Asynchronous Stochastic Learning Curve Effects in a Large Scale Production System

Roberto Francisco-Yi Lu

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington
2008

Program Authorized to Offer Degree:
Industrial Engineering

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

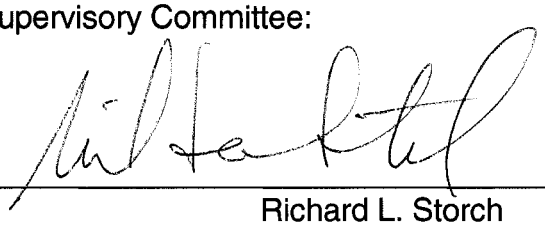
University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Roberto Francisco-Yi Lu

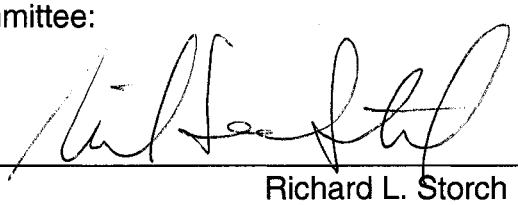
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

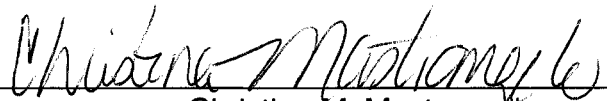


Richard L. Storch

Reading Committee:



Richard L. Storch



Christina M. Mastrangelo

Thomas G. Schmitt

Date: March 19, 2008

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Roberto F. Lu

Date March 19, 2008

University of Washington

Abstract

Asynchronous Stochastic Learning Curve Effects in
a Large Scale Production System

Roberto Francisco-Yi Lu

Chairperson of the Supervisory Committee:
Professor Richard L. Storch
Industrial Engineering

Large scale production systems contain dynamically unequal distributing lead times and improvement rates. This has made it very problematic to statistically predict the ability of global partners to meet common deadlines simultaneously. This system behavior can be attributed to the effects of asynchronous stochastic learning curves (ASLC) among all partners. There is no known prior research that provides an analytical model of the synchronized performance among partners in a large scale production system (LSPS) with these ASLC effects.

In large scale production systems, integration of complicated products such as commercial airplanes brings unique concerns. These concerns involve the synchronized integration of customized products amongst processes of different lead times and stochastic learning curves. The source of the processes may include the system integrator, partners, suppliers, product designers, and custom feature configurers.

Determining statistically predictable performance based on the roles and influence of all participants across different domains of production systems has been more of an art than a science. The fact that people improve efficiencies when doing the same task repeatedly was addressed as the learning curve effect, which was first studied in the aerospace industry in the 1930s (Wright 1936). The larger the integrated production system, the more complicated it becomes to customize product design and deployment. Challenges in

unstable individual production starting times have significant impacts on the productivity of the overall system. Other challenges in modeling this system are associated with the speed of process improvements, unequal lead times, and production rate fluctuations. Low volume and highly customized products of a large scale production system have not been previously analyzed through the ASLC approach combined with dynamic interactive discrete event simulation modeling.

This dissertation addresses research and findings associated with customized products during the early product inception stages in large scale production system integration with the ASLC effects. Data is collected and partially derived through an ASLC model with statistical linear regression analyses and augmented with some discrete event simulation modeling.

Deliverables of this research will provide a theoretical definition and practical model of the ASLC, and contributions in the following areas: ASLC impacts in large scale production system integration, discrete event simulation modeling methods in LSPS integration, and statistical presentation of ASLC in LSPS.

This research will further enhance analyses of LSPS in the following: mass customization influences, production rate change impacts to all parties in the supply chain, logistics in transporting large size items globally, customized product design and production considerations, customer ordering points for fixed and for customized configurations, impacts of unexpected events and change orders in LSPS, postponement of engineering-to-order decision points, and production balance between existing learning curves and increasing planned/unplanned changes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to acknowledge his employer, The Boeing Company, for offering the industry leading "Learning Together Program" that supported tuitions and most fees associated with the author's PhD course at the University of Washington. The author wishes to acknowledge a few key and generous leaders in The Boeing Company: Wayne Hixson, Mahender Reddy, Brad Zaback, Pete Lohr, Rich Ptacin, Rita Camarillo, Richard Butcher, J. Paul Russell, Steve Przybelinski, Craig McCormick, Jerry Marsh, and Al Miller. These genuine leaders enabled the possibility of this dissertation.

The author wishes to acknowledge faculties and staff members of the Industrial Engineering program at the University of Washington. In every possible way they have accommodated the authors work schedules and family matters throughout his graduate study at the UW. They have always been there to help and to advise the author. The author wishes to acknowledge their unparalleled passion in higher education that carried the author through his PhD journey.

DEDICATION

The author dedicates this dissertation to his family members.

To his father. His father has been a strong inspiration and source of encouragement in many key occasions throughout the author's higher education. It was his father over the phone when the author drove home from the University of Washington at 3 am over the 520 bridge with high winds blowing water from Lake Washington all over the bridge and his vehicle. It was his father who candidly analyzed and provided loving solutions in stressful situations among the author's work place, family, and UW level of rigorous graduate studies. It was his father who kept asking annoying and demanding questions about the author's progress on page and chapter numbers. The author did not start the PhD journey to delight his father, yet the author may complete the PhD because of the abundant and persistent interest from his father. Therefore, this dissertation is dedicated to his father.

To his mother. His mother has been the best mother one can ever imagine. It has been the author's mother who provided all the tender loving reminders to the author during those 15 credit-hour quarters while working overtime as a full time Boeing employee. It has been his mother who kept the author on track in maintaining his health and family harmony. Therefore, this dissertation is dedicated to his mother.

To his wife. The author's wife has been the single most important and influential person in his educational journey in all graduate studies, especially in this PhD journey. The author's wife gives him a worry-free home, healthy, happy children, and delicious home cooked meals as often as possible (given the author has a record of 18 consecutive meals on his vehicle while driving among work places and UW). As a professional certified public accountant (CPA), she has always managed the author's home excellently in all aspects. She has enabled the author to go full speed ahead without any worries and constraints. She is the person the author loves the most, the friend the author shares stories with the most, and the wife the author appreciates and treasures the most. Therefore, this dissertation is dedicated to his wife.

To his children. The author's three beautiful and good daughters have lived the equally demanding journey during the past many years. They did not have all the weekends to play with their daddy. They missed many nights of stories from their daddy. They stayed strong and continued their educations mostly with the help from the author's wife. The author is very proud of them; one is now enrolled in a reputable university with scholarship and the other two are in gifted programs. Therefore, this dissertation is also dedicated to his daughters.

The author dedicates this dissertation to his family members – his father, his mother, his wife, and his daughters.

Chapter 1: Introduction

Large scale production systems are seen in some of the ship building, high-rise apartment complex building, and aerospace industries. There are several common characteristics in such a system among these industries. The total production volume throughout the life cycle of a product is limited, maybe in the range from fewer than one hundred to approximately one thousand units. Every component can be significantly important in a large scale production system from the first article to the end of the product life. In the aerospace industry, test units are carefully planned and produced just as regular production articles. This type of system may never achieve a steady state mass production status. The rate of improvement from the first component to the one that is at the break-even point of the product development cost is critical to business performances in these industries.

Different vendors produce most major components, such as a barrel section or wings of an airplane, engines of an ocean vessel, an apartment insert of a high-rise building. Component production starting times may vary (stochastic) due to differences among components and lead-time variations (asynchronous) among component producers. For matching components, their arrival times are desired to be synchronized. Individual cycle times of different components may improve as more units are produced by different learning curve attributes. Therefore, the asynchronous stochastic learning curve (ASLC) effects exist in a large scale production system.

1.1 Contribution of the Research

This dissertation contributes to the literature in these following novel items:

- The introduction of the ASLC concept
- The ASLC framework
- Mathematical presentation of the ASLC model
- A statistical method of the ASLC model
- A discrete event simulation method of the ASLC model
- A verification method between the statistical and the simulation methods
- Applications of the ASLC model in a large scale production system

Significance of above contributions is discussed in the next section.

1.2 Significance of the Research

Wright introduced the learning curve concept in 1936. Since then, there have been many different studies about the learning curve theory with various mathematical presentations. None of the prior learning curve related research addressed the stochastic nature of asynchronous processes in a large scale production system. For a single production line shop, all processes associated with the production line can be presented by one learning curve. When there are multiple processes occurring with different starting times and not necessarily the same ending time along with different learning curve rates, then there are ASLCs in the system. Many industries have ASLC effects in their production system. The introduction of the ASLC concept is significant in elevating the conceptual awareness of the realistic view of production systems in the global business environment.

The ASLC framework is significant because it depicts the ASLC concept for both theoretical mathematical presentation and the practical application of the model. The ASLC framework is the bridge between the ASLC concept and the mathematical presentation of the ASLC model. This framework was never introduced before, and yet it is applicable to industries such as high-rise apartment construction, shipbuilding, and airplane production. The ASLC framework will be illustrated and discussed in more detail in a later section.

The ASLC concept contribution to the literature would not be as significant without a mathematical presentation of the model. The mathematical formulation of variables in the ASLC model enables the statistical method in testing ASLC related processes. Previous research outlined learning curves in many meaningful manners for single production stream processes. The mathematic model further enables researchers and practitioners to advance literacy in ASLC.

The statistical and the discrete event methods of the ASLC model enable detailed analyses and forecasts of ASLC effects in a large scale production system. These two methods contribute to not only the ability to analyze historical ASLC data, but also statistically predict future performances of ASLC effects in a large scale production

system. Hence, a verification method between these two methods is important in providing contribution to those who exercise the ASLC model.

The contribution of this research is further illuminated by its readily deployable practical application in any large scale production system. This research outlines the framework of the ASLC model, followed by the mathematical presentation and two verified applicable methods. The discrete event simulation method in depicting the ASLC is a significant application to industries such as mass customized high rise apartment complexes, engineering-to-order ship building in shipyards, and commercial airplane manufacturing.

1.3 General Introduction of the Research

In large production system integration of complicated customized products, suppliers have been involved at various levels ever since the inception of mass production in the early 1900s. Determining statistically predictable performance based on the roles and influence of supplier participation across different domains of production systems has been more art than science. Most modern manufacturing enterprises employ suppliers for activities that range from component fabrication to product design. Thus, multiple tiers of suppliers with different learning curve rates interact both vertically and horizontally with the final system integrators. The traditionally defined suppliers are more like partners in today's modern large scale production system.

Throughout eras of producing goods, many common manufacturing and fabrication approaches have been practiced among societies of different demands on levels of customization in their respective production system. The Industrial Revolution changed methods people used in producing goods in many societies during the late nineteenth century. Once hand crafted products, such as textiles, suddenly can be mass produced. Mass production started its glory days, thanks to Henry Ford's first ever moving line, in 1914 (Pope 2001). Production efficiency evolved from mass to lean, from US to Japan and then back to US (Womack et al. 1990) for almost nine decades – mostly about making a lot of the same product fast and affordable. Consumer desire and demand personalized products in recent decades promoted competition in the manufacturing industry to focus on the ability to flexibly and rapidly respond to changing market conditions (Qiao and Lu

2006). Manufacturers have found that they can no longer capture market share and gain higher profits by producing large volumes of a standard product for markets that demand customized products. Stanley M. Davis first expounded the concept of mass customization formally in the book "Future Perfect" in 1987 (Davis 1987). Joseph Pine (Pine et al. 1993) gave mass customization a clear definition as a strategy that sought to exploit the need to support greater product variety and individualization. Production systems for many types of products have shown an irreversible trend in the transition from mass production to mass customization, where unequal stochastic lead time combined with asynchronous learning curves take effect across the whole production system.

Configurations and ASLC performances among partners can be drastically different whether the final product will be mass-produced or mass-customized. Various product configurations can be demanded from different market segments. Considerations for customized personal products such as clothes and shoes, which can be manufactured in factories that do not require special infrastructure features, are not the same as those for large, integrated products such as ocean vessels and airplanes, which literally cannot be assembled without feasible infrastructure in place. This research focuses on large scale production and integration of products with ASLC effects in the system.

The level and depth of involvement among all partners in a given large scale production system is evolving to be more cohesive as the virtual distances around the world keep getting smaller. This research includes the development of the ASLC and modeling system event management in large scale production system. It utilizes both statistical and simulation methods to model events among suppliers, pre-integrators, and the final system integrator. Conceptual interim models contain assumed learning curves in place for each entity in the system. Based on the assumed "activity" levels, the "assumed" predictable statistical performance is determined using the learning curves. Different scenarios are simulated with different events, customer demands, configuration changes on different supplier schemes over different global transportation logistics for different final integration / pre-integration criteria and demands. Therefore, each entity in the system possesses a compounded improvement curve and event driven base performance. Such performance for each entity in the system changes dynamically from one to another set of statistical

distributions. Components and integrated products exist at different stages in the system dynamically and simultaneously as the simulation model runs against time. Thus, influences of various events can be modeled to simulate large production system integration.

Improvements by learning curves across the production system upon new product deployment are an important factor that can influence the overall system performance. Products such as airplanes and ocean vessels take a long time to design and develop. Each major component supplier has tasks in both individualized and integrated categories that demand different learning curve performances. Statistically predictable performances from various tiers of major component suppliers are modeled with a statistical method via linear regression.

It is highly desirable, and in many cases, required for suppliers to have plans and means of execution to manage various types of event changes across the supply chain. Event changes may come from but are not limited to these sources: product configuration, customer demand, and unexpected actions. Suppliers at different tiers, system pre-integrators, and the final system integrator have different challenges and risks associated with event changes. In large scale production system integration, it may not be financially favorable to realistically practice many of the unexpected events. However, it is totally feasible to apply statistically verified modern simulation techniques to model dynamic interactions among major entities in such a system.

Objects and modules in this series of simulation models represent individually grouped events. Event types can be a component fabrication process, a customization process, a business decision, a product integration process, etc. Event attributes contain time durations, different assignable statistical distributions, predefined process variables, predecessor successor hierarchies, decision criteria, and their learning curve attributes. These various defined events are actually meant to represent real life entities in various scenarios. Consequently, systemically organized objects and modules that encompass all of these dependent and independent discrete events can simulate impacts of ASLC in a large scale production system.

Case studies in this research explore issues from viewpoints among component ordering decision timing, inventory levels among partners, customer driven customization decision level, rate of change of the final product, interim product service rate and their respective due-date performance, logistics in the system, non-recurring design and investment impacts, and component manufacturer performance. Dynamic interactions among all entities are studied through exercising discrete event simulation modeling. This research is to depict a novel model of ASLC and then derive rational relations between ASLCs and overall system performance in large scale production system via verified statistical and discrete event simulation methods.

Chapter 2: Literature Review and Opportunities

For a single production line shop, all processes associated with the production line can be presented in one learning curve. When there are multiple processes occurring with different start times and not necessarily the same end time along with different learning curve rates, then asynchronous stochastic learning curves (ASLC) can be found in the system. There have been many different studies about learning curve theory with various mathematical presentations.

2.1 Previous Learning Curve Models

Wright first introduced the learning curve in 1936 and his log2 based model is still the best for predicting future process performances. His linear log x, log y learning curve model is:

$$\log y = a + b \log x \tag{1}$$

Thomas, Mathews, and Ward (1986) presented the cubic log x, log y learning curve model:

$$\log y = a + b(\log x) + c(\log x)^2 + d(\log x)^3 \tag{2}$$

Their model was believed to have the best fit over existing data.

Levin and Globerson (1993) compared the linear learning curve model against the power learning curve model for individual products generated from aggregated data. They did not claim any unique solution through their mathematical modeling exercises.

Norfleet (2005) studied the various causes of lost productivity which have resulted from disruptions to the planned workflow, or from disruptions in learning processes. He may have interpreted the learning curve in a very different way by using the percentage reductions from unit to unit instead of considering the nature of a log2-based curve.

Table 1 Previous Research on Learning Curves Models

Researchers	Years	Topics	Opportunities
Wright	1936	log 2 based learning curves	Asynchronous stochastic events in learning curves
Thomas, Mathews, and Ward	1986	Cubic learning curve	Ability to forecast system performance when ASLCs are in the system
Levin and Globerson	1993	Linear and power learning curve comparisons	Apply statistical and / or discrete event simulation methods to learning curve related modeling.
Norfleet	2005	Percentage reductions in learning curves that caused production disruptions	Using log 2 based learning curves to forecast system performance
Gu and Takahashi	2000	Average-case learning with ill-disposed learning algorithm	Distribution within each learning curve, and ASLC among partners
Wideman	1994	S-curve derived from a Log-Log learning curve in construction business for different stages of work	Asynchronous stochastic events in learning curves
Mosheiov and Sidney	2005	Polynomial time solution for a single machine scheduling problem to minimize the number of tardy jobs, using multiple integer programming in learning curves.	Multiple machines / processes in a system and log scale learning curves

Gu and Takahashi (2000) focused on the learning curves when the algorithm performs the best and the worst in average-case learning problems. They applied a so-called "ill-disposed" learning algorithm in terms of a system complexity. Their study led to a new understanding of the worst-case generalization in real learning situations.

Wideman (1994) used elements from the construction business process and suggested approximation profiles for both typical resource loading and progress S-curves, which was

derived from log-log based learning curves. His work showed that the process time difference could be due to the effects of learning. He applied the learning curve on a log-log scale with some rules of thumb relating to the learning curves. His S-curve depicts the cumulative total of on-going work on a construction job as plotted against time. He "force fit" the learning curve to different stages of work.

$$\frac{y_x}{y_X} = \frac{a \bullet x^{(b+1)}}{a \bullet X^{(b+1)}} \tag{3}$$

Where X is the point at which the S-curve joins the start of the next stage, and x < X.

Mosheiov and Sidney (2005) introduced a polynomial time solution for the single machine scheduling problem to minimize the number of tardy jobs with general non-increasing, job-dependent learning curves and a common due date. They applied multiple integer programming in their model without addressing multiple jobs having different processing times.

2.2 Previous Learning Curve Applications

Everett and Farghal (1994) evaluated mathematical models that describe the relationship between the activity time or cost and the cycle number. Their work showed that cubic learning curve models (Thomas, Mathews, and Ward 1986) generally give the highest correlation to completed repetitive activities. However, the cubic learning curve models were found to be poor predictors of future performance and should not be used to estimate performance beyond known historical data. Their paper indicated that the linear log x, log y learning curve model (Wright 1936) is the most reliable predictor of future performance. Pearson's coefficient of correlation, R2, was calculated to measure the correlation of each model to determine their performances. Their study utilized a dataset of as many as 60 samples in the residential construction related tasks.

Everett and Farghal (1997) fitted the same set of 54 data points using a few different methods to evaluate learning curves. The methods include unit data and cumulative-average data techniques, as well as the moving average and exponentially weighted

averaging techniques. The unit data technique was found to give the most accurate prediction of the time or cost to complete the remaining stages of the activity, whereas the cumulative-average data technique was found to give the least accurate prediction. They found that compared to the unit data technique, the exponentially weighted average could be applied later in the activity with equal predictive accuracy.

For cases when the objective function is the net present value, Anderson and Parker (2002) developed an engineering-based model of outsourcing by including the effects of learning over time on both the production of individual components and the integration of components into complete products.

Wu and Sun (2006) formulated a mixed nonlinear program for project scheduling and staff allocation problems with consideration of the learning effects on staff. Their objective was to minimize outsourcing costs. In their case study, their staff members were capable of handling different projects without additional learning effects, demonstrating a successful learning curve application based on Wright's model (Wright 1936).

Vits, Gelders, and Pintelon (2006) described a process change strategy to increase the effective capacity of a production system when unit costs are subject to a learning curve in the manufacturing departments of electronic firms. Leong and Cho (2006) applied the neural network-based, partial-least-squares methodology in feature selection that aims to discriminate between the process parameters that are relevant to the product quality and those that are not. Jaber and Guiffrida (2004) addressed the possibility that in many practical situations the process may go out of control, thus producing defective items that need to be reworked. Their paper considered the rework time per unit when measuring the learning curve. However, they did not address cases when defects were discarded, and they assumed the rate of defect generation was constant. Mosheiov and Sidney (2003) studied the case of job-dependent learning curves, with the objective of minimizing the 'make span' and total flow time on a single machine.

Biskup and Simons (2004) focused on the scheduling problem in which processing times decrease according to a learning rate, which can be influenced by an initial cost inducing

investment as well as the effects on autonomous learning-by-doing. Argote and Epple (1990) studied large increases in productivity that were typically realized as organizations gained experience in production. They identified reasons why organizational learning rates vary, such as organizational forgetting, employee turnover, knowledge transfer, and scales of the economy. Demeester and Qi (2005) studied how a firm could benefit from allocating its learning resources when it is concurrently producing two consecutive generations of one product with considerations of the learning rate, and the level of cross learning. Their results indicated that learning resources are best managed through a firm-wide coordination process that dynamically spreads learning resources throughout the oraginaization instead of concentrating them statically in a single area. This process ensures continued high returns from these learning resources.

Instead of assuming the same learning pattern for all employees, Shafer, Nembhard, and Uzumeri (2001) conducted a unique study in which a distribution of learning behavior based on an empirical population of workers was used. Their case study focused on an assembly line that produced car radios. Key factors considered by their study included the rate of worker learning, the size of the worker pool, task tenure, and the magnitude of worker forgetting in an independent work environment where workers work independently on one assembly line. Their counterintuitive result showed that the scenario with worker heterogeneity resulted in higher levels of output. Their deterministic simulation analysis was conducted by using 95% confidence intervals and an ANOVA table.

Teunter et al. (2004) proposed a class of inventory strategies for a single-item hybrid inventory system with unequal lead times. Their paper present comparisons of the optimal strategies in classes of standard push, lead-time-adjusted push, standard pull, lead-time-adjusted pull, and separate pull. They concluded that the "separate pull" performs well in almost all categories.

Table 2 Previous Learning Curve Applications

Researchers	Years	Topics	Opportunities
Everett and Farghal	1994	activity time and cost by using cubic learning curve on repetitive activities	Methodologies for Non-repetitive activities with forecasting capabilities
Anderson and Parker Wu and Sun	2002, 2006	Engineering based model for outsourcing objectives; Staff allocation to minimize outsourcing costs	Multiple learning curves in the system simultaneously
Vits, Gelders, and Pintelon; Jaber and Guiffrida	2006, 2004	Process change effectiveness when unit costs are subject to learning curves; Rework time per unit from learning curves	Different learning curves for different processes in the system
Leong and Cho; Mosheiov and Sidney	2006, 2003	Feature selections relevant to the product quality; Job dependent learning curve for a single machine flow time	Different learning curves for different processes in the system
Biskup and Simons	2004	Possible effects in the initial investment toward technological knowledge enhancement due to learning curves	Multiple processes exist in the same system
Argote and Epple	1990	Organization productivity increases based on employee learning rates	Multiple learning curves for different groups or individuals
Demeester and Qi	2005	Allocate learning resources for one product	Interactions among partners and design and engineering teams of different learning curves
Shafer, Nembhard, and Uzumeri; Teunter et al.	2001, 2004	Learning curve effects in a car radio assembly line; Inventory strategy for unequal lead-time items	The use of one learning curve versus many individual learning curves
Everett and Farghal	1997	Unit data, cumulative-average, moving average, and exponentially weighted averaging data techniques	Multiple learning curves for multiple processes in the same system

2.3 Previous Research on Large Scale Production System

Scales of production systems vary from automobiles to aircraft. Gilmore and Pine (1997) defined four distinct production system approaches: collaborative, adaptive, cosmetic, and transparent. Agrawal (2001) broadly concluded that build-to-order was already a reality: customers could place special orders for cars at the time of his study. His research showed that 17 percent of North American car buyers would purchase a build-to-order vehicle the next time they bought a car – if they did not have to pay more and got what they had ordered in eight weeks or less.

Wang and Jiao (2003) presented an optimal product family configuration model which was constructed via the application of a genetic algorithm. Their case study focused on the electronic motors of hand phones on six customizable modules. Khouja (2003) considered a simple serial supply chain case in which each supplier had a single source for each component. Khouja then optimized the synchronization level of the just-in-time schedule. His scheme is very useful in a static system when process times are static.

An architecture by Karcher and Glander (2003) defined four layers for the flexible customized integration of different systems: engineering process, basic service, process-oriented services, and existing system in a Product Data Management environment (PDM). Gao et al. (2003) illustrated a different PDM by implementing the international standard for exchange of product data model STEP in the design environment with manufacturing and enterprise resource management systems. A different approach was proposed by Fogliatto et al. (2003), who designed a customization index to estimate the viability of implementing a production system of customized products. This system is based on three variables: customer requirements, supplier delivery flexibility and production flexibility for different characteristics of a product. A Quality Function Deployment matrix was applied for the index plan. Their case study examined the manufacture of window type air-conditioning equipment.

Xu and Yan (2006) studied an injection mold design, employing the Quality Function Deployment (QFD) in a 'house of quality' environment. They developed a time estimation method for the product remodeling design, based on a fuzzy neural network model.

Similarly, they successfully presented a linear process in a linear system of a single line event.

