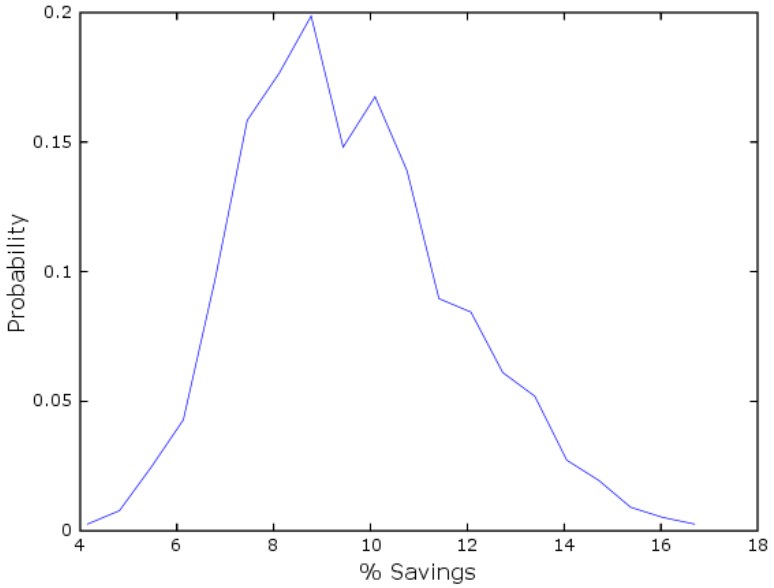

Figure 3: Percentage cost savings of the ARGOS schedule versus the standard schedule. The curve represents a probability density function estimate based on 1000 observations.

5 Discussion

In this article, we have described the SimYard system and the statistical approaches that underly its validation. SimYard was designed to simulate shipyard operations under a variety of input conditions. It can be used to compare the labor costs incurred by various schedules and to validate theoretical cost savings of ARGOS schedules relative to standard schedules.

The main statistical challenge in designing SimYard was to generate input values (especially those that are almost impossible to obtain *a priori*) that are likely to correspond to observed outputs from real shipyard projects. Because there are more inputs than outputs, a standard multivariate regression approach does not work well. As an alternative, using ideas from multivariate calibration, we used multivariate inverse regressions of inputs on outputs to infer the joint probability distribution of inputs and outputs. We described a simple example to illustrate how this approach works for this application, and also outlined some experiments that validate the use of predictive sampling in this context. We then presented actual SimYard results for a large shipyard project based on these statistical techniques.

One factor adding to the complexity of statistical validation for this application was dealing with the large number of inputs (42) and outputs (20). We investigated the use of dimension reduction techniques such as principal components (see Dunteman, 1989) to simplify modeling of the inputs and outputs. Since we

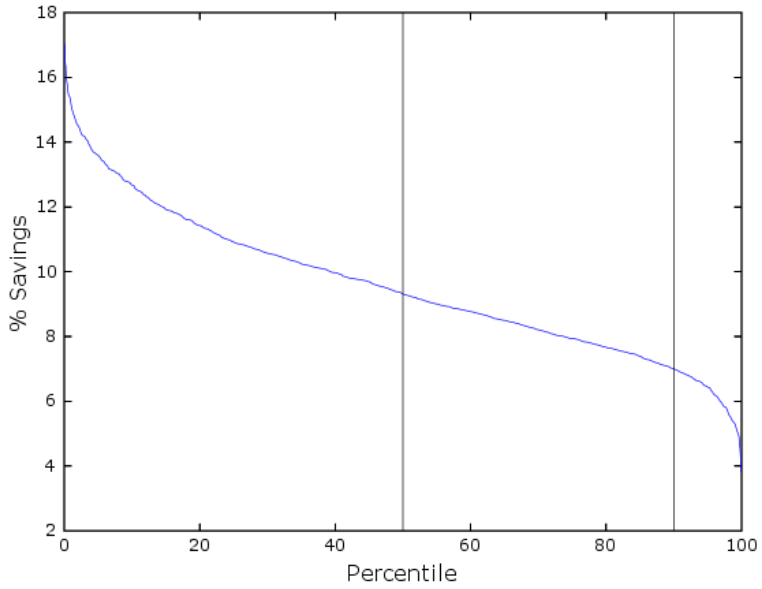

Figure 4: Cost savings confidence of the ARGOS schedule versus the standard schedule. The curve represents the expected percentage savings as a function of the confidence percentile on the horizontal axis. The two horizontal lines correspond to the 50^{th} and 90^{th} percentiles.