Herroelen and Leus (2004) said that the execution of projects may be the subject of considerable uncertainty, which could lead to numerous schedule disruptions. They offered a framework that allowed project management to identify the proper scheduling methodology for different project scheduling environments with a survey of the basics of critical chain scheduling. They also indicated in which environments their framework would be more useful. Their approach depends on levels of dependencies and variability. Faaland and Schmitt (2004) addressed economic lot scheduling (ELS) with lost sales and setup times. They addressed the ELS problem of manufacturers that make many product types on a single facility or assembly line. Their study is useful in addressing potential facility idle times with regarding to the production rates and setup times of a component manufacturer in an integrated production system.

Sugimura et al. (2001/2003) addressed flexible production control in holonic manufacturing systems. The integration of the process planning and the scheduling systems was done by using combined methods of genetic algorithms and dynamic programming. Kotak et al. (2003) described a practical system framework for holonic design and operations in a distributed manufacturing environment using multi-agent systems and simulation techniques. Three components were proposed and developed in their system: holonic control, virtual simulation, and human/system integration. Every entity in their system is a holon. Their simulation model has one source, 3 sinks, 2 decision points, and 4 conveyors in a continuous flow production. Cheng et al. (2004) used UML case, use, and activity diagrams to address a holonic information coordination system (HICS) for manufacturers. Their architecture was designed to realize agile manufacturing. A communication holon (CH) was applied to establish a HICS. Web technology was used to exchange data/information among manufacturing entities in the semiconductor industry.

Supply chains consist of interlinked networks of suppliers, manufacturers, distributors, and customers that provide a product or service to customers, according to Blackhurst et al. (2004). Stochastic operations of typical supply chains can be equipped with Internet

technologies on a network-based methodology to model, react, and change supply chain systems. Lou et al. (2004) published an architecture employing case-based reasoning to analyze the management of agile, multi-agent-based supply chains.

Villa (2002) mentioned that the actual goal of an effective production system is to obtain a proper integration of all intelligent agents, in order to make each local strategy as cooperative as possible. Their paper presented a set of design criteria that could be used by a designer to organize more efficient management systems. Thus, every manufacturing and distribution step, from raw material acquisition to final product delivery, should in principle be incorporated into a supply chain, which should connect material suppliers, producers, distributors, and customers. Shah et al. (2002) depicted a framework of supplier-to-partner and supplier-to-supplier relationships to ensure traceable products in a production system with inter-organizational information system matrix.

The methodology of integrated information systems design presented by Garcia and Dominguez (2004) was for an integrated information system design, to identify standardized information, and to develop partial standard models. They touched many systems in an overview fashion (systems such as: CAD, CAE, CAM, CRM – customer relation management, DRP – distribution requirement planning, ERP – enterprise resource planning, MES – manufacturing execution system, MRP – manufacturing requirement planning, MRP II – Manufacturing Resource Planning, SFC – Shop Floor Control, PDM – Product Data Management, SCM – Supply Chain Management). They presented much information to consider in an integrated system. Bigand et al. (2004) created an information system to integrate different viewpoints on the design and the control of a flexible manufacturing system in an automated production systems environment. Their work can be partially applied to a larger scale production system. Shunk et al. (2003) presented the function, information, dynamics, and organization (FIDO) multi-view enterprise-modeling methodology developed to support the understanding, analysis, and design of business processes aligned with information systems for inter-enterprise integration between companies as well as for a single company. They presented the application of the FIDO methodology to analyze the interactions and to establish the supply-chain process integration between primes and subs in the defense electronics

industry. The presented FIDO integrated modeling framework consisted of model integration, paradigm integration, and systems development life cycle integration.

Table 3 Previous Research on Large Scale Production Systems

Researchers	Years	Topics	Opportunities
Xu and Yan Mikkola and Gassmann; Agrawal Hartley et al.	2006, 2003, 2001, 2004	Time estimation method of product design; Desired level of lead time reductions	More than one event in the system simultaneously; Synchronization of lead time reductions
Gilmore and Pine	1997	Types of customized production systems	Asynchronous production processes
Wang and Jiao	2003	Genetic algorithm in electronic motor modules	Multi-tier of suppliers in the same production system
Khouja Villa Shah et al.	2003, 2002, 2002	Serial supply chain production system	Non-static processes of different lead times
Karcher and Glander; Gao et al., Fogliatto et al.	2003, 2003, 2003	Flexible customized integration; Production data in enterprise resource management; Product index plan in an AC manufacturer	ASLC in the system
Sugimura et al. Kotak et al. Cheng et al.	2001, 2003, 2004	Holonic manufacturing systems	More complicated systems with ASLC
Blackhurst et al. Lou et al.	2004, 2004	Supply chain architecture and modeling considerations	More complicated systems with ASLC
Herroelen and Leus Faaland and Schmitt	2004, 2004	Critical chain scheduling, economic lot scheduling	Unequal lead times of processes
Garcia and Dominguez, Bigand et al., Shunk et al., Linton, Choi et al. Chen and Lin	2004, 2004, 2003, 2003, 2002, 2004	Integrated information system design in a production system	More complicated systems with ASLC

Modular products are gaining popularity in integrated manufacturing systems. A mathematical model introducing a modularization function for analyzing the degree of modularity in a given product architecture was introduced by Mikkola and Gassmann (2003). Their paper indicates that modular products may shorten new product development time and help to introduce multiple product models quickly with new product variants at reduced costs. In addition, modular production can also facilitate the rapid introduction of many successive versions of the same product line with increased performance levels. Their concept of levels of modularity can be similarly extended to reflect levels of customizations in both product design and supply chain structures. In another study by Hartley et al. (2004), one of his findings suggests that, contrary to expectations, e-auction use and supplier collaboration are not mutually exclusive.

Objects of large scale complex products in an integrated production system consist of multiple tiers of integrators, suppliers, and designers. These integrators, suppliers, and designers do not necessarily maintain one-to-one relationships. Some of the major component integrators may be suppliers to other integrators. Design activities also may take place at all levels in such integrated systems. Producers of some of the non-customized common products may opt to design their own products. These producers often supply many major suppliers, who very possibly could also be partners and/or competitors simultaneously. Producers of customized products may design those products jointly with the final integrator, so relationships associated with participation and decision levels hardly will be a straight vertical integration or one-to-one, as seen in many traditional mass-production systems. Considering that business plan, psychological factors, and operational issues are the three major reasons that Internet-based businesses fail, partner relations can be crucial to whether such enteprises succeed or not. (Linton 2003). Three supplier-to-supplier relationship archetypes were discussed by Choi et al. (2002), (1) the competitive supplier-supplier relationship, (2) the cooperative supplier-supplier relationship, and (3) the "coopetitive" [concomitantly competing and cooperating] supplier-supplier relationship. Product development and manufacturing may involve all three of these relationships during different stages of the process. Team relationships can also be considered from the personality point of view in concurrent engineering, where

project tasks generally involve the establishment of multifunctional design teams in order to simultaneously consider various activities throughout the entire product life cycle (Chen and Lin 2004).

2.4 Previous Research on ASLC Related Statistical, Simulation, and Verification Methods

Alfieri and Brandimarte (1997) applied a simulation language: MODSIM via a modular approach. Their decision procedures included: structural network design, inventory management, and transportation management. Addressing factories, stock-points and demand points, their object oriented modeling was based on the integration of the static point of view, dynamic point of view, and functional point of view. Hardgrave and Johnson (2003) did an information systems development acceptance model via a similar object oriented approach.

A prototype intelligent concurrent design task planner for shortening the time-to-market of a product was developed by Jiao et al. (2004). They applied a genetic algorithm and claimed that eighty percent of production decisions are made during the design phase. At the end of the iteration stage, the amount of work remaining to be done for each activity was given by the sum of the remaining work, which was the difference between the amount of remaining work at the previous stage and that completed at the current stage, and the amount of rework produced at the current stage. Their prototype planner comprised three modules: design process modeling, configuration, and scheduling. Crossover operations take place within the section selected according to the weights assigned to each section.

A case study in a pharmaceutical company was performed by Hung et al. (2004). They did a realistic supply chain simulation with informed comparison between different supply chain policies. Object-oriented dynamic modeling was used to enable dynamic stochastic simulations (discrete event simulations) on supply chains. Beamon (1998, 1999, and 2001) provided detailed framework and a process quality model for single and conjoined supply chain performance measures that considered resources, output, and flexibility in convergent, divergent conjoined and general classifications. Beamon's research results

provided part of the foundation in the area of the simulation modeling of supply chain management events.

Multiple runs of Monte Carlo simulation were applied to produce cost and duration distributions in the development process for a military product by Browning and Eppinger (2002). Iteration was addressed as a fundamental feature of the product development process. Their model yielded and reinforced several managerial insights, including: how rework cascades through a product development process, trading off cost and schedule risk, interface criticality, and occasions for iterative overlapping. Verbraeck and Versteegt (2001) conducted discrete event simulation in different systems where logistics coordination is a specification of a dependency at a certain moment in time. Communication in their model was event-driven and it only occurred when the control system or the equipment had reached a certain stage in the execution of activities.

Roy and Arunachalam (2004) presented parallel discrete event simulation (PDES) with the distributed execution of large-scale system models on multiple processors. The key issue in PDES was to maintain causality relationships between system events, while maximizing parallelism in their execution. Their paper proposed a modified optimistic scheme for the distributed simulation of constituent models of a supply chain in manufacturing, which exploits the inherent operating characteristics of its domain. Thier algorithm reads input queue per its simulation time as compared with the last entry. Their method may not work in some DES models using only one CPU, since the CPU is constantly checking that entity status and cannot do anything else. They provided a useful theoretical framework, although it lacked real applications or software demonstrations. With the aim of minimizing design time, Yoshimura et al. (2003) illustrated two main techniques based on criteria interrelationships: the selection of criteria blocks and the sequencing of both the criteria blocks and the criteria within them. When dealing with design projects as large-scale optimization problems, the need to shorten the development cycle to a certain period becomes the objective. Their method was to list all criteria and design variables, express interrelationships among criteria using the functional dependence table, block criteria according to their relationships, and optimize the order of criteria for maximum efficiency. Their example was designing a bicycle.

In order to make the scheduling model more capable and practical for industrial use, Lin et al. (2004) presented a new scheduling model based on the workflow management technique (SMWMT). SMWMT was a process-oriented compound model. Their structure contained the process view, the resource view, and the job view. Their dispatching rules were earliest due date, shortest process time, smallest slack per remaining operation, and smallest slack.

In modeling a large system, (Brailsford et al. 2004) describes how system dynamics was used as a central part of a whole-system review of emergency and on-demand health care. Bandivadekar et al. (2004) presented a model to examine the effect of future changes in vehicle material composition on the automotive recycling infrastructure. A material flows and economic exchanges model was developed using VENSIM by Ventana Systems to describe the material flows in the automotive life cycle industry chain in terms of new cars and cars already present on the road. They modeled the aging chain and the recycling chain with respect to the types of materials and costs. Dynamic modeling of a production-inventory system usually involves lead-time models (Wikner 2003). Three different approaches to continuous time dynamic modeling of variable lead times based on control theory were discussed in the Wikner's paper. These three approaches to the lead-time modeling were: (1) first-order delay, (2) third-order delay, and (3) pure delay.

Considering much of the research mentioned above while facing challenges in simulation modeling, Fowler and Rose (2004) indicated the first challenge is an order of magnitude reduction in problem-solving cycles. The second challenge is the development of real-time, simulation-based problem-solving capability. The third challenge is the need for true plug-and-play interoperability of simulations and supporting software. Finally, there is the biggest challenge facing modeling and simulation analysts today, that of convincing management to sponsor modeling and simulation projects.

Levels of production system complication depend on product volume and design configurations. The more mass-produced a product is, the less the likelihood of customization. It is important for design teams to explore the customer's perception of the appearance of a target product. Tseng et al. (1998) combined virtual prototyping (VP) with

design by manufacturing simulation techniques, which allowed for a company's individual customization requirements and process capabilities to be balanced in the design stage. Types of VP presented in that paper include immersive virtual reality and analytical virtual reality. Key techniques using VP-aided design of customized products consist of product representation and model generation, human-computer interaction, manufacturing simulation, and product library.

More research literature on the use of discrete event simulation for manufacturing system design and operation problems was reviewed and classified by Smith. (Smith 2003). Three primary classes of research were considered in Smith's paper: (1) manufacturing system design, (2) manufacturing system operation, and (3) simulation language/package development for manufacturing systems applications. In simulation research conducted on a semiconductor wafer factory, Schruben and Roeder (2003) said the execution speed of a resource-driven model was found to be insensitive to system congestion, whereas a job-driven model was found to slow dramatically as the system became heavily loaded. They concluded that a resource-driven approach using event scheduling logic offers the best approach to modeling very large-scale, highly congested systems. Their finding was important in modeling low-volume, high-mix manufacturing systems.

Literature research performed in this work explored articles in the areas of learning curve theories, learning curve applications, large scale production system, and ASLC Related statistical, simulation, and verification methods. Research opportunities will be discussed in the next section.

Table 4 Previous Research on ASLC Related Statistical, Simulation, and Verification Methods

Researchers	Years	Topics	Opportunities
Alfieri and Brandimarte; Hardgrave and Johnson	1997, 2003	Object oriented modular simulation modeling	System view of process lead time changes
Jiao	2004	Weighted operations using a genetic algorithm	A statistical verification method of the model
1. Hung et al., 2. Roy and Arunachalam, 3. Yoshimura et al. 4. Beamon	1. 2004, 2. 2004, 3. 2003, 4. 1998, 1999, 2001	Supply chain simulation	Verification of the simulation method in LSPS and scales of their systems
Lin et al. Verbraeck and Versteegt	2004, 2001	Process-oriented compound model; Event decision points were pre-determined	ASLC in the system and statistical methods
Fowler and Rose	2004	Simulation modeling challenges	Verification of the simulation method
Bandivadekar et al. Brailsford et al. Wikner	2004, 2004, 2003	Material flow, large system, and lead time modeling using dynamic modeling	Discrete event simulation with statistical methods
Tseng et al. Browning and Eppinger	1998, 2002	Virtual Prototyping product features; Product development processes	Expend to model the production system
Smith Schruben and Roeder	2003, 2003	Manufacturing system design; low-volume high-mix system	ASLC in the system and statistical methods

2.5 Opportunities

Partners, suppliers, system integrators, and most other participants have different priorities and individual interests in a large scale production system. Thus there exist opportunities to research relationships among different priorities in a system in which ASLC takes effect among all parties. The comprehensive literature survey presented in the previous sections addresses available opportunities for further meaningful research. This section depicts research opportunities that tailor to the targeted contributions to literature outlined in the Chapter 1.

One common feature of existing studies was that their processes were independent of dynamic interactions with other processes characterized by different learning curves. Hence, their production systems were somewhat different from the production system in this study. There were cases in which some dynamic correlations among workers and/or processes were not as independent as had been assumed. This work uses individual learning curves for each process that are asynchronous to other processes, among which dynamic interactions are occurring.

The log2-based learning curve is believed to have the best forecasting capability (Everett and Farghal 1994), which is one of the key topics of this research. Thus, this research applies log2-based learning curves. The default 100% learning progress rate (b=1, in equation 7, on P. 49) means that for every doubling of the repeated numbers of tasks, the process time it takes to perform the task will be reduced by half.

Different customized features require different design and engineering efforts. Different design and engineering efforts have different levels of learning which possess diverse learning curve rates. Asynchronous stochastic learning curve is a commonly seen system behavior among partners in large scale production systems. The nature of ASLC with dynamic, unequally-distributed lead times and improvement rates has been very problematic for partners striving to meet deadlines simultaneously.

Table 5 Research Opportunities

Prior Research Areas	**Opportunities**	**Addressed in Dissertation?**
Learning Curve Models	Asynchronous stochastic events in learning curves	Yes
	Ability to forecast system performance when ASLCs are in the system	In case studies
	Apply statistical and / or discrete event simulation methods to learning curve related modeling.	Yes
	Using log 2 based learning curves to forecast system performance	Yes
	Distribution within each learning curve, and ASLC among partners	Can be applied in case studies
	Multiple machines / processes in a system and log scale learning curves	Yes
Learning Curve Applications	Methodologies for Non-repetitive activities with forecasting capabilities	By using log 2 based learning curve methods
	Multiple learning curves for different groups or individuals in the same system	Yes
	Interactions among partners and design and engineering teams of different learning curves	Via case studies and by ANOVA method
	The use of a single learning curve versus individual learning curves	Yes
Large Scale Production System	More than one event in the system simultaneously; More complicated systems with ASLC	Yes
	Unequal lead times of processes More complicated systems with ASLC; Non-static processes of different lead times	Yes
	Asynchronous production processes; Synchronization of lead time reductions	Yes
	Multi-tier of suppliers in the same production system	Yes, via case studies

Table 5: continued

Prior Research Areas	**Opportunities**	**Addressed in Dissertation?**
ASLC Related Statistical, Simulation, and Verification Methods	System view of process lead time changes	Yes
	A statistical verification method of the model	Yes
	Verification of the simulation method in LSPS and scales of their systems	Yes
	ASLC in the system and statistical methods	Yes
	Verification of the simulation method	Yes
	Discrete event simulation with statistical methods	Yes
	Expand to model the production system	Yes

Chapter 3: A Large Scale Production System

3.1 System Overview

A large scale production system often consists of a group of structured organizations located in different geographic regions, working cohesively to produce a family of complex and customized products. The nature of a large scale production system is likely to have evolved from and is probably still driven by product characteristics. Products such as high-rise apartment buildings, ocean vessels, and commercial airplanes have a common need for large pre-fabricated components. Their common characteristics necessitate that their major partner organizations that manufacture these components cannot feasibly be located in the same geographical area. Thus, the infrastructure spread of participating organizations combined with the physical size of final products determine the nature of Large Scale Production Systems.

The framework in a large scale production system consists of a manufacturing partnership structure, a supply chain, logistics, product design planning, and product launch scheduling. It also requires the ability to adapt to market trends, the statistically stable and predictable measurable performance throughout the system, the pull-and-push of mixed logistics in the whole system, and control of the time factor. All of these framework considerations are shown in figure 1.

Product design planning is often market driven and normally scheduled backwards. Demand for most large scale integrated products, such as airplanes and ocean vessels, rely heavily on regional and global economic stability and growth possibilities. Demand is also often cyclical in the range of decades, not just years. Thus, the planning and scheduling part of the framework goes backwards from the forecasted final product delivery to the end-item logistics.

The traditional manufacturing system ordinarily progresses according to established planning and scheduling. In a system of partners and suppliers, both pull and push (Lu 2006) will take place alternately and simultaneously as part of the production and supply chain logistics. End items gather together where large component assembly takes place.

The final product integration is performed by transporting matching large components of the same product (identified by a production sequence number) in a timely manner. Most large-scale system integrators of customized assemblies strive to minimize the "makespan" during the final product integration stage, while the time between final product deliveries and the final product integration can be much longer and less predictable than the production time resides within the final product integration. There are common potential delays in apartment building, ship building, and airplane industries. These common delays between their final product integration and final product deliveries can be caused by cosmetic finishes, regulatory inspections, functional testing, permit processes, and / or the buyers' financial readiness. It may not be feasible to apply further product configuration related processes to fully assembled products, especially after their final integration.

The schedule from end item to the final product integration, as shown in figure 1, is said to be developed backward as part of the planning activity. The End Item Schedule is led by the Component Schedule, which is driven by the Master Schedule. The ability to change levels of product configuration and production according to the schedule status along the product integration time scale varies because of constraints imposed by statistically stable and predictable end item/component/final product feature production distributions.

If this framework is applied to new product launches, timing associated with customized product design by the final integrator and all tiers of suppliers must be statistically dependable. The associated parties receive the time scale via a predefined information flow chain that shares common and secure communication protocols. Such a design information chain shall be regarded as the necessary condition for a new product launch system model framework. The same product sections can be designed by multiple parties and custom manufactured by either the same or by different parties in the system.

This framework can assist in understanding a large scale production system. Employing both statistical and simulation methods in it not only allow one to observe production activities in the system, but also enable one to make more statistically reliable forecasts of

effects caused by impromptu or customized changes made throughout the product integration cycle.

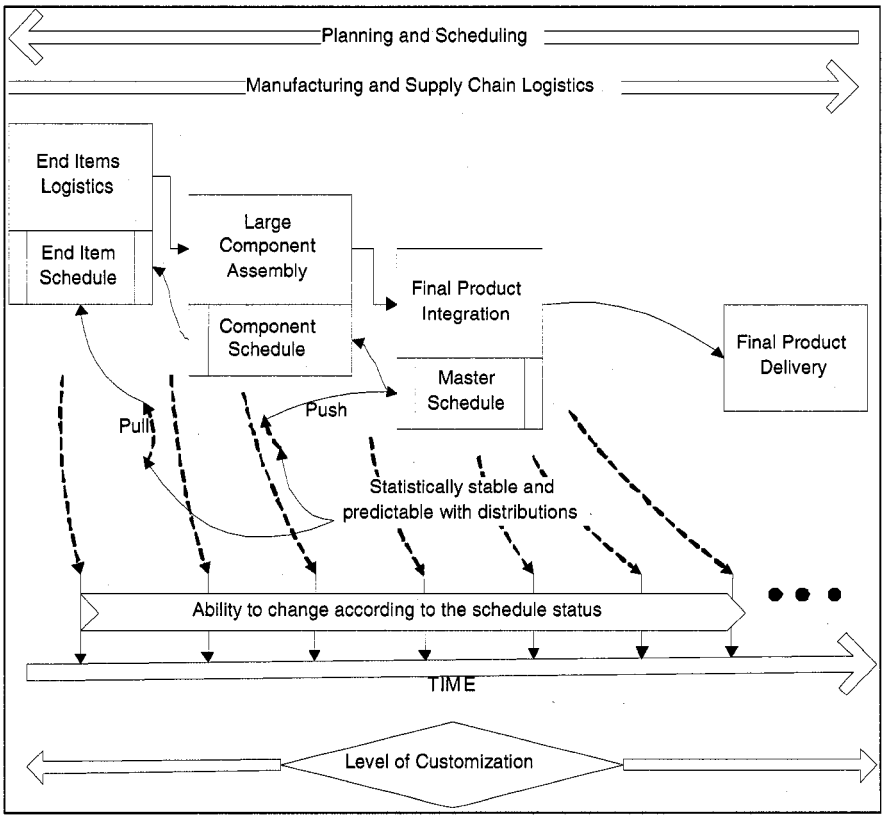


Figure 1 System Modeling of a Customized Product Supply Chain in a Large, Integrated System

3.2 System Boundaries

The effectiveness of the system does not solely rely on any single group of activities, not to mention on the performance of a single entity in the system. It is necessary for all entities to perform to gain overall system effectiveness. Appealing product design normally would be more promising in increasing market share, but only if the rest of the system can be managed and planned cohesively in an ever changing dynamic and balanced manner. The ability to allow production changes throughout the product logistics time-span must take all individual entities and the overall cohesiveness of the system into account. The consequence of issuing changes at various critical points of the system

varies drastically depending on the status of the product integration and on initial design configurations. The timing of change in a system can be a function of cost and schedule. If products requiring large-scale assemblies and integration were designed in configurations that made future dynamic customized changes difficult, no matter how individual entities performed or how efficient the supplier logistics may functioned, the whole system would not be able to act and react fast enough to positive market opportunities.

Large production system integrators face many challenges and risks during the lengthy product development phases until the production system becomes stabilized. Most companies would never be able to recover from a failure due to unexpectedly large product development costs or cost overruns upon the introduction of a brand new mass customized product such as a commercial airplane or an ocean vessel.

A large scale production system can be categorized by the timing of key groups of system events in four main stages. These stages apply mostly to the airplane industry; however, the underlying principle in asynchronous stochastic learning curve related effects applies to other industries as well.

1. Conceptual product design stage

In this stage, many different product concepts are studied. Products such as airplanes will have their key features scoped and potential partners will be considered. Scopes of key product features may include their general weight, size, operating ranges, market segments. Partner consideration may include political complications among nations and continents in addition to their respective design and manufacturing capabilities.

2. Product launch stage

In this stage, key product design and configurations will be determined, finalized, and deployed. A multi-billion dollar program with a thirty-year or longer lifespan is authorized to go forward. Detailed design and production responsibilities of tier one major component suppliers and partners are loosely defined.

3. Detailed product design and initial production stage

Major product development, design and production are taking place simultaneously throughout the global production system. Design and production responsibilities of tier 1 partners are defined. Thus, partners are involved in designing and producing their respective first articles. Since each partner has different design and process progress rate, curve rates of their learning curves are not necessarily the same. However, products of all tier 1 partners are to be completed and delivered to their respective destinations synchronized per produce serial numbers. During this stage, ASLC are affecting the system.

4. Stable product production stage

In this stage, all production activities have been stabilized. Partners of all tiers are in repetitive production processes per given rates.

The scope of this research is within the third stage: Detailed product design and initial production stage, where ASLCs are present throughout global partners.

3.3 System Components

The system in this research produces products only if there are customers who have already committed to the purchase. All major components of each product have to be traceable to their source of origin. This type of system is required for products such as commercial aircraft that involve many suppliers of different components. There are two types of suppliers in this large scale production system: mass-production producer and customized producer. The mass production suppliers repeatedly and continually produce the same items, such as common structural parts, brackets, nuts and bolts, etc. The customized product suppliers fabricate products with a set of given customization attributes per selections chosen by the final customer before and/or during different production stages.

Wang and Jiao (2003) described a product family configuration design for an assemble-to-order production strategy, in which a configuration space is first created based on the

generic variety representation of product family architectures. Customer-perceived quality and total customization cost are also identified as design objectives. Garcia and Dominguez (2004) had a methodology of integrated information systems design. They identified information as standardized to develop partial standard models which may be useful in categorizing information in mass customization. Kurniawan et al. (2003) stated that product configuration has been recognized as a key enabler for Mass Customization; allowing customers to choose their products based on choice of product attributes, which is different from the product selection process. In the decision-making process in mass customized product attribute selection, manufacturers determine which attributes consumers can customize, and let consumers make the selection of an attribute value in each attribute. These attributes are then combined to enable consumers to perform the configuration process. Their case study involved a t-shirt designing and purchasing process.

The nature of components in this study, however, can be categorized into three different types:

Category 1. The fixed optional component.

This type of component has one attribute: to be installed or not. If yes, it will be a straight plug-in. An example is whether to have a forward cargo air conditioning system or not in a commercial airplane.

Category 2. A configurable optional component.

This type of component has multiple attributes. Further configuration has to be performed before the production and installation of such components can take place. A common example is the in-flight entertainment (IFE) system in a commercial airplane. Most airlines have different configurations of their IFE for different routes.

Category 3. A retro-fit optional and/or customized component.

This type of component has almost unlimited attributes. Most of these types of components are totally customized and installed after the main product has been finished. An example of such components is the interior of a private business jet.

In the literature review by Da Silveira et al. (2001), they stated that future research could focus on the formulation of methodologies that enable the rapid reconfiguration of existing organizational structures and processes into an integrated production system.

The system in this research is framed around the first two categories mentioned above. From the final customer's point of view, it is desirable to delay decisions for as many customized product attributes as possible, which normally is the opposite from the system integrator's point of view.

Components in a large scale production system are addressed from the product, process, and system entities categories. Product and process component categories are listed in table 6 below.

Table 6 Product and Process Components in a LSPS

	Component Categories		
Product	The Final Product	Integrated	Major Structure
	Customer Furnished	Assembled	Detailed
	System	Plug-and-play	Custom made
Process	Major component design	Transportation logistics	Customized configuration
	Minor component design	Assembly sequences	Work package definitions
	Due-Date criteria	Customization criteria	Component ordering

These system entities are companies categorized by their functional responsibilities. All of them operate according to the general direction set by the final system integrator.

However, they are not likely to have the same performance priority individually. System entities regarded in this research are:

- Detail part manufacturers without customization. They produce the same part routinely until there is a minor model introduction and/or product revision or change.

- Component producers without customization. They produce sub-assemblies by using non-customized detail parts.

- Component integrators with customization. They integrate components together with customization. They are also involved in the design processes.

- Major work package integrators with customization. Their role is similar to the above except they handle more complicated work packages, which are composed of integrated components.

- The overall final integrator with customization. They are the ultimate responsible entity who integrate all work packages together with customized features.

Given the described product configuration-management-centered system, the product structure is illustrated in figure 2 below. The final product integration contains a group of large component assemblies, whose end items are supported by a given number and type of suppliers. All of the small common parts, such as commonly used rivets, can be mass-produced by sub-suppliers. There may be multiple tiers of end item suppliers and sub-suppliers. Large component assembly is likely to take place simultaneously at multiple geographical locations.

End-item (Ei) suppliers (Si), designers (Di), and the component itself (Ci) contribute to the level of interactions of each large component, LiXi. Multiple levels of interaction complexities (X1, X2, etc.) of the same large component assembly, say L1, may have multiple suppliers (S1, S2, etc.), designers (D1, D2, etc.), and components (C1, C2, etc.).

Based on such understanding, one can then derive a hierarchical structure of all customized large-component assemblies, defined as follows.

$$\sum_i \sum_j L_i X_j = \{E_i, S_i, D_i, C_i\} \tag{4}$$

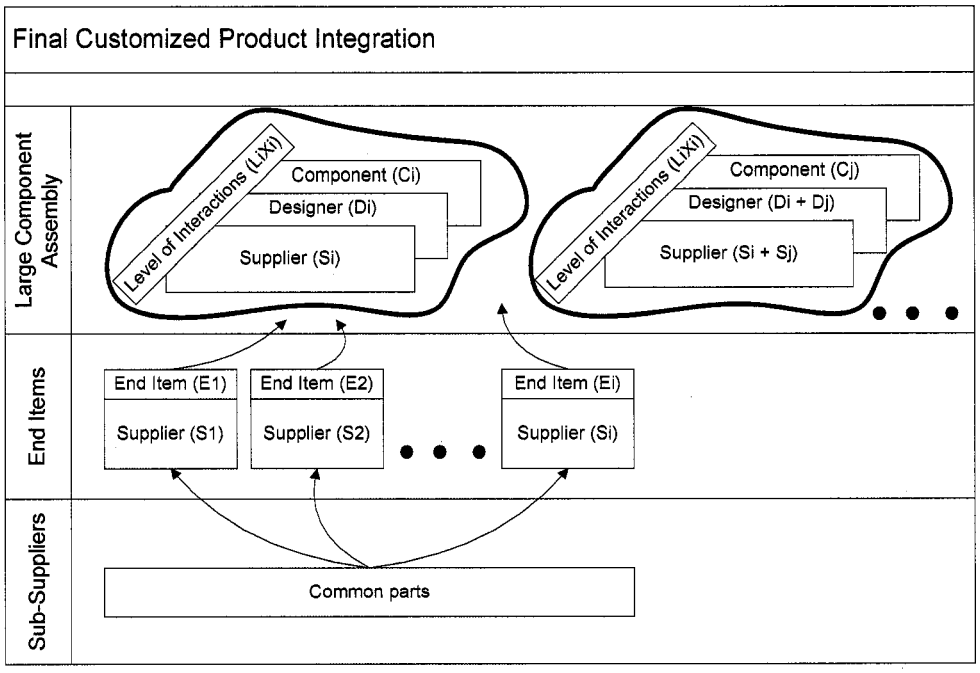


Figure 2 Product Structure

3.4 System Characteristics

Consumer commodity type products that can be acquired from multiple sources normally do not require customized design or a special production system infrastructure. A company that produces all the components for one type of product most likely will not need to be concerned about system integration or partner collaborations. Interfaces for most of the commodity consumer products have already been configured and commonly accepted. Customized products that require electrical and mechanical continuities between sections that come from different suppliers will need a different level of integration throughout the product development and launch phases. There are benefits when suppliers and partners participate in the product design and launch from the early

phases of new product development. Thus, system characteristics of complex and customized products greatly differ from consumer commodity type products.

For a customized product that takes a few years to design and for which it also takes more than one year to produce and integrate the first production unit, it is important for suppliers to be involved in the early stages of customized product design and testing cohesively with the pre-integration and the final integration parties. All partners concurrently progress through the phases of the product launch process. The benefits of orchestrating such a horizontally and vertically integrated manufacturing system with major suppliers and partners engaged from the very early stage include risk sharing and a faster product launch for the next generation of the same product family. Suppliers mostly likely will have guaranteed future business for the given unique component for the life of the final product, which can last more than thirty years. However, the final system integrator is supposedly capable of producing most of the components if it chooses to do so. Because components do not have the same level of complexity, not all suppliers / partners improve through the component manufacturing phases at the same rate. Tremendous amounts of learning takes place when the final system integrator is designing, testing, and inventing product and process characteristics. In this system when suppliers and partners share the risks and benefits of the whole system, all suppliers / partners throughout the supply chain will need to learn together and improve processes.

Configuration decisions of customized product design for customer needs involve interactions beyond just those between the design and the marketing departments. For customized products that require large-scale production in an integrated system, each entity in the system has its own desired configuration decisions. Entities of such a system may consist of the final product integrator, designers, suppliers, and customers. Individual customer-preferred configuration considerations in a large scale production system are illustrated in figure 3 below.

Stummer and Heidenberger (2003) took into account the theme profiles of the objectives, various project interdependencies, logical and strategic requirements, as well as resource and benefit constraints in describing a three-phase approach to assist research and

development managers in obtaining the most attractive project portfolio. Comparable analogies can also be applied in product design and configuration processes.

As indicated in the figure 3, the design activity may have its own preference in order to optimize certain product features in the design process. When multiple exclusive suppliers are involved, say supplier type I and type II, with different configuration priorities, design activities will then have to accommodate per suppliers' specialties. Some of the components can be mass-produced and some will need to be custom manufactured for supplier type I and II, respectively. The configuration management of customer furnished components, purchased and often partially specified by customers directly from supplier type I and/or II, and then directly delivered to the final product integration location for assembly involves customers, product designers and suppliers. Assembly capability is one of the key considerations that cannot be ignored by the product configuration management and designers. The customer's opinions can be communicated to the designers in the form of quality function deployment (QFD).

Most major components in a large scale production system require unusual logistics in the supply chain infrastructure because of their large physical sizes. There are limited number of capable suppliers globally that can produce certain large size components. Furthermore, the locations of these suppliers and their available logistics resources vary across the whole system. Most major suppliers in this type of system collaborate in relationships that resemble partnerships. It has been very common to have more than sixty percent of the final integrated product value the responsibilities of the suppliers. The integration of the manufacturing processes and supplier event management with respect to their different learning curve rates are the keys to the success of such system integration.

There are many different types of suppliers/partners, some of which produce the same product over and over, while others only make specialized products for certain configuration requirements, and most others are somewhere in between. There are three types of suppliers considered in the system: the type 1 suppliers (S1) provide components of same configurations to the pre-integration location, the type 2 suppliers (S2) provide

minor variations of the same component to the pre-integration location, and the type 3 suppliers (S3) provide components of more variations to the final integration site.

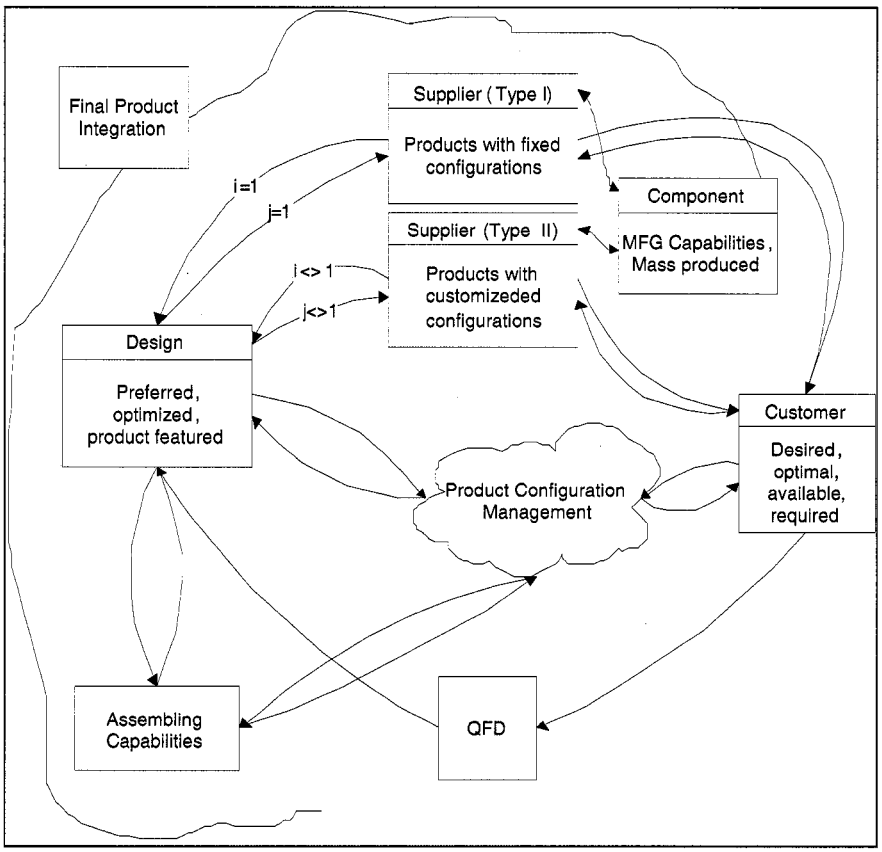


Figure 3 Individual Customer-Preferred Configuration Considerations in an LSPS

Most products produced from large production system have individually specified customers predetermined many years and/or months prior to product final integration and delivery. Demand forecasts throughout all configurable levels in this system are not easily nor accurately developed. However, such forecasts are necessary to plan supplier related events. Customers prefer that product configurations be finalized as late as possible. On the other hand, the system integrator and suppliers prefer finalized configurations as early as possible. In an actual system, some configurations are common throughout all product

lines and some customized configurations are to be determined at a shorter lead-time prior to the final product delivery. S1 type suppliers do not handle variable configurations, hence, S1 type suppliers can stock up at a level deemed practical for business execution. S2 type suppliers provide mostly fixed variations for certain customers that do not vary every time. S3 type suppliers only provide customer and product specific configurations directly to the final integration site. Lead times of S3 type suppliers are critical to customers who wish to make final product configurations at the latest possible stage of the product integration stream. Figure 4 illustrates a brief overview of this example system.

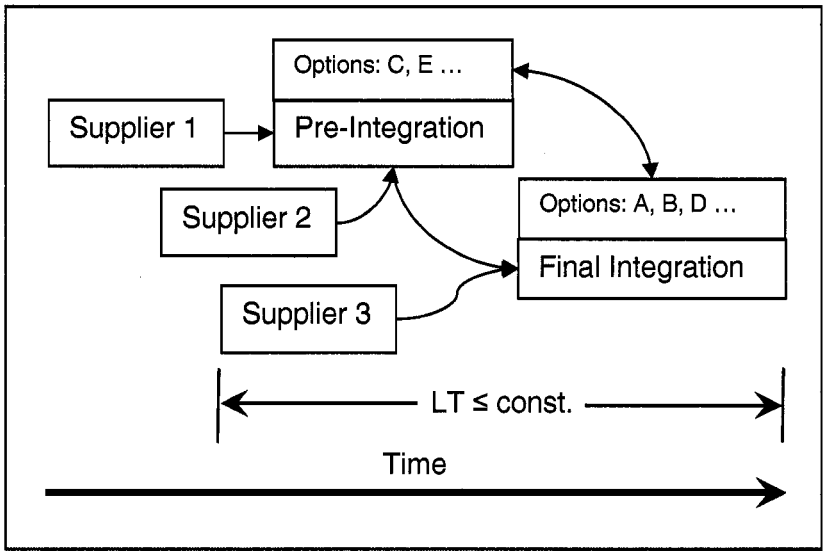


Figure 4 Overview of The Sample System

There are many unique characteristics in this system such as all partners/suppliers are solo suppliers without any backup sources, almost none of the two suppliers have the same lead-time or learning curve, and almost all produced components are to be integrated to the final product. Each entity in the system possesses the learning / improvement curve characteristics.

All suppliers in this system are handling the more complicated tasks that require more product precision, while those suppliers which produce the most ordinary mass-produced

consumer type products handle less complex tasks. The later would have neither the technology nor the resources to manage operations within the large manufacturing integration arena. However, even the most sophisticated suppliers must exert considerable effort in order to perform complicated large-scale manufacturing tasks especially during the early stages of a new product launch. Consequently, the rate of improvement for each supplier is a critical system characteristic that can deeply influence overall system efficiencies.

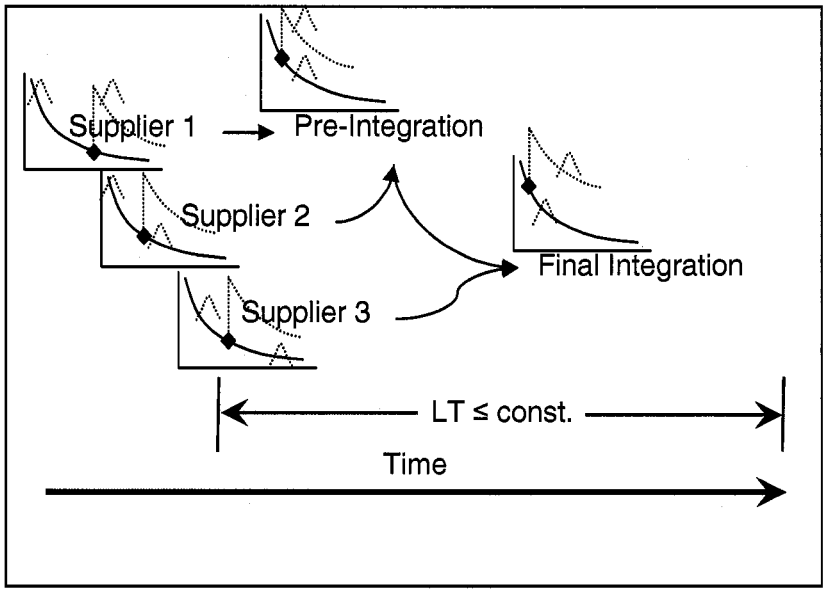


Figure 5 Improvement Curves in the System with Lead-Time Constraints

Not all suppliers have the same process time for their products, and the starting time of suppliers on the same product sequence number component will not be the same. In this system, all components of the same sequence number will need to be gathered together to go on the same sequence number product. Therefore, the supplier's improvement curve influences the overall system performance. For example, one supplier is working on the sequence number 25 and another supplier is on the sequence number 56. Both of them can be at different points on their improvement curve at the same time point. Total system inventory and logistics can be significantly influenced by their unequal rate of

improvement. Figure 5 shows improvement curves in the system with a lead time constraint. Black diamonds on the curve indicate that if a change is made in the system, each supplier will not be at the same location on their respective improvement curves.

Planned and unexpected are basically two types of changes in the system before all suppliers can improve their processes to a steady state. Planned changes can be a catalogue refresh and unexpected changes can be product recalls, safety bulletins, natural disasters, etc. Since each supplier has to learn how to make a new product and then improve the process performance per their curve rate, if the product requirement changes then the process rate very likely will be riding on a new curve as seen in the figure 5. It is preferable that the changes not happen too often, since this offsets the curve before the process can improve along its original track.

Logistics impacts are another important issue in this type of system. The transportation time of large components of the same serial number across several continents varies between large air cargo freighters and ocean vessels, especially when there are changes in the system.

The next chapter describes asynchronous stochastic learning curves in a large scale production system.

Chapter 4: Asynchronous Stochastic Learning Curves

4.1 ASLC Framework

The ASLC effect is at its fullest strength during the detailed product design and initial production stage, which were both outlined in the previous chapter. In the industries where ASLC matters, every major component produced is integrated into the final product. Figure 6 outlines the ASLC component Gantt chart of an example system with three major components of three serial units. The horizontal axis presents time that goes from left to right. The left and right edges of each bar present their process starting and ending times, respectively. Thus, the width of each bar presents the process duration.

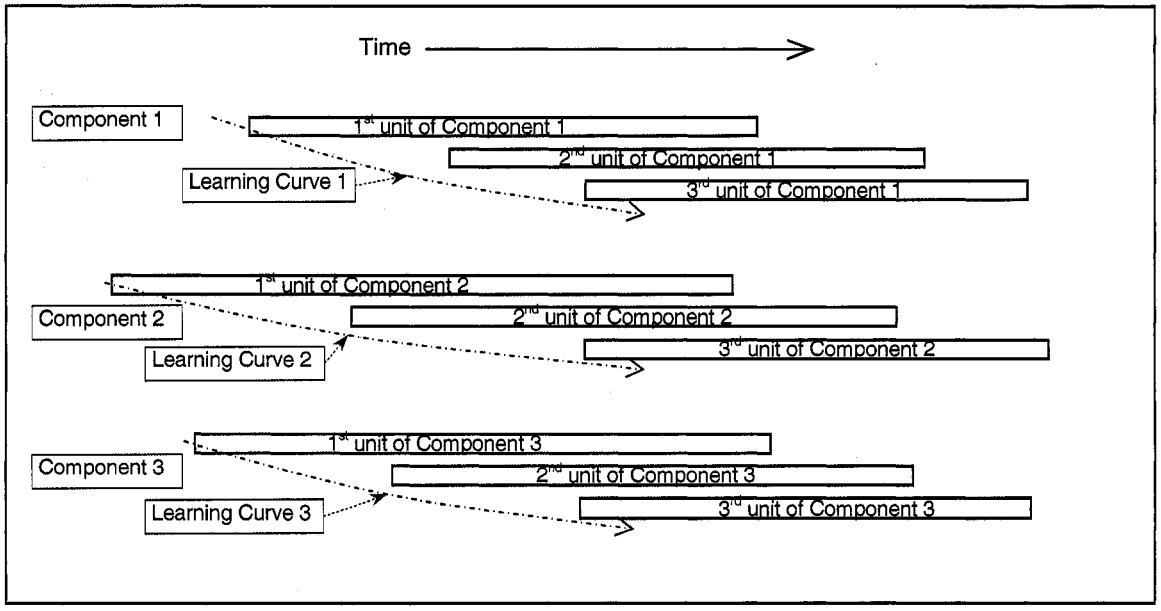


Figure 6 The ASLC Component Gantt Chart

The 2nd unit of component 1 takes less time to produce than the 1st unit of the component 1, and so on. Thus, the process times of three consecutively produced units is gradually reduced in each component according its learning curve characteristics. There are three different learning curves for three components presented in the figure 6. It is highly desirable that the ending times of all three components per serial unit be as close as possible, while the starting times of each unit sequence number among different

components vary per their different lead times, which are distinguished by their learning curves. This is due to the asynchronous stochastic nature of the production system.

The rate of process time reductions among these three components can be different and also stochastic in nature, since it is desirable to have same serial number units among all components finish as close in time as possible. From the right-to-left scheduling concept, the same serial number units of different components do not have their starting times progress through the system in the same rates; i.e. the starting time of the 1st unit of component one is after the 1st unit of component two; but the starting time of the 3rd unit of component one is about the same as the 3rd unit of component two. In this example, the component two process times have a faster improvement rate than the component one.

The Gantt chart of three components of three units each in Figure 6 illustrates this ASLC framework by showing the asynchronous lead times per process in a stochastic system of different and often random process finish times. The next section addresses additional elements in a large scale production system with ASLC effects.

4.2 ASLC Elements

There are many elements to consider in framing a production system with ASLC effects. Elements considered in this research mainly reside in the detailed product design and initial production stage. Table 7 outlines ASLC elements and their relevance to the production unit, component and the final product due dates.

ASLC Element relevance is based on whether the element has direct influences on the system with respect to Production Units, Components, and / or Final Product Due Dates from the learning curve effect point of view. Most of the ASLC elements are relevant to production units, components, and final product due dates; except these elements: unit serial number, process time to produce the 1st unit, learning progress rate, and learning curve rate. The unit serial number is an attribute of the production unit itself. The process time to produce the 1st unit does not relate to the final product due dates because the learning curve effect starts after the 1st unit has been produced. Both learning progress rates and learning curve rates are attributes of a given learning curve per component, thus

there is no indicator relevance to the production unit within the component family. The rest of the ASLC elements have indicators that relate to both production units and components. Mathematical formulas of these ASLC elements will be illustrated in the next chapter.

Table 7 ASLC Elements

ASLC Elements	Element Relevance		
	Production Units	Components	Final Product Due Dates
Unit serial number	Yes	No	No
Process time to product the 1st unit	Yes	Yes	No
Process time to product the xth unit	Yes	Yes	Yes
Learning Progress Rate	No	Yes	Yes
Learning Curve Rate	No	Yes	Yes
Targeted unit starting times	Yes	Yes	Yes
Actual unit starting times	Yes	Yes	Yes
Targeted unit finish times	Yes	Yes	Yes
Actual unit finish times	Yes	Yes	Yes
Targeted unit total flow times	Yes	Yes	Yes
Actual unit total flow times	Yes	Yes	Yes
Forecasted serial number unit that flatten the learning curve	Yes	Yes	Yes
The on-time performance	Yes	Yes	Yes

4.3 ASLC Stages and Performances

Processes that follow the commonly known log2 based learning curve improve at a pace according to their learning curve rate. The log2 based learning curve reduces process times by multiplying their learning curve rate by the unit process time whenever the process units are doubled. The unit of the learning curve rate is in percentage. Therefore, process time reductions of the initial few production units can be at a drastic rate.

The nature of the initial steep process improvement directly influences the lead times of component units. As shown in the figure 6, the 1st unit of component two needs the longest lead time than 1st units of components one and three. They have different learning curve characteristics, at the 3rd unit, the lead times of all three components are about the

same. Therefore, the first stage in an ASLC influenced large scale production system is the fast process time improvement stage, illustrated in Figure 7. The first ASLC stage is very noticeable when the performance of any process deviates from the desired path, either for better or for worse.

The dotted line in the Figure 7 presents the theoretical learning curve of process flow times against the product serial numbers; the solid line represents the planned process takt time changes. It is desirable to have both lines overlap as much as possible. If the dotted line (process times) lies above the solid line (takt times), then the process times will not likely to meet planned takt times. If the dotted line lies below the solid line, extra capacity exists from the process time point of view.