randomly generated input values to tune SimYard, dimension reduction of the input space would probably not be very meaningful, but dimension reduction of the output space perhaps offers more possibilities for progress. However, since SimYard works only with the full set of inputs and outputs, complications ensue when mapping between all outputs in SimYard and a reduced set of output variates (say) in a statistical model of SimYard.

Although SimYard was built primarily to validate theoretical cost savings of ARGOS schedules, it has a number of other possible uses by the Navy and commercial shipyards. In particular, SimYard could be used to:

- synthesize shipyard performance data and answer questions such as “how often were tasks paused?”;
- analyze input/output dependencies to answer questions such as “which factors influence the amount by which deadlines are broken?”;
- investigate hypotheses concerning shipyard behavior (for example, SimYard could help understand whether shipyard behavior is different in the first half of a project than in the second half);
- compare different shipyards based on performance data; and
- help understand the impact of additional work (for example, if a shipyard is considering a new project,

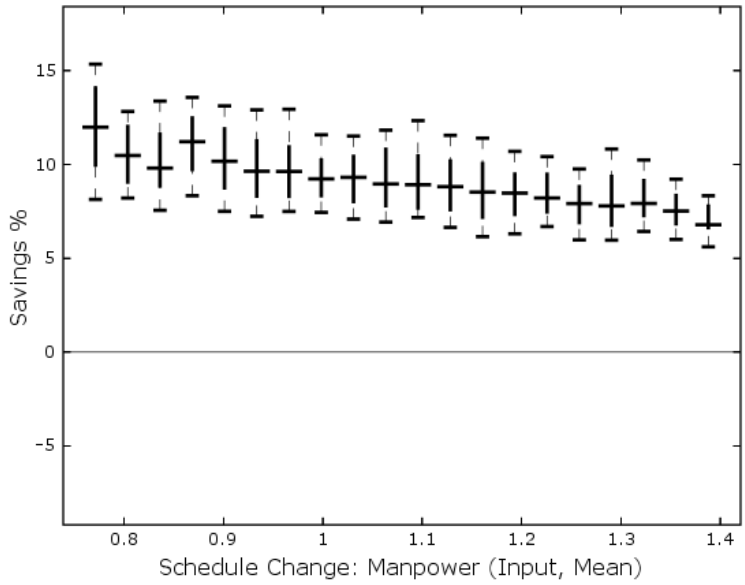

Figure 5: Savings sensitivity for the input that measures the average change in manpower requirements from unexpected shipyard events. The boxplots represent medians, upper and lower quartiles, and 10^{th} and 90^{th} percentiles for the generated savings estimates.

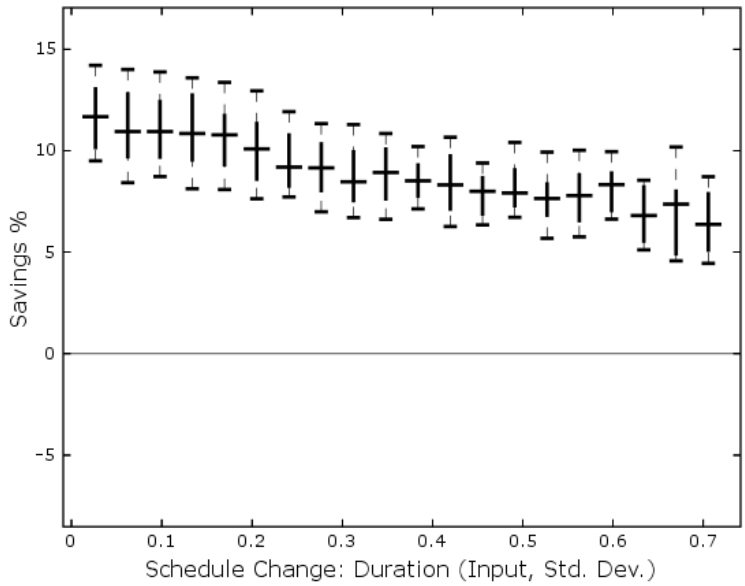

Figure 6: Savings sensitivity for the input that measures the variability of task durations.