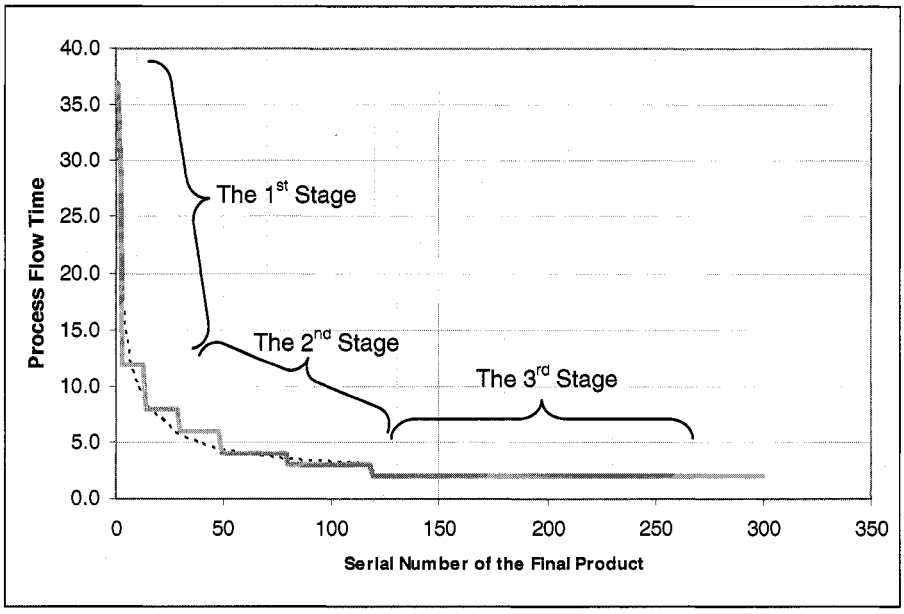


Figure 7 ASLC Stages

The noticeable transition of processes progress from the first ASLC stage to the steady learning improvement, the second stage, in an ASLC production system can be a subjective judgment. Since the process learning curve (the dotted line in Figure 7), flows continuously, the takt time (the solid line in Figure 7) reductions can be an indicator when the process has reached the second stage. The production rate changes, as seen as takt

times, more then 400% (from about 37 to 8) in just a few production units as the process progresses through the first ASLC stage. During the second ASLC stage, the production rate may change another 400% (from about 8 to 2), but it will take more than 100 production units to go through the learning curve. At the end of the second ASLC stage, the production takt time can be a flattened line.

The third ASLC stage is the steady process time stage with a flattened learning curve. This is the stage in which processes are operating just as in a mass production environment. The starting point of the third ASLC stage may be predetermined and / or observed from experiences. It is important to the production system that the dotted line and the solid line overlap throughout the third ASLC stage completely. Figure 8 illustrates the same information as seen in the Figure 7, except axes in the Figure 8 are in log10-based scales.

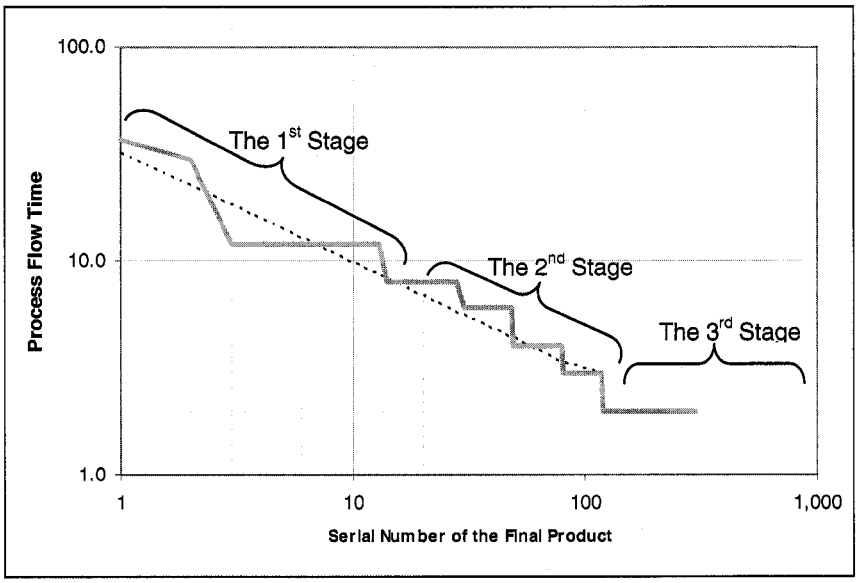


Figure 8 ASLC Stages in Log Scale

ASLC effects in a large scale production system are more important in the first two stages than in the third stage. As mentioned earlier, all components produced during these first two stages are going to be integrated into the final integrated product in this system. Most

large scale production systems in the real world strive to gain the best possible operating practices during the first two ASLC stages. Hence, a more detailed description and modeling of the ASLC could contribute significantly to production efficiency. The next few chapters will address the mathematical ASLC model and its methodologies in both statistical and discrete event simulation methods.

Chapter 5: ASLC Mathematical Presentation and Hypotheses

5.1 ASLC Assumptions and Variables

The ASLC model described in this research contains unique features based on the following assumptions in the system.

Assumptions:

1. The time needed to process each unit of each component can be different from unit to unit and from component to component
2. Each production unit has its own unique sequence number, or commonly known as serial number
3. There is one learning curve progress rate per component in the system
4. There is one learning curve rate per component in the system
5. The time needed to produce a unit of a component of a given sequence number has a target value per component
6. The actual time it takes to produce a unit of a component of a given sequence number may have a different value than its target value
7. Each unit of a component of a given sequence number has a targeted production starting time
8. The actual starting time of each unit of a component of a given sequence number may have a different value than its target value
9. Each unit of a component of a given sequence number has a targeted total flow time
10. The actual total flow time of each unit of a component of a given sequence number may have a different value than its target value
11. Component process time improvements per its given learning curve characteristics will eventually come to a flattened point at which learning curve effects can no longer contribute process time reductions. The flattened point is presented by a given component sequence number which is the same serial number unit across

all components. All components may not reach the same serial number unit, that starts the flattening of the learning curve effect, at the same time

12. All units of their respective components of the same sequence number are intended to arrive at the final integration point simultaneously
13. All units of their respective components of the same sequence number may not arrive at the final integration point simultaneously

The attributes of a learning curve, the asynchronous nature of process start and end times and the stochastic nature of the process durations in a large scale production system all have their respective assumptions listed above. Variables in the system per the assumption are listed below.

Variables:

$y_{i,x}$: Time needed to process the x_i^{th} unit of the i^{th} component in the system

x_i : The produced unit sequence number of the i^{th} component in the system

a_i : Time needed to produce the 1^{st} unit of the i^{th} component in the system

b_i : Learning progress rate of the i^{th} component in the system

c_i : Learning curve rate of the i^{th} component in the system

$F_{i,x}$: The targeted x^{th} unit of the i^{th} component arrival time at the final assembly

$f_{i,x}$: The actual x^{th} unit of the i^{th} component arrival time at the final assembly

$S_{i,x}$: The targeted x^{th} unit of the i^{th} component production starting time

$s_{i,x}$: The actual x^{th} unit of the i^{th} component production starting time

$T_{i,x}$: The targeted total flow time of the x^{th} unit of the i^{th} component

$t_{i,x}$: The actual total flow time of the x^{th} unit of the i^{th} component

m_i : The sequence number, which the learning curve flattens, of the i^{th} component

$N_{i,x}$: The on-time performance of the x^{th} unit of the i^{th} component

5.2 ASLC Mathematical Model

Wright first introduced the learning curve in 1936 and his log2 based model is still the best model in predicting future process performance in a system in which learning curve has an effect. Their linear log x, log y learning curve model is:

$$y_x = ax^{-b} \tag{5}$$

$$\log y_x = \log a - b \log x \tag{6}$$

Where:

y = time needed to produce the x^{th} unit

x = cumulative number of units produced

a = time needed to produce the 1st unit

b = progress rate

The commonly known method of determining the progress rate is to take the logarithm of an improvement percentage, or curve rate, divided by log 2, as shown in the equation (7) below. Basically, if a supplier has a 50 percent CurveRate, then this supplier is capable of reducing its process time by half when the quantity of product produced has doubled. In most cases, the CurveRate will not be at 100 percent, which means all component units have the same process time without the learning curve effect.

$$b = -\frac{\log(CurveRate)}{\log 2} \quad \text{where } 0 < \text{CurveRate} \leq 1 \tag{7}$$

Since it is assumed that all processes achieve no further improvement after a certain pre-determined quantity of units has produced, the process time of any given component is the maximum value between y_n in the equation (6) and the forecasted target process time of the 100th unit.

The process time of the n^{th} unit thus can be derived as:

$$y_n = 10^{[\log(a) - b\log(n)]} \tag{8}$$

The process improvement normally goes only so far before it is flattened. Thus, if the objective is to achieve a stable process time at the 100th unit, and a desired Curve Rate has been established. The process time of the 1st unit derived, from equation (6), is:

$$\log(a) = \log(y_{100}) + b\log(100) \tag{9}$$

$$a = 10^{[\log(y_{100}) + b\log(100)]} \tag{10}$$

The production performance over time of all component suppliers may not have the same set of learning curve characteristics in a large scale production system with the ASLC effects. The ASLC in this work is based on Wright's model with additional indices to present individual learning curve effects per a given component supplier. Therefore, the time needed to process the x_i^{th} unit of the i^{th} component in the system is:

$$y_{i,x} = a_i x_i^{-b_i} \tag{11}$$

$$b_i = -\frac{\log(c_i)}{\log 2} \tag{12}$$

The c_i is the "CurveRate" mentioned in the equation (7). Equation (11) can also be presented by applying logarithm on both sides:

$$\log(a_i) = \log(y_{i,m_i}) + b_i \log(m_i) \tag{13}$$

Time needed to produce the 1^{st} unit of the i^{th} component in the system:

$$a_i = 10^{\left[\log(y_{i,m_i}) + b_i \log(m_i)\right]} \tag{14}$$

The time needed to process the x_i^{th} unit of the i^{th} component in the system without considering the flattening of the learn curve:

$$y_{i,x_i} = 10^{\lceil \log(a_i) - b_i \log(x_i) \rceil} \quad (15)$$

In case where the flattening of the learning curve takes effect after the m^{th} unit. The time needed to process the x_i^{th} unit of the i^{th} component in the system considering the flattening of the learn curve:

$$y_{i,x_i} = Maximum\left[10^{\lceil \log(a_i) - b_i \log(x_i) \rceil}, 10^{\lceil \log(a_i) - b_i \log(m_i) \rceil}\right] \quad (16)$$

The targeted x^{th} unit of the i^{th} component arrival time at the final assembly subtracts the time needed to process the x_i^{th} unit of the i^{th} component in the system yields the targeted x^{th} unit of the i^{th} component production starting time, which is shown in the equation (17) below.

$$S_{i,x_i} = F_{i,x_i} - y_{i,x_i} \quad (17)$$

The targeted x^{th} unit of the i^{th} component arrival time at the final assembly subtracts the targeted x^{th} unit of the i^{th} component production starting time yields the targeted total flow time of the x^{th} unit of the i^{th} component, which is shown in the equation (18) below.

$$T_{i,x} = F_{i,x} - S_{i,x} \quad (18)$$

The actual x^{th} unit of the i^{th} component arrival time at the final assembly subtracts the actual x^{th} unit of the i^{th} component production starting time yields the actual total flow time of the x^{th} unit of the i^{th} component, which is shown in the equation (19) below.

$$t_{i,x} = f_{i,x} - s_{i,x} \quad (19)$$

It is highly desirable to have all the targeted x^{th} unit of the i^{th} component arrival times at the final assembly ($F_{i,x}$) and the actual x^{th} unit of the i^{th} component arrival times at the final assembly ($f_{i,x}$) as close to the same as possible. Due to the nature of the ASLC, it may

be very difficult to have all targeted ($S_{i,x}$) and actual ($s_{i,x}$) x^{th} unit of the i^{th} component production starting times synchronized. The motivation for having process starting times as less asynchronous as possible is to have the on-time performance of the x^{th} unit of the i^{th} component as less stochastic as possible.

The on-time arrival performance ($N_{i,x}$) of the x^{th} unit of the i^{th} component can be obtained by subtracting the actual and the targeted x^{th} unit of the i^{th} component arrival times at the final assembly. This is shown in the equation (20) below.

$$N_{i,x} = f_{i,x} - F_x \tag{20}$$

The mathematical model of the ASLC presented from equation (5) to (20) is depicted graphically in the Figure 9. Three groups of component processes are demonstrated in the system with the ASLC effects in the form of three sets of three horizontal bars. Each component shows three consecutive production units and each component has a set of unique learning curve characteristics. Figure 9 presents a portion of the same production system as the Figure 6. Figure 9 illustrates the full nature of the ASLC in an example production system. The asynchronous process starting times are shown in three groups with shaded circular, hexagon, and diamond shapes. These different shapes are to illustrate different process start times among the same sequence number units of different components. All circular shaped processes (the 1^{st} unit of a component) should have finish times that are as close as possible. The finish times among all hexagon shaped (the 2^{nd} unit of a component) processes should also be close in time. However, their starting times and process durations are not necessarily the same. In fact, they can be very asynchronous.

Three different learning curves for three respective components are shown as equations y_{1x}, y_{2x}, and y_{3x}. Each of the learning curve equation represent all units of a component. As mentioned in the assumptions, there is one and only one learning curve per component in the system. Learning curves affect the process time durations of individual units of their respective components. The process durations and ending times of each unit of the same sequence number across all components may vary from the designed value due to many

factors. Hence, the process end times of each unit of the same sequence number can be stochastic. The targeted and actual component arrival times ($F_{i,x}$, $f_{i,x}$) of each sequence unit number may also vary from one sequence number to the next. The on-time arrival performance then varies from one sequence number to the next. The stochastic nature of the system performance can be seen as the distribution of N_1, N_2, and N_3, and are stochastically different.

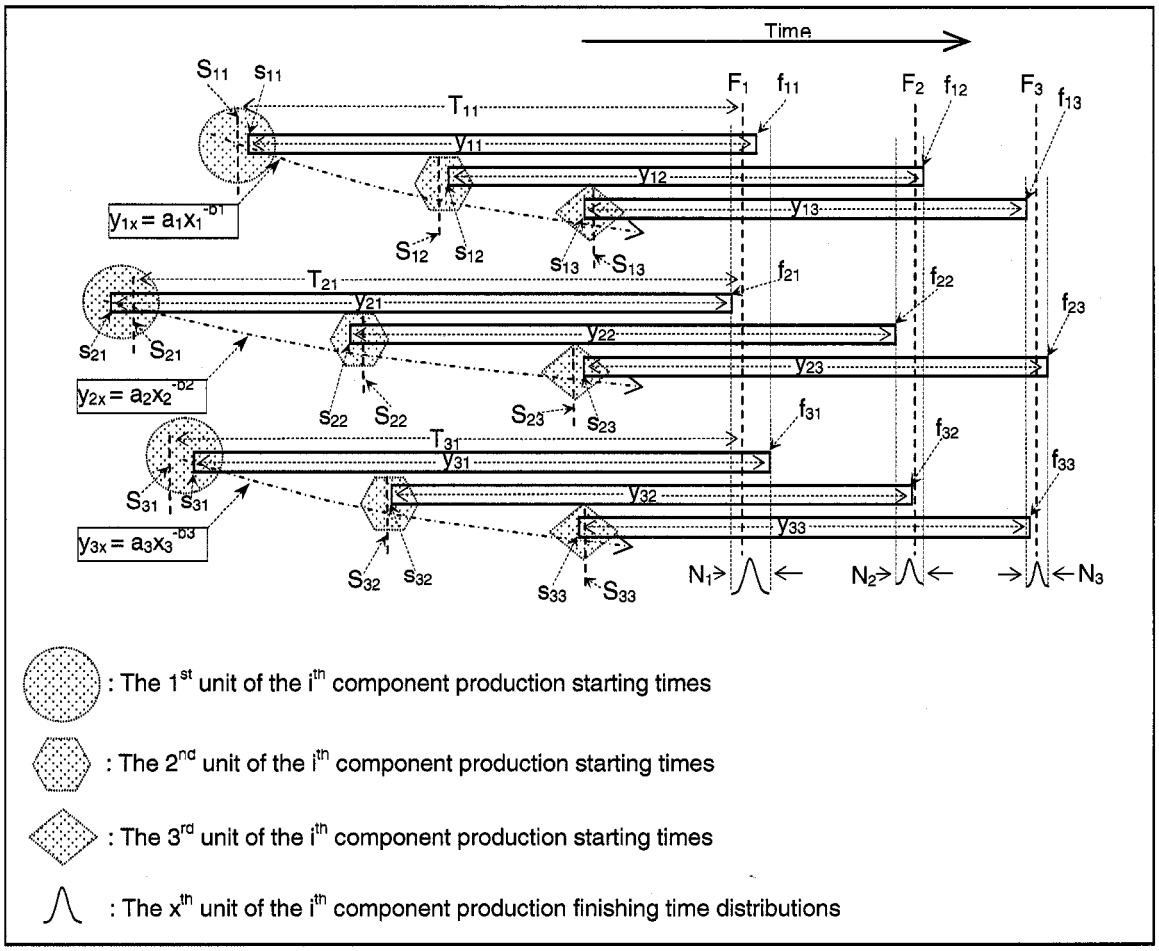


Figure 9 ASLC Effects in a Large Scale Production System

It is important to capture the system performance by using the on-time arrival performance ($N_{i,x}$) of all units of all components as the response. It is a system performance indicator, which presents effects, variables, and attributes across the whole production system

collectively. Asynchronous learning curves results, stochastic process durations, final integration cycle rates, and the rate that the cycle rate changes are all possible contributing factors that influence the on-time arrival performance ($N_{i,x}$)

The hypotheses of the on-time arrival performance response in a large scale production system with ASLC effects will be addressed in the next section.

5.3 ASLC Hypotheses

The objective in the described system of ASLCs is to have all components arrive at their destinations simultaneously and on time. The null hypothesis states that there is no fluctuation of component arrival times and the response of the null hypothesis can be stated as:

Null Hypothesis:

Ho: Learning curves from all partners are stable, repeatable, and with the same flattened pattern so that their process lead times are predictable. Thus, the on-time arrival performance of final product finishing dates is deterministically predictable.

Ho:

$$Y = N_{i,x_i} \approx 0 \quad \forall i \quad \forall x_i \tag{21}$$

Alternative Hypotheses:

Ha_1: Learning curves of all processes from all partners are stochastically asynchronous with different statistical distributions and potentially various flattened patterns. The process lead times of all partners are dynamically different due to the nature of ASLCs. Thus, the on-time arrival performance of final product finishing dates fluctuates.

Ha_1:

$$Y = N_{i,x_i} \neq 0 \quad \forall i \quad \forall x_i \tag{22}$$

Ha_2: The finish rate decreases as the learning curve improvement rates in different component production systems approach one another.

$$Y = N_{i,x_i} \approx 0 \; when \sum_{i=1}^{n-1} | c_{i+1} - c_i | \approx 0 \quad \forall i \quad \forall x_i \tag{23}$$

where n = total number of components

Ha_3: Under deterministically fixed component process durations for each given unit, the final on-time arrival performance of final production finishing rate decreases as the production cycle time stays constant.

$$Y = N_{i,x_i} \approx 0 \text{ when } \left\{ \sum_{i=1}^{n-2} \left(| F_{i+2} - F_{i+1} | - | F_{i+1} - F_i | \right) \approx 0 \text{ and } \sum_{j=1}^{n-1} | y_{i,j+1} - y_{i,j} | \approx 0 \right\} \quad \forall i \quad \forall x_i \tag{24}$$

Ha_4: Under randomized distributions of component process duration for each given unit, the final on-time arrival performance of final production finishing rate decreases as the learning curve rate of each component is very close to or at 100%.

$$Y = N_{i,x_i} \approx 0 \text{ when } \left\{ \sum_{i=1}^{n} | c_i - 100 | \approx 0 \text{ and } \sum_{j=1}^{n-1} | y_{i,j+1} - y_{i,j} | \neq 0 \right\} \quad \forall i \quad \forall x_i \tag{25}$$

Four alternative hypotheses are discussed in this research. Methodologies to test all four of them will be discussed in the next chapter. Applications and case studies in using these hypotheses will be illustrated in following two chapters.

Chapter 6: ASLC Methodologies

6.1 ASLC Statistical Method

This section depicts a statistical methodology which addresses the ASLC model. All assumptions listed in the prior chapter are applicable to this statistical ASLC methodology. The objective of using a statistical methodology in a large scale production system with ASLC effects is to statistically evaluate the system response: the on-time performance of the x^{th} unit of the i^{th} component ($N_{i,x}$). The sequential steps in the ASLC statistical method are listed below:

1. Process times forecast of the first production unit of a component for all components.
2. Process times forecast of the m^{th} production unit of a component for all components.
3. Varying the Learning Curve Rate (c_i) in equations (11) and (12) to obtain a matching set of Learning Progress Rates (b_i) for each component in the system.
4. Using equations (15) and (16) to obtain process times of each unit of a component for all components in the system.
5. Targeted x^{th} unit of the i^{th} component production starting times can then be obtained by using equation (17) together with the targeted x^{th} unit of the i^{th} component arrival time at the final assembly.
6. Apply randomization schemes to process actual starting times and durations for each production unit of a component. The following equations illustrate this step:

$$y_{ix} = y_{ix} \times (1 + \text{random}(\#) \times v\% \text{ changes}) \quad (26)$$

$$s_{ix} = S_{ix} + (0.5 - \text{random}(\#)) \times (n\% \text{ of the Finish Rate} \times \text{random}(\#) \times \text{Finish Rate}) \quad (27)$$

Where:

v: process changes due to additional configuration and/or customization demands

n% of the Finish Rate: In equation (27), the product Finish Rate is the time duration between two consecutive F's. Practitioners may assume an n% (e.g.: 10%) value that indicates the sensitivity of the product Finish Rate. The higher the percentage value in n, the more fluctuations will occur in process starting times. Equation (27) uses the targeted x^{th} unit of the i^{th} component production starting time plus or minus certain random percentages of the Finish Rate at the x^{th} unit. Then the statistically actual x^{th} unit of the i^{th} component production starting time can be estimated. Examples of using this method along with detailed data structures will be further discussed and illustrated in the next two chapters.

7. The targeted ($T_{i,x}$) and actual ($t_{i,x}$) total flow times of the x^{th} unit of the i^{th} component are then derived by using equations (18) and (19).

8. The system performance response ($N_{i,x}$), on-time performances of the x^{th} unit of the i^{th} component, is obtained by using equation (20).

9. A table that consists of these following variables can be constructed as shown below.

   m_i, $N_{i,x}$, $f_{i,x}$, $F_{i,x}$, $t_{i,x}$, $T_{i,x}$, $S_{i,x}$, $s_{i,x}$, $y_{i,x}$, and Finish Rates

   where i = 1 to 100 and x = 3

10. Populate the table, with i from 1 to 100 and x = 3, by using a portion of a data set that has i value from 1 to 300 and x value greater than 30.

11. Apply methodologies in statistical linear regression by treating the N as the response (Y) and the rest of all variables as factors (X's).

$$N = Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + e_i \qquad (28)$$

Where

$$N_x = MAX(f_{1x}, f_{2x}, f_{3x}, F_x) - MIN(f_{1x}, f_{2x}, f_{3x}, F_x) \quad (29)$$

The response, N, in equation (28) presents the system performance time span from the earliest to the latest component arrival times of the same unit sequence number, as shown in the equation (29).

It has been assumed that the residuals, given sufficient linearity, will have a common variance, which is both independent as well as identically distributed within a normal distribution.

12. Identify factors that have singularities among them after exercising the equation (28) via a linear regression exercise. Linear regression fails when factors which possess 100% correlation exist.

13. Repeat steps 11 and 12 above until singularity has been eliminated from all factors.

14. Perform an Analysis of Variance (ANOVA). The following ANOVA table for the response, N, is to be filled.

Table 8 An ANOVA Table Example [Montgomery 2005]

Source	Degree of Freedom	Sum of Squares	Mean Square	F	P
Factor 1					
Factor 2					
...					
Factor i					
Residual					
R^2					

Contents in the Table 8 above consist of a set of typical ANOVA practice as depicted by Montgomery (2005)

15. For alternative hypotheses, the same statistical methodology can be applied to derive an ANOVA table of the same response, N_x.

The statistical ASLC methodology allows feasibility in using the same set of process data as a discrete event simulation methodology and derives a set of response, N_x, on-time performance of component units. The statistical methodology using linear regression and ANOVA discussed in this section does not contribute any new techniques to the fundamentals of statistics, however, it provides a means to illustrate ASLC statistically.

6.2 ASLC Discrete Event Simulation Method

This section depicts a discrete event simulation methodology to address the ASLC model. Simulation modeling can be applied as a tool to oversee, examine, and predict the possible effects of products that have ASLC featured processes in a large scale production system.

Table 9 Basic Simulation Modules *

Modules	**Functions**
Entity	Represent end item units and components
Schedule	Govern release times of entities or resource scheduled availabilities
Create	Generate entities per schedules or random schemes, starts simulation
Queue	Store inventory entities and release them per rules
Process	Specify methods (resources / times) needed to process an entity
Resource	Resource(s) / costs needed for processes
Variable	User defines global variables that can be used in the model
Batch	Group entities together by numbers or their attributes
Dispose	Terminate entities in the system, ends simulation
Assign	Assign / update variables and attributes to entities.
Record	Collect statistical information
Decide	Route process sequence per percentage or conditional decisions
Record	Keep track of statistical performances in the model
Separate	Duplicate entities and / or reverse the batching process

* These modules are mainly based on Arena of the Rockwell Automation Technologies

There are several fundamental simulation elements in a discrete simulation model. These simulation elements may be referred to as modules, such as create, process, batch, assign, dispose, decide, separate, record, entity, queue, resource, etc. To create a framework for a large, customized integrated system, these simulation modeling objects need to be constructed appropriately. Basic functions of simulation modules are listed in table 9.

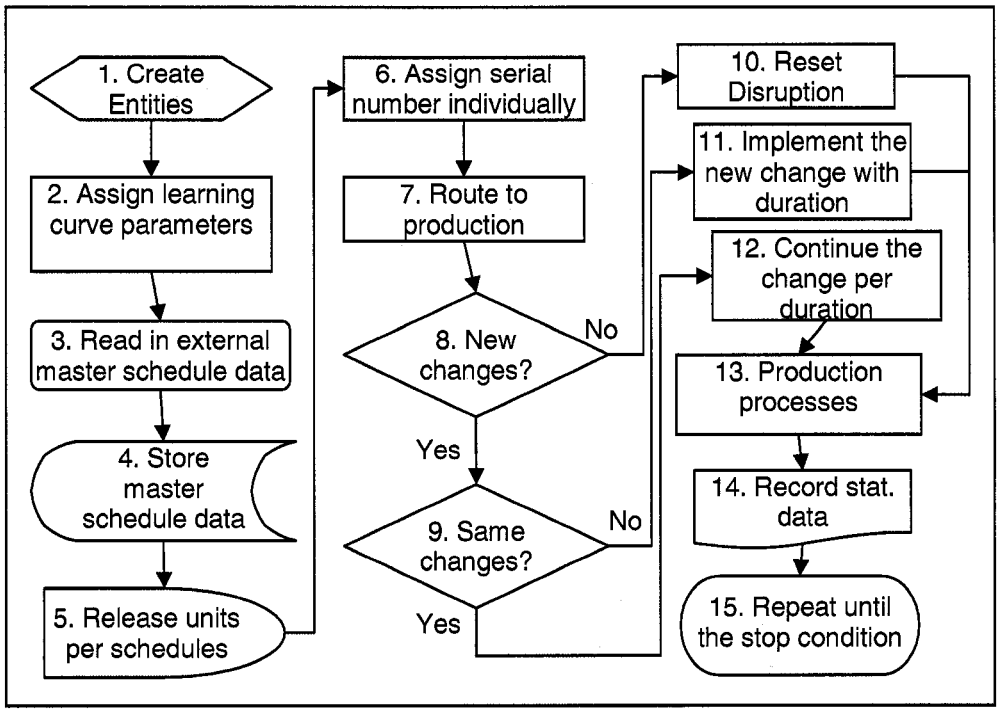


Figure 10 ASLC Simulation Method High Level Flow Diagram

Customized product handling in simulation models requires disciplined arrangements, because almost all end items need to be traceable in most large-scale customized assembly manufacturing processes. Traceable component batching from different suppliers with respect to their part transportation logistics has to be maintained. Therefore, a serial number for each entity in the system is needed among all modules for traceability and control. Additionally, the simulation model needs to properly address the

possibility that each component producing entity in the system performs on its own unique improvement curve rate with a statistical distribution.

Since every component has its own serial number attached as an attribute, each component is created individually in step 1 as seen in the Figure 10. The learning curve parameters are then assigned with the same component in step 2. External master schedule data can come from different formats; thousands of data points can be normalized and organized in a spreadsheet and then be read in during step 3. Step 4 then groups all of the component master schedule data individually for the whole simulation duration. Step 5 releases the production order according to the master schedule to each component process accordingly. Before the production order reaches any of the processes, serial numbers are assigned per component as in step 6. Step 7 receives the production order and then starts the process within their individual statistical process distribution.

Steps 8 and 9 check for any changes. If a change occurs in the system for the first time, then step 11 will take place. Step 12 manages the duration of the change. At the end of the current change, step 10 resets the change variable. Step 13 runs the process. Step 14 can record many different simulation results. Step 15 monitors whether the simulation stop condition has been met or not.

More detailed descriptions and heuristics of the discrete event simulation methods are discussed below.

1. Create Entities: A "Create" module creates one and only one entity per component type at the very beginning, time zero, of the simulation model. There are as many Create modules as component types.

2. Assign learning curve parameters: There are two attributes which need to be assigned early in the model: one is the learning curve rate (c_i) combined with the respective learning progress rate (b_i). This expression can be described as in the equation (12):

$$b_i = - \log(c_i/100) / \log(2) \quad (30)$$

where b_i is a variable and the whole expression is an attribute of an entity. 100 means that the learning curve of the process flattens at the 100^{th} serial number production unit.

The other is the processing time of the first production unit of a component. This can be done by utilizing equation (14).

$$a_i = 10 \wedge (\log(y_{i,mi}) + b_i * \log(m_i)) \quad (31)$$

3. Read in external master schedule data. External schedule data of all units of all components can be organized in spreadsheets, text files, and/or databases. The Simulation model can be connected to the data source by using a Read/Write module combined with a File module that has the data location specified. Schedule data in the data file presents actual process start times ($s_{i,x}$). Each one of the components shall have all of its unit process actual start times listed in a column that has a unique range name identification that enables the simulation model to distinguish these different start times. Table 10 illustrates an example of three components, each possessing ten production units. The process start times can be recorded in either a common calendar year/month/day/time format or in an Excel date format using numeric values as seen in the table 9. The values of the actual component unit starting times are derived via equation (28). The "Read" module reads in one value at one time for one component. In order to read-in actual starting times of all production units, a "Separate" module can be used right after the Read module. Every entity passing through the Separate module will be duplicated into two identical entities. One of the duplicated entities will go forward to the subsequent module, the other will be routed back to the Read module so that the next actual unit process starting time can be read-in. This combined use of both the Read and the Separate modules enables the Create module to create only one entity per component from the beginning of the simulation model.

Table 10 External data format example - process starting times

m_i	s1	s2	s3
1	13564.89	13842.80	13845.46
2	13703.42	13917.89	13911.40
3	13787.57	13969.30	13958.77
4	13825.57	13992.25	13977.42
5	13860.06	14013.93	13996.74
6	13890.83	14031.77	14014.93
7	13914.89	14049.91	14030.94
8	13939.10	14068.17	14046.87
9	13961.65	14086.14	14062.88
10	13982.24	14103.10	14078.14
...	...	...	...

4. Store master schedule data. Every entity passing through the Read module represents a production unit of a component. Every value that reads into the simulation model is stored as an attribute of its respective entity. An attribute of an entity stays with the entity throughout the simulation model. Every entity can have many attributes. Therefore all of the actual starting times of all units of all components are stored as attributes of their respective production units.

5. Release units per schedules. Since every production unit of a component has an attribute that represents the actual production starting time, a Delay module can be used to provide the needed time delay value the same as the schedule attribute value. Thus, each production unit of a component can be released to the simulation model according to its actual process start times ($s_{i,x}$). Figure 11 is an example of steps 1 through 5 in using a discrete event simulation method for an example component – Main Gear.

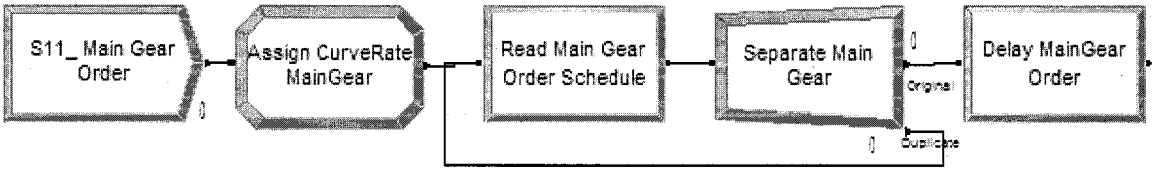


Figure 11 A simulation method to represent actual process starting times

6. Assign serial number individually. This step assigns a unique serial number to each unit of a component. Each production unit is like an entity and the serial number is as an attribute of the entity in the simulation model. In a large scale production system, units of many components are to be integrated into sub-assemblies, then to major assemblies, followed by the final integration. It is necessary to have all units with matching serial numbers assembled together throughout the production system. In order to model this system feature, each unit of a component has its own serial number. The serial number attribute is necessary to derive the process times of all units throughout the production system. These following equations illustrate a method for assigning serial number to a production unit:

$$varY1 = varY1 + 1 \tag{32}$$

$$attSN = varY1 \tag{33}$$

where:

varY1 is a variable that is not associated with any entity, varY1 has an initial value of zero and it increases its count by one whenever one entity has been processed by the Assign module.

attSN is an attribute that is attached to the entity that has just been processed by the Assign module. The value of the attSN is assigned in the same Assign module right after the variable value of varY1 had been updated.

The sequence of equation (32) and (33) shall not be reversed.

7. Route to production. A pair of modules: Route and Station modules are used to route entities to their respective production processes. The advantage of using this method instead of hard line connections among previous steps and all subsequent steps is to have flexibility in routing entities.

8. New changes. This step is an optional step that can be utilized to modify values of variables and/or attributes that will be part of process parameters of production units. Decisions with regard to disruptions of the system and/or customization features per particular condition can also be evaluated in this step. An example of this method is illustrated below.

   varTrigger1 == attSN || varTrigger2 ==attSN || varTrigger3 == attSN (34)

   Equation (34) above evaluates whether the serial number (attSN) of the current production unit satisfies any of the three possible triggering conditions (varTrigger1, varTrigger2, and varTrigger3), judged by serial numbers. The number of triggering conditions can be greater or less than three. Values of these triggering conditions can be generated by a random number generator and/or assigned with predetermined production serial numbers. Natural disruptions in the production system can constitute a case that randomly generates triggering conditions. The application of special processes and/or customization to certain serial number production units can constitute another case in which triggering conditions with predetermined values will be generated.

9. Same changes. This step works together with the previous step to examine whether the changes are the same as the last time, so that attributes and variables representing the change can be updated accordingly. More importantly, for changes of the same nature, the respective learning curves will continue; for a different new change, the respective learning curves will have a new set of parameters. The previous step uses a variable to decide whether there is a new

change. This step, however, uses an attribute to decide if the change is same as the previous change.

attNewTrigger == 0 ? (35)

10. Reset disruption. This step simply reset values of variables and attributes. The attribute that is used for disruption purposes will then be reset to zero as shown in the equation below.

attDisruption ==0 (36)

11. Implement the new change with duration. Process times of units of components will have their variables and attributes updated with the new change in this step. This step is a placeholder as the result of prior decisions.

These following attribute assignments take place for a new change:

attDisruption == 1 (37)

attDuration == varDuration1 (38)

attDelta1 == varDelta1 x (attDuration / varDuration1) (39)

attDuration == maximum ((attDuration – 1), 0) (40)

attNewTrigger == 1 (41)

where equation (37) assigns the binary disruption attribute, attDisruption, to 1; equation (38) assigns the duration of the disruption, stored as an integer variable: varDuration1, to the duration attribute, attDuration; equation (39) provides an attribute, attDelta1, that will be used later as part of process times of a unit of a component. The value of attDelta1 reduces gradually since the value of attribute: attDuration reduces the next time the same change takes place, as seen in the equation (40) while the variable varDuration remains the same as the

predetermined value. varDelta1 is a variable that represents the additional process time due to the new change. The change can affect the process time only as many as the varDuration allows. Equation (41) sets the binary attribute that will trigger process times with additional durations per prior equations.

12. Continue the change. When the new change is the same as the last one, this step processes the same change one more time to related process time of units in their component group.

    If it is the same change that has happened previously, then equations (37), (39), and (40) would be applicable as before. Equation (38) and (41) would not be applicable, since attribute values in attDuration and attNewTrigger had been assigned previously when it was a new change.

13. Production processes. This is the step where component units receive their respective process time and resource allocations. Learning curve effects, disruptions, and process changes are captured and statistically represented in this step. Process times at an integration process can be represented as:

    (Maximum [varCurrentProcessTime], [varFlattenProcessTime]) * (attDelta1 * attDisruption + 1) (42)

    Where,

    attDelta1 and attDisruption have been defined between equations (36) and (41). Since values of the attDisruption are either one or zero, and attDelta is in percentage, so (attDelta1 * attDisruption + 1) will provide a meaningful modification to the current process time of the unit of a component.

    Both the varCurrentProcessTime, the current production unit process time of a component, and varFlattenProcessTime, the unit process flatten time of the current production component, can be expressed in statistical distributions and equations (43) and (44) below show them in triangular distribution.

varCurrentProcessTime = Triangular ((varMinPercentage * $y_{i,attSN}$), ($y_{i,attSN}$), (varMaxPercentage * $y_{i,attSN}$)) (43)

varFlattenProcessTime = Triangular ((varMinPercentage * $y_{i,mi}$), ($y_{i,mi}$), (varMaxPercentage * $y_{i,mi}$)) (44)

The varMinPercentage has a value between 0 and 1. The varMaxPercentage has a value equal to or greater than 1. The unit process time of a component needs to have its serial number attribute (attSN) attached to it as in the equation (45) below.

$y_{i,attSN} = 10 \wedge (-b_i * \log(attSN) + \log(a_i))$ (45)

Since the process time of a unit of a component has to have its serial number as one of the entity attributes, it is necessary in a simulation model to express the process time of a component unit with all contents from equation (42) to (45) included.

Another type of process in the system is an optional process. An optional process handles customized and / or unexpected events that require additional process times. The process time in an optional process can be expressed as:

Triangular (varMinPercentage * varProcessVariation, varProcessVariation, varMaxPercentage * varProcessVariation) (46)

Where,

$$\text{var Pr}\,ocessVariation = \sum_{i=1}^{k} \text{var}\,Option[i] \times \left(\text{var}\,Option[i]Time + \sum_{j=1}^{n} attDelay[j] \right) \quad (47)$$

varOption[i] is a binary variable that represents types of possible options that may be represented in an optional process. varOption[i]Time is a real variable that contains the respective times needed for that option. attDelay[j] is an array of real attributes that are attached to each production component unit. Equation (47)

enables certain options to be added with consideration of the attributes of component units.

14. Record statistical data. The total process times are tracked by recording the differences between the stamped time before and after the processes. Therefore the impacts of all optional processes can be analyzed per component unit serial numbers in a tally data format. The recorded process time tally data is then outputted to another electronic file via a statistical output module. Thus, the response, N_x, the on time performance of all of the same serial numbered i^{th} units of their respective components, can be collected and evaluated.

The ASLC discrete event simulation method given in this section provides a usable methodology for executing the ASLC model. Possible applications in using such methodology are discussed in the next chapter.

6.3 ASLC Verification and Validation Method

If the ASLC model scheme represents a large scale production system correctly, as this research claims, and the discrete event simulation (DES) methodology applies the ASLC model correctly, then the statistical methodology result shall almost perfectly match the DES result. This section presents an ASLC verification and validation method that verifies results between the discrete event simulation and statistical methods in their depictions of the ASLC model.

Beamon (2004) mentioned that verification is the process of determining whether or not the simulation model is performing as intended, and that is an accurate representation of the conceptual model. One of the verification methods is to vary the input parameters and confirm that the output is reasonable.

Beamon (2004) also mentioned that validation is to confirm the simulation model is an approximate representation of the actual system of interest by using two possible levels. The two levels of validation are face validity and comparison with the real system. The face validity means that the model appears to be reasonable to an individual who is

familiar with the actual system of interest. Analysis of historical data may prove useful when comparison with the real system.

ASLC methods verification and validation processes in this research are outlined in figure 12, where steps 1 through 5 have been discussed in the previous two sections. Steps 6 through 10 are main steps used for result validation and verification. The verification of the DES method is performed in step 6 by using Beamon's method. The analytic statistical method calculates process durations (y_{ix}) for all units of all components in a spreadsheet or by using a calculator. The DES method calculates process durations (y_{ix}) dynamically as each entity passing through their process modules. Process duration is calculated for each component unit only when its representative entity is in the process module. The verification of the DES method is conducted by verifying component unit process times in the DES model against their equivalent component units in the analytic statistical method.

Once ASLC featured component unit process times are verified, the validation of the DES result can start, beginning at step 7. The ideal validation is to validate the DES modeled system with the real world system. Since, the real world system data has been non-dimensionalized, the analytic statistical method is applied to validate results from the DES method.

Steps 7 and 8 derives the system response, N, using equation (29), via statistical and DES methods, respectively. The statistical method calculates all component unit arrival times (f_{ix}) in a spreadsheet. The DES method captures all component unit arrival times at the very moment that each unit has reached its component process line "Record" module. The DES result is captured during each simulation run near the end of the simulation model. Once statistical and DES versions of the system responses, $N_{x,stat}$ and $N_{x,\ DES}$ have been collected, then the DES result can be validated by the next following two steps.

In step 9, correlations are calculated using results from both methods, $N_{x,stat}$, and $N_{x,DES}$, respectively. The correlation between the two methods is calculated using Pearson correlation coefficient, as shown in the equation (48).

$$r_{N_{stat}N_{DES}} = \frac{n\sum N_{x,stat}N_{x,DES} - \sum N_{x,stat}\sum N_{x,DES}}{\sqrt{n\sum N_{x,stat}{}^2 - (N_{x,stat})^2}\sqrt{n\sum N_{x,DES}{}^2 - (N_{x,DES})^2}} \quad (48)$$

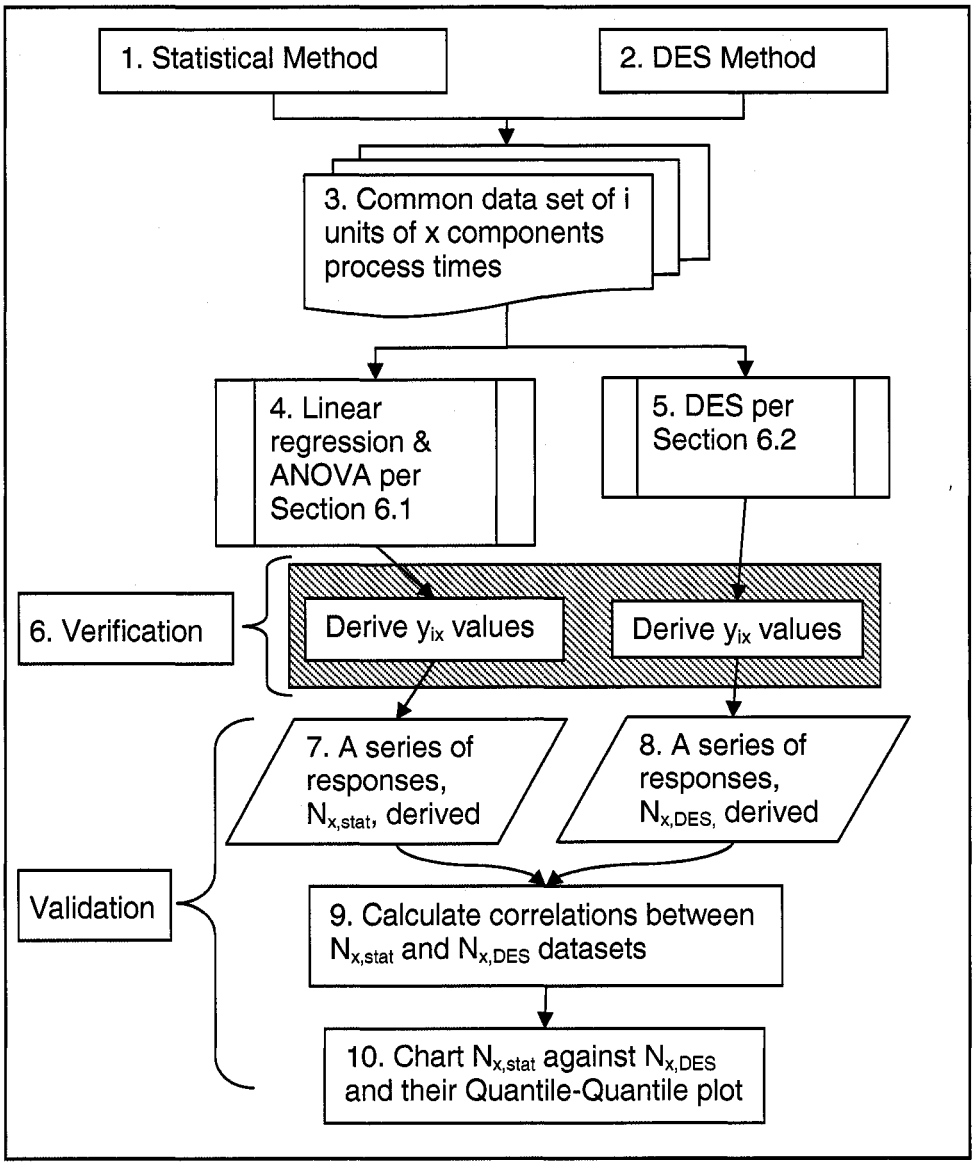


Figure 12 ASLC methods verification processes

The correlation result of both methods $r_{N_{stat}N_{DES}}$ will be between -1 and 1. If the value of $r_{N_{stat}N_{DES}}$ is in the positive range of 0.9 or higher, then there is a strong correlation between the statistical and DES methods. Hence, one may use the verified and validated DES method to further employ the ASLC model techniques for different hypotheses.

The step 10 plots $N_{x,stat}$ against $N_{x,DES}$, under a high percentage correlation, i.e. 90% or better. Data points of both methods shall overlap between results from both methods regardless of the trend of the data in N_x. For the same high percentage correlation, the Quantile – Quantile plot shall very closely show data points following a straight line diagonally across the graph with the slope value close to +1. These plots will be demonstrated in following chapters in which both methodologies are exercised on practical ASLC applications in large scale production systems.

Chapter 7: ASLC Applications

Chapters 4 and 5 reveal the ASLC model. Chapter 6 describes ASLC methodologies. This chapter applies ASLC methodologies based on the ASLC model and hypotheses that have previously been discussed.

7.1 ASLC Statistical Application

A set of data is needed to deploy an example statistical application using the stated ASLC model and methodologies. The full data structure available for this research is commercially proprietary in nature. It consists of the targeted component arrival dates for the first three hundred units (x = 300) of an airplane company. In order to use this set of proprietary data in the public domain, an extracted portion of the data for these three hundred production units has been non-dimensionalized using a series of non-disclosed transformations. Therefore, this data set does not reflect the absolute accuracy of the aircraft production system. However, they do relatively represent the nature of an ASLC featured large scale production system. The numerical data format xxxx.xx is used for each data point from Table 11 to 17. The numerical data format has the same resolution to the time unit of minute as the popular date and time format: mm/dd/yyyy hh:mm. Each production unit represents one aircraft. Each aircraft has thirty major components. The data set contains thirty components (i = 30) for each production unit. Thus, each scenario of production has 9000 data points in this system. In order to make a meaningful analysis of the ASLC model, this research utilized a sub-set of the full data. This sub-set data has one hundred (x = 100) production units that have three (i= 3) components each. Table 11 below represents the first ten process times (yi,x) and component production starting times (si,x) of the one hundred production units of the three components.

Step 1 of the ASLC statistical method, mentioned in Section 6.1, is to forecast the process times of the first production units. Process times of the first production units of three example processes are shown in Table 11 in row 2 from column 2 to 4, 506.47, 211.99, and 229.91 time units.

Step 2 is to forecast process times of the m^{th} production units. In this application, m=100. Process times of the 100^{th} production units of three example process are shown in Table 11 in the bottom row from column 2 to 4, 114.89, 65.00, and 106.04 time units.

Table 11 Process and starting times example

x	$y_{1,1} - y_{1,10}$	$y_{2,1} - y_{2,10}$	$y_{3,1} - y_{3,10}$	$s_{1,1} - s_{1,10}$	$s_{2,1} - s_{2,10}$	$s_{3,1} - s_{3,10}$	Finish Rate
1	506.47	211.99	229.91	13565.32	13843.15	13845.92	49
2	404.80	177.44	204.70	13703.97	13916.80	13913.69	38
3	355.26	159.90	191.21	13786.29	13968.88	13958.00	28
4	323.84	148.52	182.18	13824.44	13991.67	13977.67	19
5	301.39	140.25	175.47	13859.93	14013.84	13996.97	13
6	284.21	133.84	170.18	13890.87	14032.22	14015.03	12
7	270.45	128.65	165.82	13914.42	14050.27	14031.04	12
8	259.07	124.31	162.14	13938.52	14068.14	14046.90	12
9	249.43	120.61	158.96	13962.20	14085.98	14062.72	12
10	241.11	117.39	156.17	13982.28	14103.06	14078.00	12
...	...	...	...	...	...	...	...
100	114.89	65.00	106.04	14604.98	14654.10	14615.01	3

Step 3 is to identify their respective learning curve rate (c_i) and learning progress rates (b_i). Using equations (11) and (12), learning curve rates of these three example processes are: 80.00%, 83.70%, and 89.00%; learning progress rates of these three example processes are: 32.19%, 26.00%, and 17.00%, respectively. These rates are shown in Table 15.