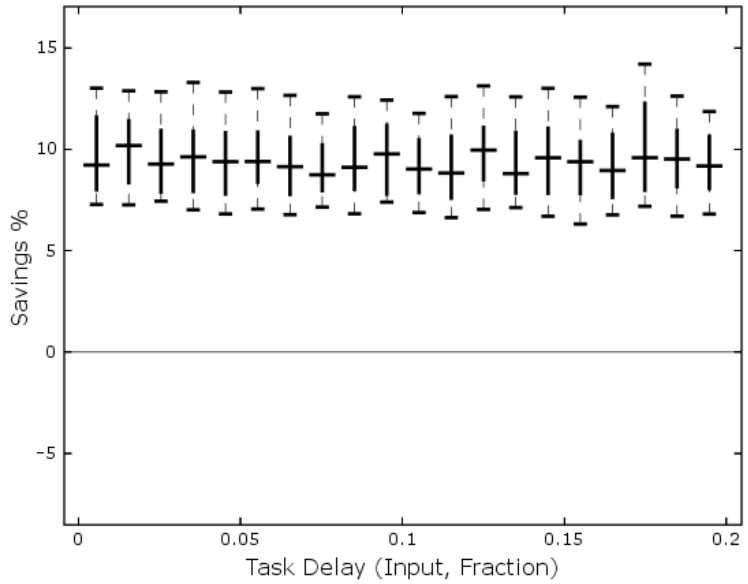

Figure 7: Savings sensitivity for the input that measures the fraction of tasks that get unexpectedly delayed.

SimYard could be used to predict the potential cost impact and disruption likely to result from that project, and the most appropriate time-frame for that project).

Therefore, SimYard has the potential to be a hugely important tool for shipyards. This potential rests heavily on whether cost savings estimates from SimYard can be considered accurate and reliable. The approaches to validating SimYard described in this article go some way to demonstrating this to be the case.

References

Asok, K. A. and K. Aoyama (2005). Risk management in modular ship hull construction considering indefinite nature of tasks. In *Proceedings of the 12th International Conference on Computer Applications in Shipbuilding*, Busan, Korea.

Brown, P. J. (1982). Multivariate calibration (with discussion). *Journal of the Royal Statistical Society, Series B (Methodological) 44*, 287–321.

Dain, O. M., D. W. Etherington, M. L. Ginsberg, E. O. Keenan, and T. B. Smith (2005). Automated scheduling to minimize shipbuilding cost. In *Proceedings of the 12th International Conference on Computer Applications in Shipbuilding*, Busan, Korea.

Dunteman, G. H. (1989). *Principal Components Analysis*. Quantitative Applications in the Social Sciences. Newbury Park, CA: Sage Publications.

Johnson, R. A. and D. W. Wichern (2002). *Applied Multivariate Statistical Analysis* (5th ed.). Upper Saddle River, NJ: Prentice Hall.

Keyes, T. K. and M. S. Levey (1996). Goodness of prediction fit for multivariate linear models. *Journal of the American Statistical Association 91*, 191–197.

McKay, M. D., W. J. Conover, and R. J. Beckman (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 239–245.

McLean, C. and G. Shao (2001). Simulation of shipbuilding operations. In B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer (Eds.), *Proceedings of 2001 Winter Simulation Conference*, Arlington, pp. 870–876. The Society for Computer Simulation International.

R Development Core Team (2005). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Raiffa, H. and R. Schlaifer (2000). *Applied Statistical Decision Theory*. New York: Wiley.

Williams, D. L., D. A. Finke, D. J. Medeiros, and M. T. Traband (2001). Discrete simulation development for a proposed shipyard steel processing facility. In B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer (Eds.), *Proceedings of the 2001 Winter Simulation Conference*, Arlington, pp. 882–887. The Society for Computer Simulation International.