Table 12 Targeted process starting and finishing times example

x	$F_{i,x}$	$S_{1,1} - S_{1,10}$	$S_{2,1} - S_{2,10}$	$S_{3,1} - S_{3,10}$
1	14099.00	13566.00	13843.00	13846.00
2	14137.00	13704.00	13917.00	13913.00
3	14165.00	13787.00	13969.00	13958.00
4	14184.00	13825.00	13992.00	13978.00
5	14197.00	13860.00	14014.00	13997.00
6	14209.00	13891.00	14032.00	14015.00
7	14221.00	13915.00	14050.00	14031.00
8	14233.00	13939.00	14068.00	14047.00
9	14245.00	13962.00	14086.00	14063.00
10	14257.00	13982.00	14103.00	14078.00
...	...	...	...	...
100	14720.00	14605.00	14654.00	14615.00

Step 4 is to obtain process times ($y_{i,x}$) by using equations (15) and (16). Component unit process times of these three example processes are in Table 11 in columns 2, 3 and 4.

Step 5 is to use equation (17) to derive targeted x^{th} unit of the i^{th} component production starting times ($S_{i,x}$), given the targeted component unit arrival times ($F_{i,x}$). An example of the data is in Table 12.

Lu, Petersen, and Storch (2007) outlined Engineering-To-Order (ETO) customization processes in an ASLC featured environment. Various characteristics of ETO processes were addressed in that paper. Most commercial aircraft and ocean vessels require certain levels of ETO processes throughout their product life cycle, mainly during the design and initial production phases. ETO processes were categorized into three distinct categories: minimum, medium, and large changes. Process times of those respective changes may have 2%, 8%, and 15% respective impacts, according to Lu, Petersen, and Storch. Therefore, the next step is to apply the stated method, in Section 6.1, to address these various levels of ETOs.

Table 13 Process and starting times example with 15% ETO

x	$y_{1,1ETO} - y_{1,10ETO}$	$y_{2,1ETO} - y_{2,10ETO}$	$y_{3,1ETO} - y_{3,10ETO}$	$s_{1,1ETO} - s_{1,10ETO}$	$s_{2,1ETO} - s_{2,10ETO}$	$s_{3,1ETO} - s_{3,10ETO}$	Finish Rate
1	558.61	216.67	233.83	13564.20	13843.28	13846.34	49
2	460.73	191.16	216.03	13704.68	13916.94	13913.05	38
3	380.40	165.66	208.88	13785.96	13968.51	13958.32	28
4	346.58	163.71	197.35	13824.63	13991.89	13978.58	19
5	306.49	140.27	176.35	13859.90	14013.99	13996.61	13
6	285.96	143.50	194.12	13891.45	14032.28	14015.01	12
7	310.36	140.59	166.27	13914.92	14050.11	14030.99	12
8	267.32	139.42	179.48	13939.05	14068.51	14047.26	12
9	268.68	124.58	168.05	13962.19	14086.07	14062.99	12
10	251.02	125.90	179.49	13981.91	14103.31	14078.33	12
...	...	...	...	...	...	...	...
100	119.29	73.10	118.18	14605.02	14654.00	14615.01	3

Step 6 applies randomization schemes to derive process and actual starting times with ETO considerations, using equations (26) and (27). This practice set the n% value at 10% in equation (27). An example of process and component unit production starting times ($s_{i,x}$) with 0% ETO is listed in Table 11 and with 15% ETO is listed in Table 13.

At this point, both actual process durations ($y_{i,x}$) and actual process starting times ($s_{i,x}$) are obtained for a non-ETO case and a 15% ETO case. Utilizing the stated ASLC model in the prior section, component unit actual ($f_{i,x}$) arrival times can then be obtained, as shown in columns 3, 4, and 5 in Tables 14 and 15.

Table 14 Responses, Actual and Targeted Arrival Times

x	$N_1 - N_{10}$	$f_{1,1} - f_{1,10}$	$f_{2,1} - f_{2,10}$	$f_{3,1} - f_{3,10}$	$F_1 - F_{10}$
1	44.97	14072.42	14054.03	14076.22	14099
2	22.45	14108.91	14094.73	14117.18	14137
3	20.01	14142.28	14128.34	14148.35	14165
4	19.30	14148.90	14140.56	14159.86	14184
5	18.19	14161.49	14154.27	14172.46	14197
6	19.16	14175.23	14165.87	14185.03	14209
7	18.39	14185.39	14178.53	14196.91	14221
8	17.20	14198.19	14192.07	14209.27	14233
9	15.32	14211.06	14206.36	14221.67	14245
10	13.65	14223.24	14220.40	14234.05	14257
...	...	...	...	...	...
100	2.02	14719.85	14719.02	14721.04	14720

Table 15 Responses, Actual and Targeted Arrival Times of 15% ETO

x	$N_{1ETO} - N_{10ETO}$	$f_{1,1ETO} - f_{1,10ETO}$	$f_{2,1ETO} - f_{2,10ETO}$	$f_{3,1ETO} - f_{3,10ETO}$	$F_1 - F_{10}$
1	72.34	14142.33	14069.99	14078.50	14099
2	31.73	14122.57	14104.05	14135.78	14137
3	34.69	14173.62	14138.93	14161.36	14165
4	27.24	14161.09	14158.53	14185.77	14184
5	23.39	14194.01	14173.29	14196.68	14197
6	24.09	14184.76	14176.19	14200.28	14209
7	26.51	14190.49	14185.78	14212.29	14221
8	36.95	14232.42	14195.47	14230.48	14233
9	16.20	14227.97	14221.36	14237.56	14245
10	24.83	14235.41	14220.45	14245.28	14257
...	...	...	...	...	...
100	15.73	14734.86	14720.70	14736.43	14720

Step 7 of the statistical method uses equations (18) and (19) to obtain targeted ($T_{i,x}$) and actual ($t_{i,x}$) total flow times, as shown in Tables 16 and 17. In this practice, each component unit has one single process from the start to finish as seen in Figure 9 (on

page 53). Therefore, the actual total process flow times ($t_{i,x}$) shall equal to their respective actual component unit process times ($y_{i,x}$). Hence, values in columns 2, 3, and 4 in Tables 16 and 17 shall equal to values in columns 2, 3, and 4 in Tables 11 and 13, respectively.

Table 16 Actual and targeted total flow times

x	$t_{1,1} - t_{1,10}$	$t_{2,1} - t_{2,10}$	$t_{3,1} - t_{3,10}$	$T_{1,1} - T_{1,10}$	$T_{2,1} - T_{2,10}$	$T_{3,1} - T_{3,10}$
1	506.47	211.99	229.91	533.00	256.00	253.00
2	404.80	177.44	204.70	433.00	220.00	224.00
3	355.26	159.90	191.21	378.00	196.00	207.00
4	323.84	148.52	182.18	359.00	192.00	206.00
5	301.39	140.25	175.47	337.00	183.00	200.00
6	284.21	133.84	170.18	318.00	177.00	194.00
7	270.45	128.65	165.82	306.00	171.00	190.00
8	259.07	124.31	162.14	294.00	165.00	186.00
9	249.43	120.61	158.96	283.00	159.00	182.00
10	241.11	117.39	156.17	275.00	154.00	179.00
...	...	...	...	...	...	...
100	114.89	65.00	106.04	115.00	66.00	105.00

Table 17 Actual and targeted total flow times with 15% ETO

x	$t_{1,1ETO} - t_{1,10ETO}$	$t_{2,1ETO} - t_{2,10ETO}$	$t_{3,1ETO} - t_{3,10ETO}$	$T_{1,1ETO} - T_{1,10ETO}$	$T_{2,1ETO} - T_{2,10ETO}$	$T_{3,1ETO} - T_{3,10ETO}$
1	558.61	216.67	233.83	533.00	256.00	253.00
2	460.73	191.16	216.03	433.00	220.00	224.00
3	380.40	165.66	208.88	378.00	196.00	207.00
4	346.58	163.71	197.35	359.00	192.00	206.00
5	306.49	140.27	176.35	337.00	183.00	200.00
6	285.96	143.50	194.12	318.00	177.00	194.00
7	310.36	140.59	166.27	306.00	171.00	190.00
8	267.32	139.42	179.48	294.00	165.00	186.00
9	268.68	124.58	168.05	283.00	159.00	182.00
10	251.02	125.90	179.49	275.00	154.00	179.00
...	...	...	...	...	...	...
100	119.29	73.10	118.18	115.00	66.00	105.00

Step 8 aims to obtain the main response (Nx), on-time arrival performance of component units, using equation (20). Values of the main response for the non-ETO case and the 15% ETO case are listed in Tables 14 and 15.

Steps 9 and 10 populate values of all variables from previous steps into tables. Tables 11 through 17 have a portion of these data presented. The rest of the data is listed in the Appendix A.

Steps 11 to 13 apply linear regression based on equations (28) and (29). Based on correlation analysis results, listed in Appendix B, factors y and t are having exact correlation. Since this model has a single process per component unit per learning curve, thus the sum of all process times per given component unit, t, is the same as y. Hence, the actual total flow time of a given unit of a component ($t_{i,x}$) has singularity with the process time of that component ($y_{i,x}$). Additionally, the actual ($s_{i,x}$) and the targeted ($S_{i,x}$) component unit production starting times are having a strong correlation (Appendix B). Thus, the linear regression model includes all of the s values, their respective y and F values, and the Finish Rate as factors. The response variable is the on-time performance, N. The linear regression formula is shown in equation (49) below.

$$\text{lm}(\text{formula} = N \sim F_{ix} + y_{1x} + y_{2x} + y_{3x} + \text{Finish.Rate} + s_{1x} + s_{2x} + s_{3x}) \quad (49)$$

Results of the linear regression:

Table 18 Linear regression results without any ETO influence

Coefficients	Value	Std. Error	t value	Pr(>|t|)
(Intercept)	1658.2511	2831.2923	0.5857	0.5595
F_{ix}	-0.2394	0.1380	-1.7346	0.0862
y_{1x}	3.2628	8.7633	0.3723	0.7105
y_{2x}	-10.8578	43.8397	-0.2477	0.8049
y_{3x}	2.4896	25.0572	0.0994	0.9211
Finish Rate	-0.3452	0.2122	-1.6267	0.1073
s_{1x}	0.0914	0.0857	1.0660	0.2892
s_{2x}	-1.1175	0.1841	-6.0693	0.0000
s_{3x}	1.1617	0.1334	8.7104	0.0000

Residual standard error: 2.122 on 91 degrees of freedom
Multiple R-Squared: 0.9389
F-statistic: 174.6 on 8 and 91 degrees of freedom, the p-value is 0

Residual of this liner regression model has a relatively high degree of freedom as compared among all factors. However, this model also showed a multiple R-Squared

value at 93.89%, which means the percentage of the response can be explained by those included factors. Their F and P values are discussed in the next step.

Step 14 is to perform an analysis of variance (ANOVA). This step continues from previous steps with the same group of factors. The ANOVA result is in Table 19 below.

Table 19 Analysis of Variance Table without any ETO influence

Factors	DF	Sum of Sq	Mean Sq	F Value	Pr(F)
F_{lx}	1	4674.319	4674.319	1037.991	0.0000
y_{1x}	1	8.210	8.210	1.823	0.1803
y_{2x}	1	936.789	936.789	208.026	0.0000
y_{3x}	1	6.214	6.214	1.380	0.2432
Finish Rate	1	5.317	5.317	1.181	0.2801
s_{1x}	1	261.455	261.455	58.059	0.0000
s_{2x}	1	57.880	57.880	12.853	0.0005
s_{3x}	1	341.667	341.667	75.871	0.0000
Residuals	91	409.794	4.503		

The data structure in utilizing the stated ASLC model provides a methodology to separate component arrival time deviations at the final integration. The component starting times ($s_{i,x}$) fluctuate naturally due to many real world reasons and the ETO effect contributes to the variation of the process times ($y_{i,x}$). Thus, component arrival times ($f_{i,x}$) can have at least one more layer of resolution, which contributes to the analysis of the overall component on-time performance in the whole mass-customized ETO production system.

This section outlines the ASLC model data structure with sanitized real world numbers. However, relative significance among different levels of ETOs and component starting, processing, and/or finishing times is not noticeable. The next step depicts results by employing the data structure with linear regression and ANOVA of different ETO levels.

Step 15 is to perform hypothesis testing by extending and repeating all of the above steps on different levels of ETOs. As stated before, many variables are closely related among themselves and shown as singularities in the linear regression analyses.

Equation (50) shows when ANOVA yielded coefficients of factors are applied to equation (49).

$$\begin{aligned} &\text{lm(formula} = N \sim 1658.251 - 0.239\, F_{ix} + 3.263\, y_{1x} - 10.858\, y_{2x} + 2.490\, y_{3x} \\ &- 0.345\, \text{Finish.Rate} + 0.091\, s_{1x} - 1.118\, s_{2x} + 1.612\, s_{3x}) \end{aligned} \tag{50}$$

When there is no ETO effect (0% ETO), F-statistic is at 174.6 on 8 and 91 degrees of freedom with the p-value at 0. Therefore, the null hypothesis (equation (21)) does not have sufficient evidence to be acceptable and thus can be rejected by the first alternative hypothesis, as stated in equation (22).

Table 20 outlines results from the ASLC model among three ETO cases with same set of factors as the previous steps.

Table 20 Linear regression and ANOVA results

Factors	0% ETO		2% ETO		8% ETO		15% ETO		Learning Curve Rate (c_i)	Learning Progress Rate (b_i)
	F Value	P	F Value	P	F Value	P	F Value	P		
F_{ix}	1037.991	0.000	736.358	0.000	489.049	0.000	244.485	0.000		
y_{1xETO}	1.823	0.180	3.888	0.052	1.933	0.168	0.438	0.510	80.00%	32.19%
y_{2xETO}	208.026	0.000	15.387	0.000	35.605	0.000	48.941	0.000	83.70%	26.00%
y_{3xETO}	1.380	0.243	2.558	0.113	73.104	0.000	56.751	0.000	89.00%	17.00%
Finish Rate	1.181	0.280	44.628	0.000	48.148	0.000	6.955	0.010		
s_{1x}	58.059	0.000	18.405	0.000	2.375	0.127	4.601	0.035		
s_{2x}	12.853	0.001	20.979	0.000	4.673	0.033	2.994	0.087		
s_{3x}	75.871	0.000	94.569	0.000	55.163	0.000	19.261	0.000		
R^2	0.939		0.912		0.886		0.809			

The y_{1xETO} had the best learning progress rate (b_i) and the best learning curve rate (c_i) among the three, and displayed as an insignificant ($P > 0.05$) factor in all ETO cases. The y_{3xETO} had the worst learning progress rate (89%) among the three, and was a significant factor (P values at 0.000) when the ETOs are at levels 8% and 15%. Process times may seem to be a critical factor in a production system. This study showed that until the level of ETO increases to certain percentages, such as 8% in this case, that 9% extra learning curve rate between y_{3ETO} (c_i = 89%) and y_{1ETO} (c_i = 80%) is not significant. In the case of a faster learning process, such as the y_{1ETO}, additional customization related fluctuation does not influence the response factor as much as the overall product Finish Rate.

At 2% ETO, the y_{1xETO} F and P values (F = 3.888, P = 0.052) were not between the respective F and P values of 0% and 8% ETOs. This can also be seen as the 0% ETO y_{1xETO} F and P values (F = 1.823, P = 0.180) are not as significant as the suggested trend from rest of the y_{1xETO} F and P values of 2%, 8%, and 15% ETOs. The low y_{1xETO} F-value (or high P-value) at 0% ETO shows that the y_{1xETO} is not as significant as other factors when there is 0% ETO related customization in the system. At 0% ETO, y_{2xETO} is the most significant factor among all three processes duration factors. This can be explained by viewing Table 11, that y_{2xETO} has the shortest process durations among y_{1xETO}, y_{2xETO}, and y_{3xETO}. Since the shorter the process duration, the less the influences from the ASLC effect, which can be explained by examine equations (5), (6), and (7).

The increase of the ETO somewhat decreased the significance of the s_{ix} factors, mostly in s_{2x}, which P-value increased to 8.7%, greater than 5%, at the 15% ETO level. It is notable that the R^2 value consistently decreased as the ETO percentage increased. Intuitively, this makes sense. The sensitivity of the Finish Rate increases quickly (from 0.280 to 0.000) as soon as there was 2% ETO in the system.

The Finish Rate showed as a pivotal factor that may have more effect in the overall system performance than both process ($y_{i,x}$) and starting times ($s_{i,x}$) in this case. Again, the Finish Rate is the time period between any two adjacent final integration starting times. The Finish Rate can also be treated as the whole system takt time or heart beat rate. Faster demanded system finish rates combined with higher ETO percentages, make the system more sensitive to the component process times and the less sensitive to the component starting times. This finding shows that for faster demands of highly customized products, the ability to produce individually customized components in a timely manner becomes very important.

This section outlined a feasible statistical application in a system with ASLC effects. Additionally, the method is deployable to analyze ETO customization in an ASLC featured large scale production system. The next section addresses a discrete event simulation method to analyze the system by using the same data set.

7.2 ASLC Discrete Event Simulation Application

This section applies the ASLC model by using a discrete event simulation (DES) method. A high-level flow diagram (Figure 10) in section 6.2 outlined major steps in a simulation model. This section addresses discrete event simulation steps in more detail. All data used in this section is the same set as the previous section. Therefore, the verification between the statistical and discrete event simulation methods can be discussed later.

These following steps enable the modeling of an ASLC featured LSPS by using a discrete event simulation method. The discrete event simulation method applied in this research uses the Arena software from the Rockwell Automation Technologies, Inc.

Step 1 is to create one entity by using one Create module. Each Create module represents one component type. There are as many Create modules in the model as quantities of different components in the production system. In the Create module, one entity is generated only once at the time zero. Due to the nature of the ASLC in the system, entity arrival frequencies are not the same within each component and among components. Step 2 in the statistical method generates needed process starting times for all units in all components (Table 11). This set of process starting times will be accessed in step 3 later.

Step 2 is to assign two variables associated with the component which was created from the previous step. For example, using equation (30) and (31):

varSlopeMainGear = log (80/100)/log(2) (51)

where the learning curve rate (c_l) of the Main Gear is at 80%.

varMainGearLN1 = 10 ** (log(varMainGearLN100) – varSlopeMainGear * log(100)) (52)

where varMainGearLN100 is the process time of the 100^{th} unit (m = 100) of the component "Main Gear" in this case. The variable value of the varMainGearLN100 is stored in the Variable data module. Equation (52) in the Assign module generates the

value of the process time of the first Main Gear production unit (varMainGearLN1). The varMainGearLN1 will be used later in the model.

Step 3 reads in schedule data from an external file. This external data file can be a database, spreadsheet, or a text file. Here an Excel spreadsheet is used. This step requires links between the simulation model and the spreadsheet file to be established via the File data module. In the spreadsheet file, process times of all units of each component must be named with a unique identification in their Define Name (Insert | Name | Define). For example, units of the Main Gear starting times are listed in a column and are named as: "s11_start_time".

In the File data module in the simulation model, the name of the spreadsheet is listed in the "Operating System File Name". Recordsets of the File data module contain two fields: Recordset Name and Named Range. The Recoreset Name is user defined in the simulation module, e.g.: "Recoreset_s11". The Named Range has to have the identical characters as the Define Name in the spreadsheet; in this case, it is "s11_start_time".

One spreadsheet file may contain all needed data values and one File data module may contain all needed Recordsets that match their respective data values. This data matching between the spreadsheet and the simulation model can be executed via both Read and Write methods. Within the ReadWrite module, the Type field is: Read from File. The RecordsetID in this case is "Recordset_s11", which is the same one as defined earlier in the File data module.

After each ReadWrite model that reads in data from the spreadsheet, a Separate module duplicates the original entity into two entities. One of the two entities moves forward to a subsequent module and the other one routes right back to the ReadWrite module again. The type of the Separate module is "Duplicate Original". The percent to duplicate is 50%. The number of duplicates is one. Therefore, the ReadWrite module can read in all data in the Recordset one at a time. Each moving forward entity representing a unit of a component will carry an attribute with a value that was assigned by the ReadWrite module.

Step 4 is to store starting times of each unit per component. This step takes place together with reading-in the component unit production schedule from a spreadsheet. The Assignments field in the ReadWrite module needs a new "Attribute" to store the individual starting time value for each component. This attribute is named "*MainGearTime*" in this example. The attribute MainGearTime stays with each entity throughout the whole DES model.

Step 5 is to release the actual process starting time per component unit. This step uses a Delay module. The Delay Time in this case is "*MainGearTime*", which has to be identical to the assigned attribute name in the ReadWrite module in the previous step.

Steps 2 to 5 are to bringing in the stochastic nature of actual starting times ($s_{i,x}$) with their learning effects into a discrete event simulation model.

Step 6 is to assign a serial number to each component unit in the system. Each component has its own serial number stream. There are multiple serial number streams in the system representing multiple components. Serial numbers are assigned by using the Assign module. One Assign module is used for all units of one component. Two statements are used in an Assign module.

The type of the first one is a Variable. The Variable Name is: "varPartMainGear". Using equation (32), varPartMainGear = varPartMainGear +1. The initial value of the variable varPartMainGear is zero, defined in the Variable data module. The type of the second one is an Attribute. The Attribute Name is: "attSNy11". Using equation (33), attSNy11 = varPartMainGear. Therefore, every entity passing through this Assign module will have an assigned serial number style attribute with a sequencing increment value of one.

Step 7 routes entities to their respective process modules. This step uses both the Route and the Station modules. The Route module can have a unique name to represent the entity that it is routing. Route time is optional. The Destination Type is: "Station". The Station Name must match the entity destination station. For example, the Station Name in a Route module is: "StnMainGearOrder". And then in the Station module, the Station

Name shall also be "StnMainGearOrder". The Station Type in the Station module needs to be "Station", to match the Route module.

In DES models, representing a simple system with just a few processes, there is no advantage in using the Route | Station module combination. To model a large scale production system, the Route | Station module combination enables the simulation model to represent more complicated scenarios in a more organized way. This is because all production schedule data input can be grouped together before the Route modules. Further detail in using Route | Station module combinations is beyond the scope of this research.

Step 8 is an optional step because it deals with modeling of potential product customization and/or disruption events. A Decide module is used to represent this step. One or more variables, such as varTriggerLN1 as seen in equation (34), are defined in the Variable data module. Entity attribute, such as attSN, is defined in the previous step.

The type of the Decide module is: "Two-way by condition". The condition is an Expression, which can be predefined in the Expression module with a unique expression name, or use the content as shown in the equation (34). This step examines whether the current entity serial number matches any of the triggering event numbers. If yes, then the entity will be routed to step 9. If no, then the entity will be routed to step 10.

Step 9 is also an optional step. This step examines if the trigger event from the previous step is a new event or not. A Decide module is used in this step. The decision type is: "Two-way by condition". It checks the value of the entity Attribute as stated in equation (35). If it is a new triggering event, then the entity will be routed to step 11, otherwise it will be routed to step 12.

Step 10 is an optional step that follows the step 8 when the triggered change is not a new one. Step 10 uses an Assign module, that set the disruption attribute to zero as seen in equation (36). The entity will then go to the next process module, step 13, without going through step 11 and 12.

Step 11 is an optional step that follows step 9 when the triggered event is a new one. This step uses an Assign module with six assignments. The first five of the six assignments are listed from equation (37) to (41). Since entities routed to this assign module are the ones with new triggering events, once they leave this current step, their status will not be new. The assignment in equation (41) sets the attNewTrigger attribute to the value of 1, which means it is not a new triggering event anymore.

Step 12 is an optional step that follows step 9 when the triggered event is an old one. Similar to the previous step, step 12 uses an Assign module with five assignments. As in step 11, the triggering duration attribute will be reduced by one occurrence as stated in the equation (40). However, the new trigger attribute, attNewTrigger, will be set to the value of the triggering duration attribute.

attNewTrigger == attDuration (53)

The attribute attDuration indicates how many times triggered events will take place in the system. attDuration reduces by one count each time an entity goes through either step 11 or step 12 as seen in equation (40). After all of the triggering events have gone through the system, then the attNewTrigger value will be zero, according to equation (40).

A common assignment among steps 10, 11, and 12 is an attribute to have a time stamp on the current entity before it goes to subsequent process modules. This attribute is named as "attFAstart" with a value of "TNOW". TNOW rerepresents current time in the simulation model when an entity is passing through its respective Assign module. The attribute, attFAstart, will stay with the entity throughout all subsequent processes. Therefore, later in step 14, the process time of each entity can be collected.

Step 13 represents the major process steps that represent component unit process times with ASLC effects. This step is represented by a Process module. The Delay Type in the Process module is an Expression. The Expression has to be totally stated within this Process module for the component unit serial number, attSN, to influence the process time on the learning curve.

Combining equations (42), (43), and (44), the content of a process expression can be stated in equation (54) below.

(Maximum (TRIA
(0.9 * (10 ^ ((varSlopeFinalAssembly * log(attSN)) + log(varFinalAssemblyLN1))),
(10 ^ ((varSlopeFinalAssembly * LOG(attSN)) + log(varFinalAssemblyLN1))),
1.1 * (10 ^ ((varSlopeFinalAssembly * LOG(attSN)) + log(varFinalAssemblyLN1)))),
TRIA (0.9*varFinalAssemblyLN100, varFinalAssemblyLN100, 1.1 *
varFinalAssemblyLN100))) * (attDelta1 * attDisruption + 1) (54)

Equation (54) is a discrete event simulation application representation of equations (45), (46), and (47) in Section 6.2. The whole equation (54) represents the process time ($y_{i,x}$) of a component unit with optional distribution and customization event triggering.

Where:

varSlopeFinalAssembly is b_i
varFinalAssemblyLN1 is a_1
varFinalAssemblyLN100 is $y_{i,100}$

More detailed explanation and codes of this step are listed in Appendix D and E.

Step 14 records process times to component specific record sets. This step uses Record module to tally a time series. The type of the Record module is Expression. The Value of the Expression is "TNOW – attFAstart" with a Tally Name: "TotalFATime". The attribute "attFAstart" has been previously assigned with a time value before entities reach their respective processes. This step uses the current simulation system clock, TNOW, to obtain total process times per entity type, or component units.

A Statistic data module with the same Tally Name, TotalFATime, is needed to output the time collected from the Record module to an output file. The format of the output file can be either a common delimited text file or software specific format binary file.

ReadWrite modules can be used to write component unit finish times. One ReadWrite module is used per component type. The output file name and data Recordset IDs are defined in the File data module previously described in Step 3.

Process times reduce according to their ASLC effects in a large scale production system. Thus, data captured during the beginning stage of each simulation run is important. As a result, the discrete simulation model method does not have a warm-up period. The simulation run terminating condition can be set when a certain number of products have been produced, such as: "Final Field.NumberOut >= 350", where "Final Field" is the name of a process. Another method of controlling the terminating condition is to have fixed rows of records in the data file for the simulation model to read in step 3.

Detailed simulation model codes with regard to all of the above steps are listed in Appendix C, D, and E.

7.3 ASLC Application Verification and Validation

This section applies the verification method, outlined in section 6.3, to verify and validate discrete event simulation results. Two preceding sections applied both statistical and simulation methods and arrived with two sets of component unit process times and arrival on-time performance ($N_{x,stat}$ and $N_{x,DES}$) measures. They are listed in Appendix F. These two set of measures represent results from steps 1 to 8 in figure 12 of section 6.3.

If the ASLC model scheme were correct, and the discrete event simulation (DES) method applied all formulations accurately, then the statistical method result should match the discrete event simulation method result when there is no customization involved in the system. Step 6 of the verification and validation method is to verify whether the DES method generates ASLC featured component unit process times accurately.

Table 21 Correlations between the statistical and DES ASLC methods on component unit process times

Correlations		
y11	y21	y31
0.999999806	0.999999993	0.9999999

High correlation values in table 21 demonstrate that the DES method accurately derives process times for all component units. Therefore, the lengthy equation (54), of the DES method, in all process modules executes correctly and is verified.

For the purpose of validation, the system response, N, is collected in both methods as described in section 6.3. Table 22 shows partial results of the system response and unit process times for verification and validation purposes. The full data set for the rest of the table 22 is listed in appendix F.

Table 22 Partial results from both methods for verification and validation

m	N_DES	N_Stat	y11_DES	y11_stat	y21_DES	y21_stat	y31_DES	y31_stat
1	45.0663	42.8324	506.000	506.470	212.000	211.990	230.000	229.910
2	25.2227	24.2083	404.800	404.800	177.444	177.440	204.700	204.700
3	19.9807	21.4628	355.264	355.260	159.904	159.900	191.211	191.210
4	20.2845	20.0685	323.840	323.840	148.521	148.520	182.183	182.180
5	18.1518	17.7050	301.392	301.390	140.252	140.250	175.475	175.470
6	19.2431	19.2904	284.212	284.210	133.839	133.840	170.178	170.180
7	17.8192	18.3140	270.452	270.450	128.647	128.650	165.824	165.820
8	16.5384	16.5144	259.072	259.070	124.312	124.310	162.143	162.140
9	15.4060	15.8563	249.433	249.430	120.609	120.610	158.964	158.960
10	13.0086	13.7235	241.114	241.110	117.391	117.390	156.173	156.170
...	...	...	...	...	...	...	...	...
100	2.0100	2.1237	114.893	114.890	65.003	65.000	106.043	106.040

Table 23 Correlations between the statistical and DES ASLC methods on component unit on-time performance

	N_Stat	N_DES
N_Stat	1.0000000	0.9991649
N_DES	0.9991649	1.0000000

Utilizing the same correlation approach, table 23 shows system responses of both methods. Clearly both methods yielded highly correlated response values. Furthermore,

if responses of both methods are plotted as seen in the Figure 13 below, differences between these two methods are reasonably acceptable. Therefore, the DES method is successfully validated by using an analytic statistical method.

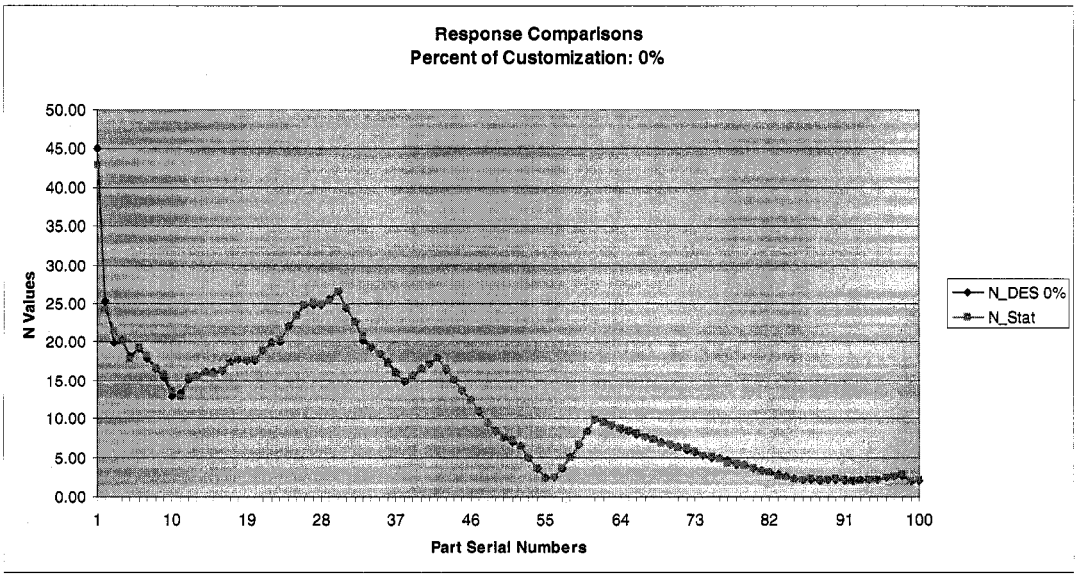


Figure 13 Response comparisons with no customization

The N value reduces initially, in figure 13, then increases, followed by a couple more ups and downs. The trend of N values from both methods is not an expected finding; even though there is a good match between results from the statistical and the DES modeling methods.

Figure 14 further illustrates that the ASLC model through the exercise of DES modeling can be validated by the stated statistical method. Their normal quantile - quantile plot shows a very good fit between the two methods.

Tables 21, 22, 23, Figures 13, and 14 shown that the DES method is verified and validated by the statistical method within a very reasonable margin. The statistical analysis method may be very practical for a small production system with ASLC. The exercise of using the DES method to model different levels of mass customization may be more efficient and practical in a large scale production system with ASLC.

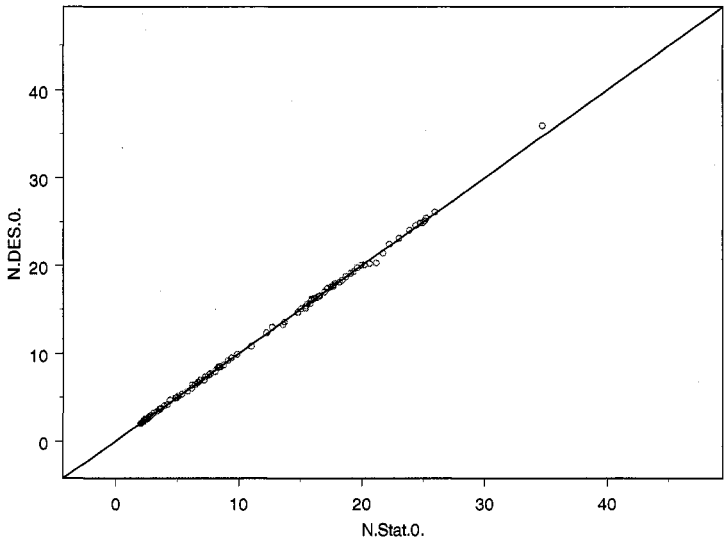


Figure 14 The Quantile - Quantile plot of verification methods

Section 5.3 outlined a null hypothesis (equation (21)) and four alternative hypotheses (equations (22), (23), (24), and (25)) in an ASLC featured large scale production system. Both statistical and DES methods showed N values other than zero by a large margin. Therefore, the first alternative hypothesis rejects the null hypothesis.

The next section discusses the rest of the alternative hypotheses by applying the verified and validated DES methodologies.

7.4 Alternative Hypotheses Evaluations using the DES Methodology

The ASLC featured large scale production system desires to have component units of the same serial numbers arrive at their destination simultaneously. The null hypothesis, stated in section 5.3, is that they will arrive with zero time difference (equation (21)). Evaluations of the first alternative hypothesis (equation (22)) using both statistical linear regression and discrete event simulation methods (sections 7.1 and 7.2) rejects the null hypothesis. This section addresses the other three alternative hypotheses.

The second alternative hypothesis, Ha_2, (equation (23)) calls for the same learning curve improvement rate among different components. The Ha_2 learning curve rate is set 85%,

about the average of c_1 (80%), c_2 (83.7%), and c_3 (89%) in table 20. The new set of c_i's is equal to 85%. DES model is revised per equation (54) for all components.

Results of the Ha_2 are listed in Appendix G and figure 15 below.

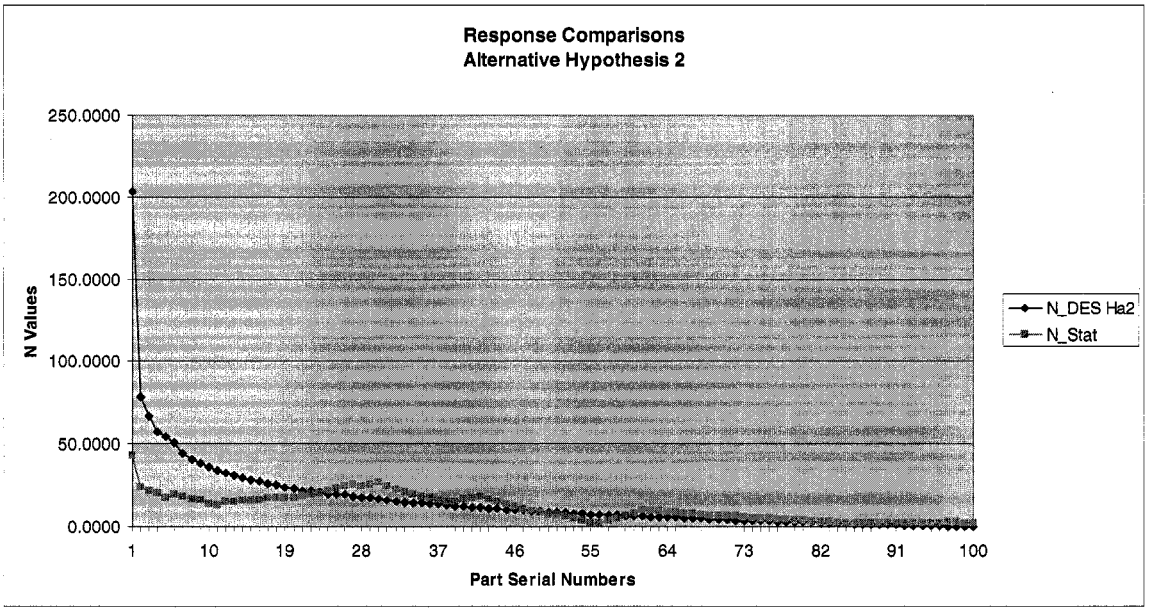


Figure 15 Simulation Results of the Alternative Hypothesis 2

In figure 15, the square shaped data line represents the same statistical result (in Figure 13) from the first alternative hypothesis. The diamond shaped data line represents results from the second alternative hypothesis. This alternative hypothesis has the same learning curve rate for all three processes. Process times of all three processes are still different among them. It is noticeable that having the same learning curve rate among components did not help in brining the response, N, close to zero at the beginning, because this alternative hypothesis does not address individual process durations, $y_{i,x}$. Differences among three process times are more noticeable, reflected in the high N values, in earlier part serial numbers in figure 15. The response, N, reduces to much lower values as the learning curve effect gradually taking places in all three processes toward higher part serial numbers, over the right side of the figure 15. However, the same learning curve rate does help in making the response value smoother and sometimes closer to zero than the

other data line. The next two hypotheses examine effects of reducing differences among process durations.

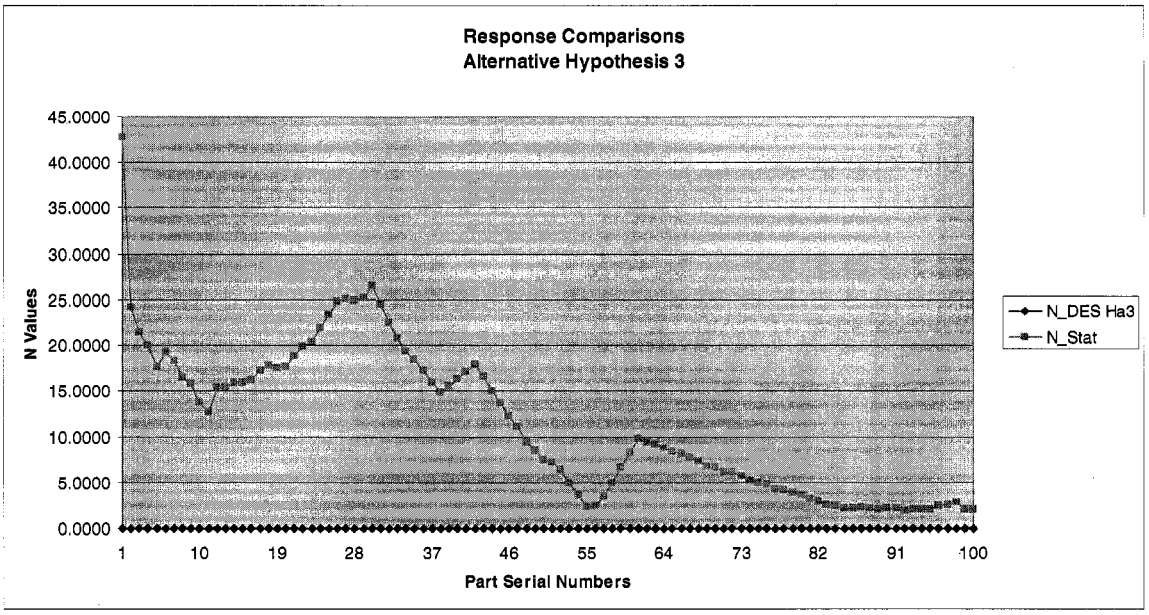


Figure 16 Simulation Results of the Alternative Hypothesis 3

The third alternative hypothesis, Ha_3, equation (24), reduces process duration differences within each component and reduces finishing rate fluctuations. Thus, learning curve rate within each component process is removed. The Ha_3 assumes a deterministic production system. The finishing rate is set at five days in this hypothesis. This means one final product delivery every five working days. However, process times among components are different. Due to the deterministic nature of this hypothesis, give different process times among components, the lead times are the same among all units within each component.

Appendix H shows numerical results of the third alternative hypothesis. Figure 16 shows the response, N, represented by the lower data line, of the third alternative hypothesis. The response value is at zero for all units of all components in this hypothesis. The upper data line in figure 16 represents the statistical method results from the first alternative hypothesis.

The deterministic nature of the third alternative hypothesis removes the stochastic effect from the production system. Additionally, the given consistent final production rate removes the asynchronous nature of the production system. Therefore, it is expected to see a flat line response value at zero for this hypothesis.

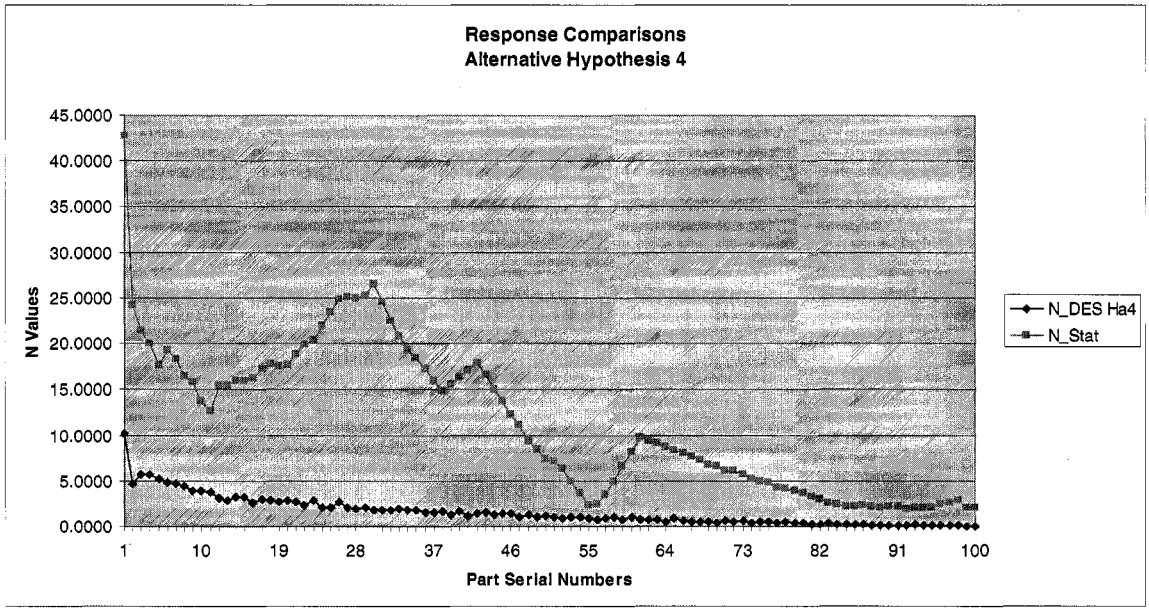


Figure 17 Simulation Results of the Alternative Hypothesis 4

The fourth alternative hypothesis, Ha_4, equation (25), has learning curve rates of all components close to 100% while keeping the stochastic nature of process times (equations (26) and (27)). Learning curve rates are set at 99%, 98.5%, and 98% for three components in the model.

Appendix I shows numerical results of the fourth alternative hypothesis. Figure 17 shows the response, N, represented by the lower data line. Points fluctuate up and down along the lower data line representing the stochastic nature of the system. Since the learning curve effect has been taken out in this hypothesis, it is expected to see a nearly flat data line that also approaches to zero as more units are produced at later serial numbers.

An analytic statistical method successfully verified and validated the DES method in the first alternative hypothesis. The other three hypotheses also apply the DES method in an ASLC featured large scale production system. Ha1 has the full ASLC effects. Ha2 uses the same learning curve rates that show consistent improvement. Ha3 took the whole ASLC out of the system and shows perfect on-time performance in the system. Ha4 illustrates what happens if there is no learning curve effects in the system.

The verification and validation combined with findings of hypotheses testing give evidence that the DES method can correctly model events in an ASLC featured large scale production system. The next chapter utilizes the DES methodology on case studies. All case studies are in large scale production systems with ASLC effects.

Chapter 8: ASLC in Large Scale Production System

8.1 ASLC Application Approach

The ASLC model, application methods, verification of methods, and hypotheses testing have been addressed in previous chapters. This chapter extends the ASLC application with two case studies.

In an ASLC featured large scale production system, dynamic interactions among all entities can be a challenge to represent via a deterministic model. A stochastic simulation modeling approach thus enables a framework to represent such a system dynamically. Since the DES methodology has been verified with an analytical statistic method. Using DES to model an ASLC featured large scale production system can be divided into, but not limited to, the following scenarios:

- Products' time to market, based on various design configurations.
- Logistics performance based on levels and complexities of supplier involvement in both design and manufacture and on product configuration-driven supplier allocations.
- Time consequences, based on given levels of product customization configurations and relative sequences and schedules of multiple customization requests during production processes.
- Overall system performance, based on choices of suppliers and their respective levels of involvement.
- Miscellaneous end-item, component, and final product integration related production and business processes.

The next section discusses couple case studies using the DES methodology.

8.2 Case Study One

The first case is a study of engineering changes applied to illustrate the stated ASLC model (Lu, Petersen, and Storch 2007).

Aalborg Industries is in the ocean vessel steam boiler business. When Aalborg Industries sells a steam boiler plant, the plant is configured in a product configuration system, which automates parts of the sale-to-delivery process. The product range implemented in this system comprises so-called standard products for which a number of engineering processes are automated. Being Engineer-to-Order products however, customers often request features for the products which are not included in the standard product portfolio. These range from minor adjustments to major structural changes of the main components of the plant. In this case study, these engineering changes are categorized into three distinct categories: minimum, medium, and large changes. Examples of changes within each of these categories are presented below.

Minimum change

A typical example of a minimum configuration change is changing the placement of a valve. The reason for this kind of change is typically linked to the layout and piping of the boiler room in the ship where the boiler plant is to be installed, and if this layout is changed, the valve placements may also be required to be changed. Depending on the specific case, this kind of change in a configuration will increase the total engineering duration for the plant up to approximately 2%.

Medium change

An example of a medium change of a configuration of a steam boiler plant is a reconfiguration of the burner. Each burner is designed specifically for working with a specific boiler and a specific fuel type. If a customer wishes to use a different type of fuel, this change in requirements has an impact on the burner design. The impact of this type of configuration change will vary depending on how much the new configuration deviates

from the standard product portfolio, but it may increase the plant engineering time duration up to 8%.

Large change

More significant changes also occur, although less frequently than the minimum and medium engineering changes. One example of this is changes to the exhaust gas section of waste heat recovery boilers. If the customer changes the type of main engine, the characteristics of the exhaust gas also are likely to change. This subsequently changes the requirements for the exhaust gas section, which therefore must be reengineered. This kind of configuration change can increase the total duration of the plant engineering as much as 15% since this is basically a redesign of one of the major components in the plant.

Since the DES method has been previously verified and validated, it is applied in this case study. This case study is based on the same data set as in chapters 6 and 7. The 0% change or 0% ETO level was used for the null and the first alternative hypotheses analyses. When there is a minimum change such as 2% in ETO, the ASLC featured system response as compared with the 0% ETO is plotted in figure 18.

Figure 19 compares the medium ETO change at 8% with the 0% ETO. It is noticeable that the 8% ETO system response fluctuates more than the 2% ETO. Figure 20 compares the large ETO change at 15% with the 0% ETO. The 15% ETO has much greater fluctuations than the 0% ETO. The 0% ETO response data is plotted as the common reference line among different percents of ETOs in figures 18, 19, 20, and 21. In figure 20, the 15% ETO system response data line hardly portraits the underlined 0% ETO performance in an ASLC featured large scale production system. Figure 21 demonstrates all four different levels of ETOs and their predicted on-time performances from the first unit to the 100th unit. Clearly, more ETO customization will result in more variations in the system.

The DES method is very applicable and effective in analyzing an ASLC system such as this ETO case study of the Aalborg Industries.

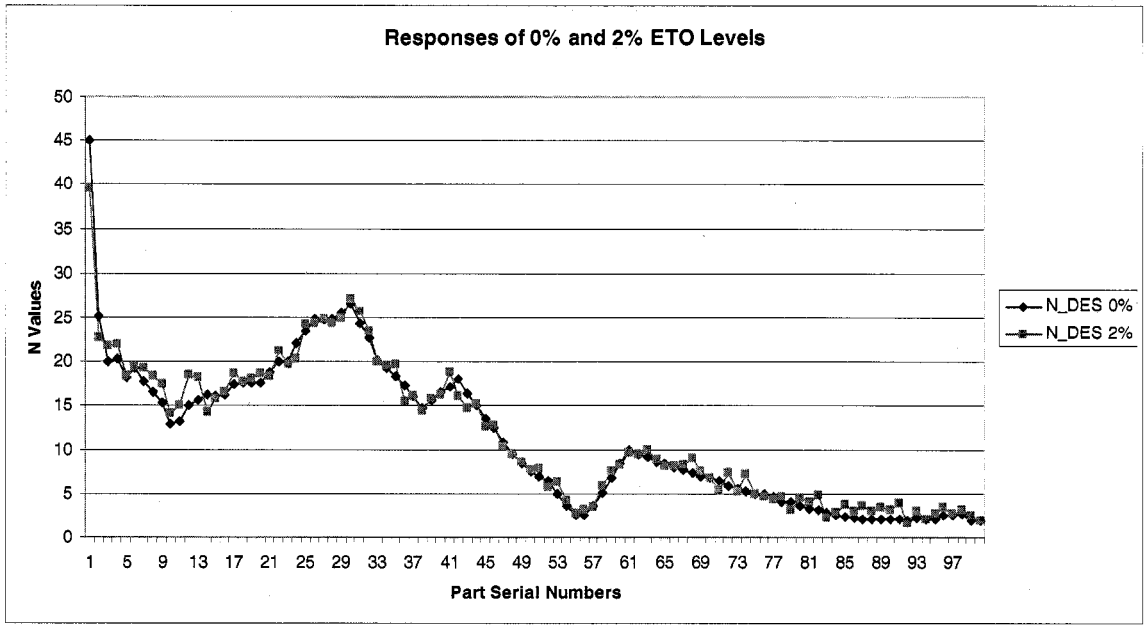


Figure 18 Case study 1: response of 0% and 2% ETO levels

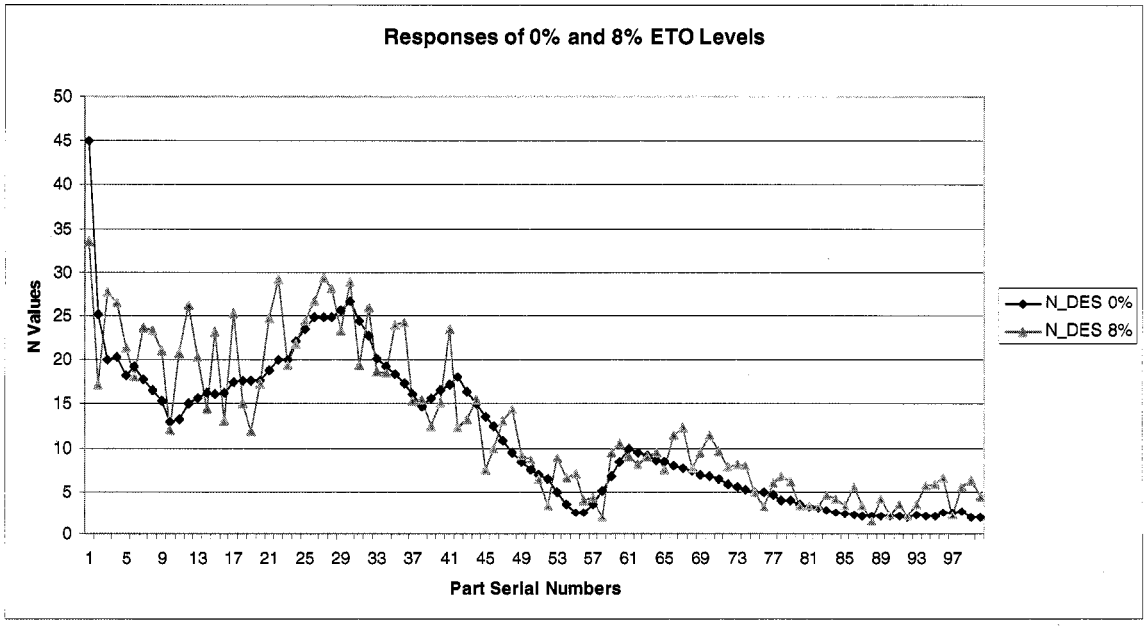


Figure 19 Case study 1: response of 0% and 8% ETO levels

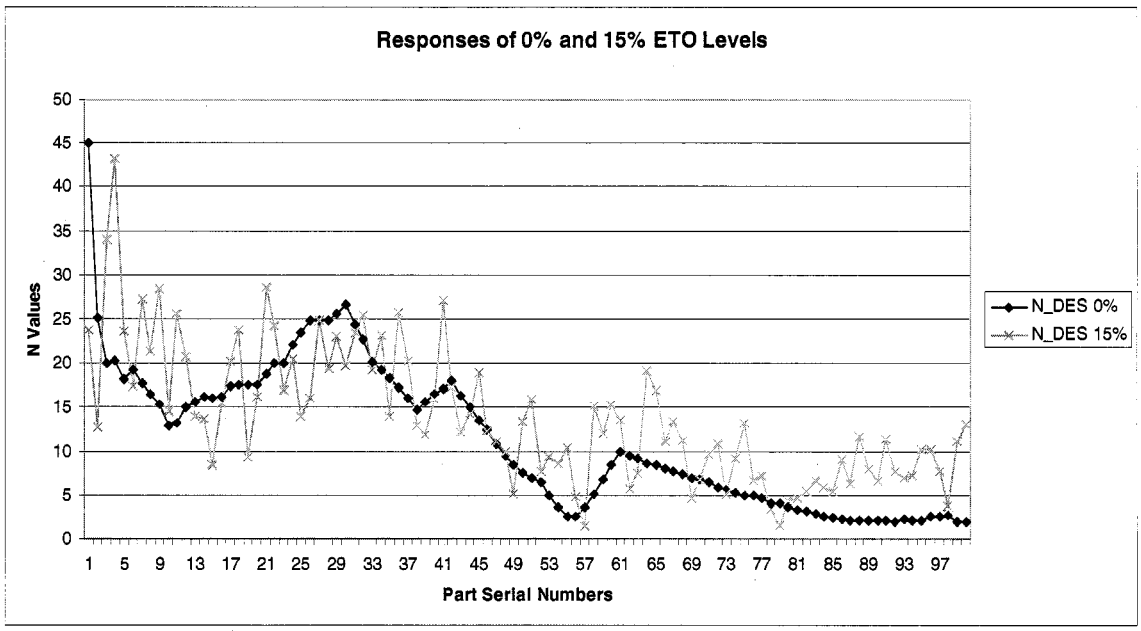


Figure 20 Case study 1: response of 0% and 15% ETO levels

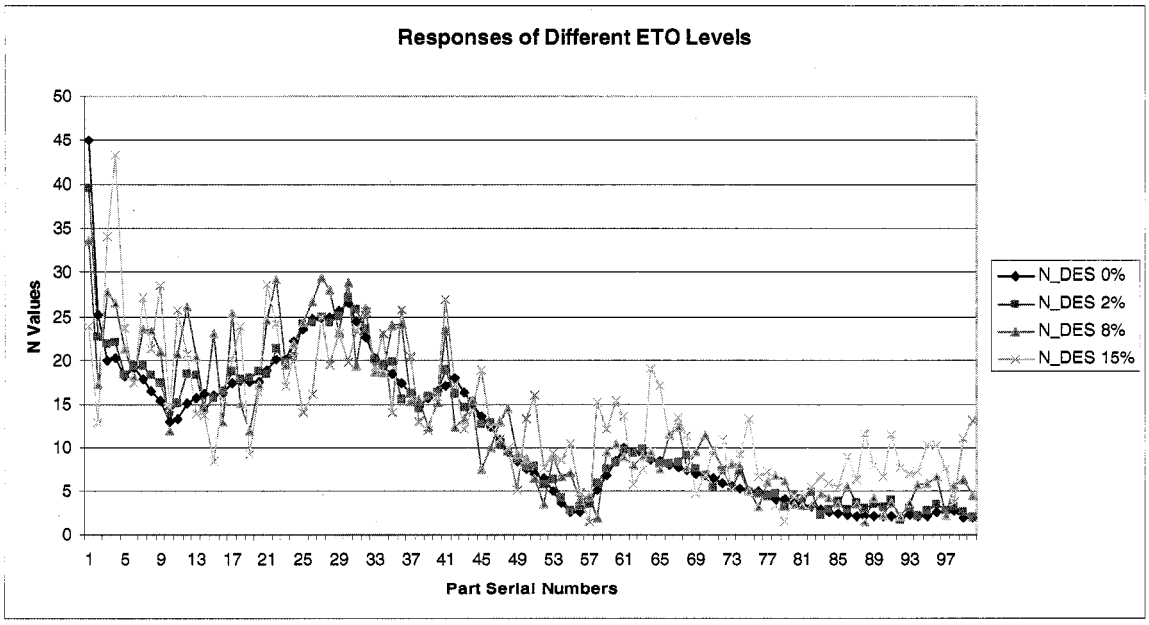


Figure 21 Case study 1: response of different ETO levels

8.3 Case Study Two

The second case study researches an ASLC featured large scale production system with mass customization influences (Lu and Storch, 2005a, 2005b).

This case study model is framed on a mass customized ASLC system and applies discrete event interactions between the times of customizations selected by customers and the impacts to the final integrations among more than one hundred processes over durations of longer than three years.

In this case, the ASLC featured mass customized production system that has three possible decision gate zones to allow customers to make configuration decisions for their customized products. Each decision gate zone presents a limited time period in the production system. The custom decision triggering mechanism at one of the three decision gate zones enables the selection of which fixed optional component to order, and which configurable optional component to order. Production time of the same component of both categories changes depending on which gate the decision was made. Naturally, the earlier the decision is made, the easier it is on the production efficiency. This model framework reflects the final product production time penalties when a customer makes a series of customized product decisions at the last possible decision gate zone.

Figure 22 simplified a real ASLC featured large scale production system into a conceptual model frame of the system structure. Gate 1 happens right around the time when the work order is released, gate 2 takes place in the middle of a major component production, and gate 3 is right before the final integration. At any given time all suppliers perform at different locations on their learning curve, as shown in the diamond shape dots. Hence, customer decisions at different gates produce un-equal system impacts all the way to the final integration, per component and per supplier.

All elements in the system are modeled using the DES method from previous chapters. Elements in the system are connected according to both the information flow and component flow networks, in which all are operated on one master production schedule. Each component unit has its own production starting time (s_{ix}), which is reflected as in the

gate 1 zone. Each component process has its unique ASLC of different progress rates. The gate 2 zone resides within each component unit processes times (y_{ix}). In an ASLC featured system, component unit process progresses vary from component to component. The diamond-shaped dot within each supplier's process box in figure 22 represents process status on its learning curve for that supplier. Suppliers are not likely to have their processes status on the relatively same learning curve locations in a system that is both asynchronous and stochastic. The gate 3 zone is before the starting of last group of process times (y_{ix}) in the final integration and at the same time some of the component unit processes are still taking places. Thus, some of the component finish times (f_{ix}) are overlapped with the decision triggering time in the gate 3 zone. It is intuitively reasonable that component customization decisions are the least desirable to trigger during the gate 3 zone.

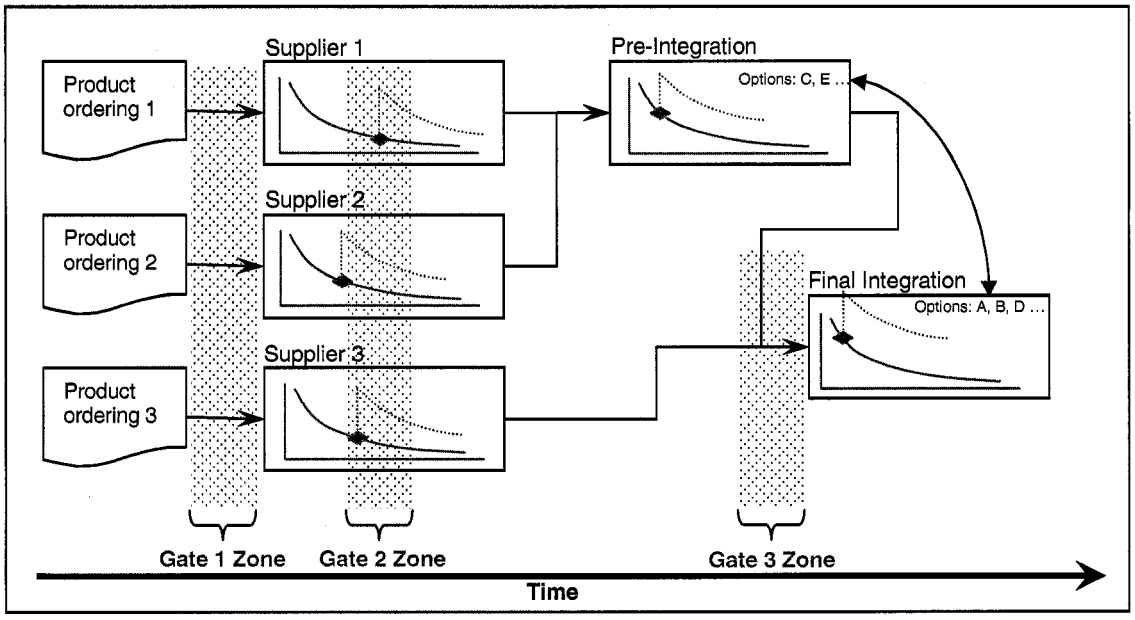


Figure 22 A simulation model of a production system example with decision gates

The master production schedule initiates the start of a process for all suppliers. This master production schedule is not randomly distributed; it is very deterministically planned. In the static model, production plans for all suppliers and partners are determined by the final product delivery dates and all related lead times. Only a few suppliers are

responsible for making customized products that have shorter lead time than the rest of the suppliers. All normal demands are planned ahead of time between the customer and the system integrator. However, none of the unexpected changes are in the master schedule and they will have to be implemented without sacrificing the final product delivery time by other means such as increasing resources.

To model the system with more fidelity, process triggering objects in the system may come from several different seeds -- randomly generated unexpected events to different entities in the system. Given that supplier 1 has one random stream with one unexpected change distribution, supplier 2 has a different random stream combined with another failure distribution on all of its processes, and the same rule goes throughout the system.

Additionally, statistical distribution of performances of each supplier varies in the real environment. Their respective normal process performance and unexpected event distribution varies just as they are performing dynamically differently on their individual learning curves. For changes associated with the "fixed" options – all subsequent products from all related suppliers change. For changes associated with the customized options – only affected suppliers need to change. Not all suppliers are on the same serial number, learning curve rate, or have the same statistical process distribution at the same time; therefore it is not feasible for all related suppliers to change their process at the same time on the same serial number. Consequently, picking a good time to trigger any customization decisions with some level of predictable consequences becomes an art that can be benefited by applying the stated ASLC model with the DES method.

Three categories of system components were mentioned earlier in section 3.3 on page 31. Figure 23 shows processes in a simulation model where both categories 1 and 2 customized selections take place earlier in the final integration. Their respective customized works are performed during the early stage of the final integration. Category 3 related selections happen too late in the system and can only be performed near the end of the final integration. Figure 23 is a more detailed view of processes among final integration and final testing and delivery as depicted earlier in figure 1 on page 28.

Customization decisions triggered earlier among all three gate zones have different levels of influences among category 1, 2, and/or 3 processes during the final integration.

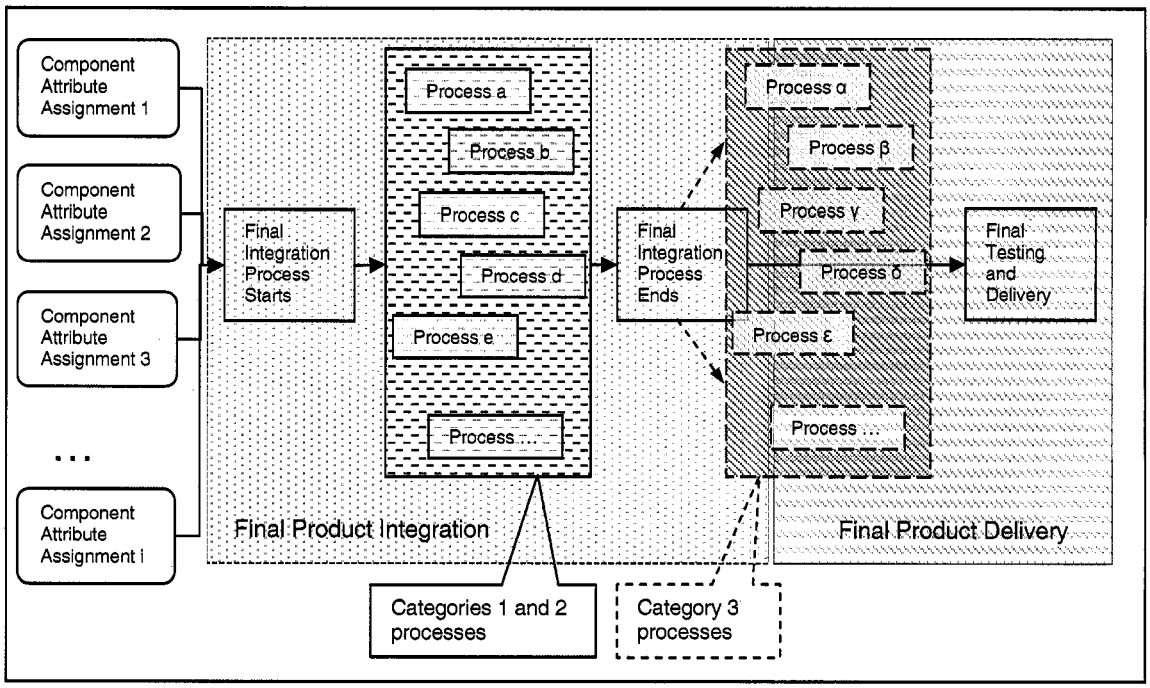


Figure 23 A case model of processes in final product integration and delivery

Over the left side in figure 23, a series of component attributes are assigned in Assign modules in the DES model. In the middle of figure 23, a large dot-shaded box represents the Final Product Integration group of processes. Over the right side in the figure, another large shaded box represents the Final Product Delivery group of processes. Category 1 and 2 processes are all within the Final Product Integration process group. Category 3 processes, however, are spread across both the Final Product Integration and the Final Product Delivery groups of processes. Process times and occurrences in figure 23 depend on customization decisions made among those three gate zones. Some of the category 3 processes will take place when certain component units are in the final product integration and most of the category 3 related processes are done in the final product delivery group of processes. DES codes of this case study are listed in appendix J.

In case study two, three different scenarios are simulated with different decision percentages at each decision opportunity. Table 24 lists differences among these scenarios, the respective location of these gate zones are demonstrated in figure 22. Gate 1 is right after the product ordering event. Gate 2 is in the middle of a component production process. Gate 3 is at the end of all component processes and just before the final integration. There are three levels of customizing decision making scenarios in table 24.

Scenario A: 25 percent of the customization selections are made at the gate 1, 50 percent at the gate 2, and 25 percent at the gate 3. Scenario B and C have different percentage selections at different gates as seen in table 24. In the simulation model, these decision percentages are represented by three decision variables with discrete distribution targeted at their scenario percentages.

Table 24 Case study 2 process scenarios

Scenarios	A	B	C
Gate 1	25%	15%	100%
Gate 2	50%	15%	0%
Gate 3	25%	70%	0%

Customization related process delays (e.g.: attDelay1 + attDelay2 + ...) are represented as attributes, because they need to be associated with component unit serial numbers. It is necessary to use all attributes of each component throughout the whole system. Since the value of an assigned variable will not change across multiple processes unless there is an action to change it, the value of an attribute may change from a process to the next process due to its ASLC and customization characteristics.

Disruptions happened at three different gates to reflect major product customization events and/or major design changes. Design changes can be a derivative product and/or a minor product model release in the middle of processes when the other models are being built. Process times in these two final stages contain all of the customization work statements that are selected among any of the three gate zones. Simulation modeling results, based on three customization scenarios in table 24, of the processing time during a given final integration stage of the whole system are shown in figures 24 through 28. Process time in figures 24 through 28 is the sum of process times of all customization and regular processes in the final product integration and the final product delivery.

The general curvature of the process time is due to the nature of the ASLCs. Three spikes of process time increases at around 850, 1040 and 1130 calendar date units (in Figure 24) are caused by three major customization events. The height of those spikes (an attribute) reduces gradually from the first occurrence to the last occurrence. If a curve is drawn by connecting three highest points of those three spikes, the curve presents another learning curve that reflects customization decision influences in the system.

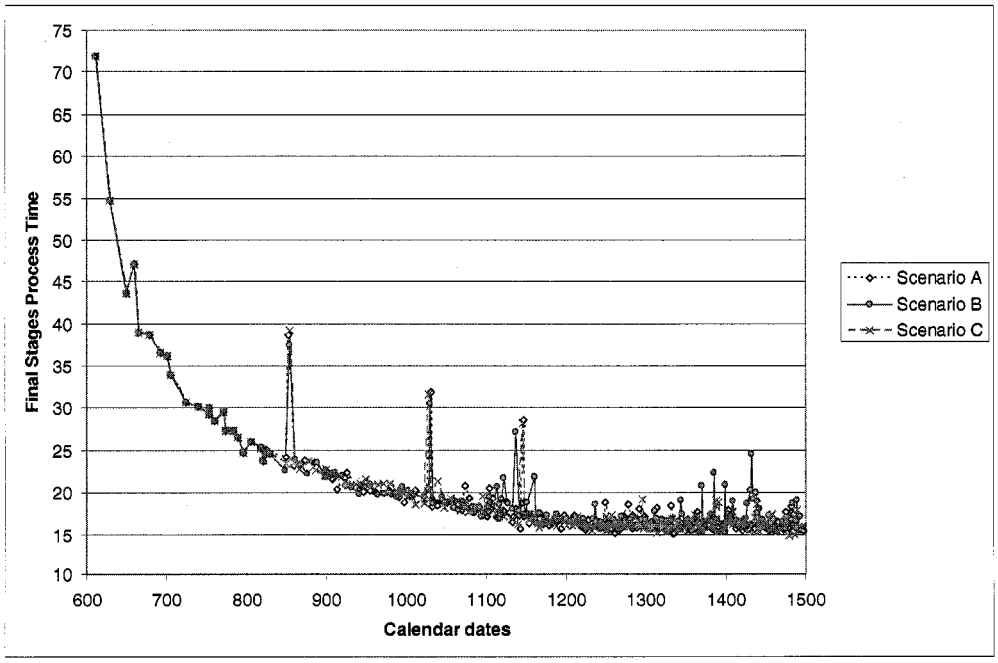


Figure 24 Case study 2: Process times of the final integration

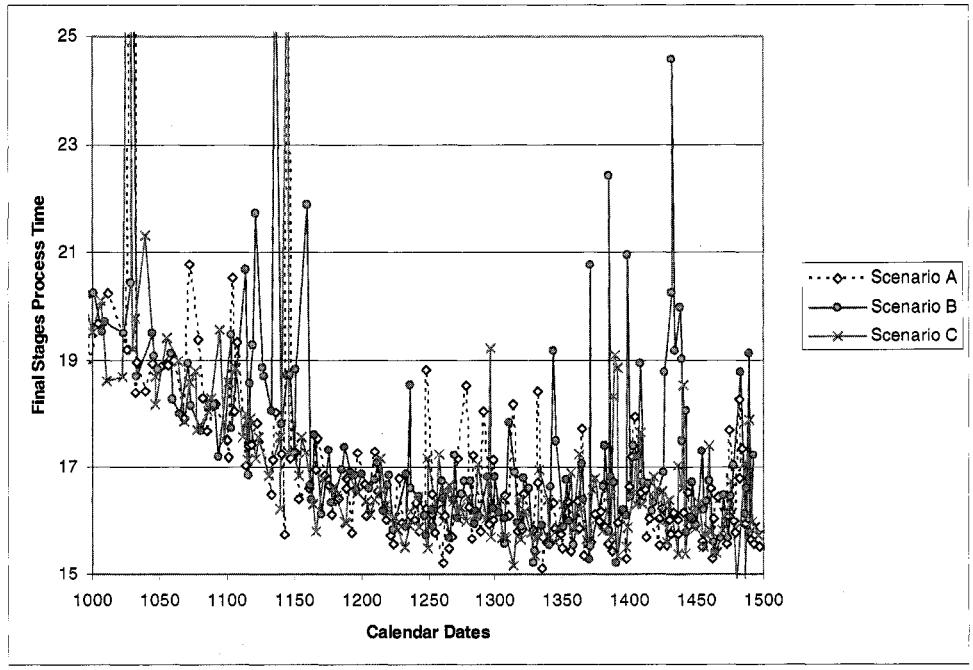


Figure 25 Case study 2: Detailed view of process times

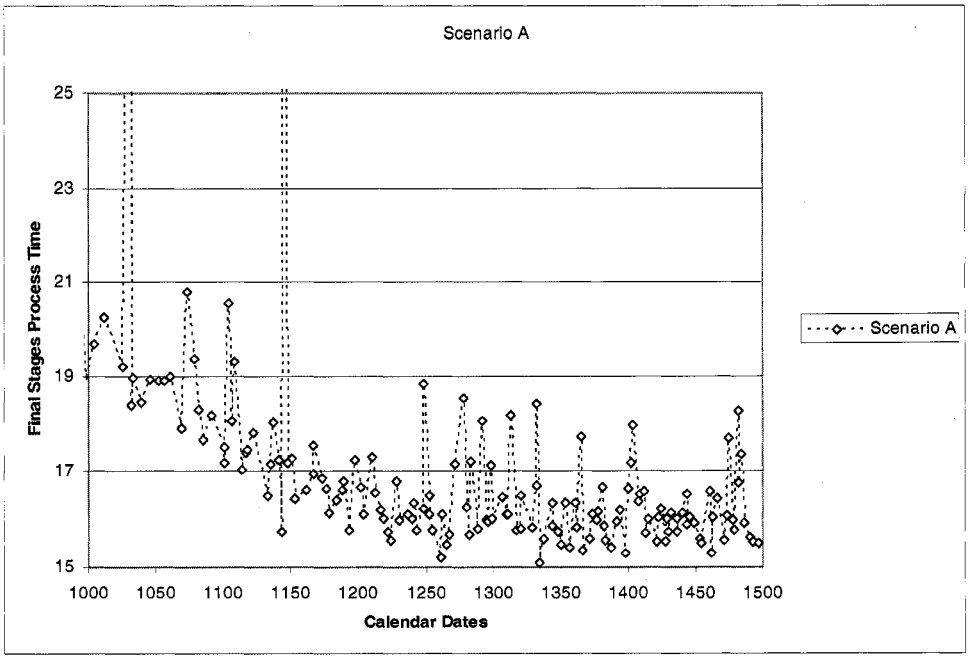


Figure 26 Case study 2: Detailed view of scenario A process times

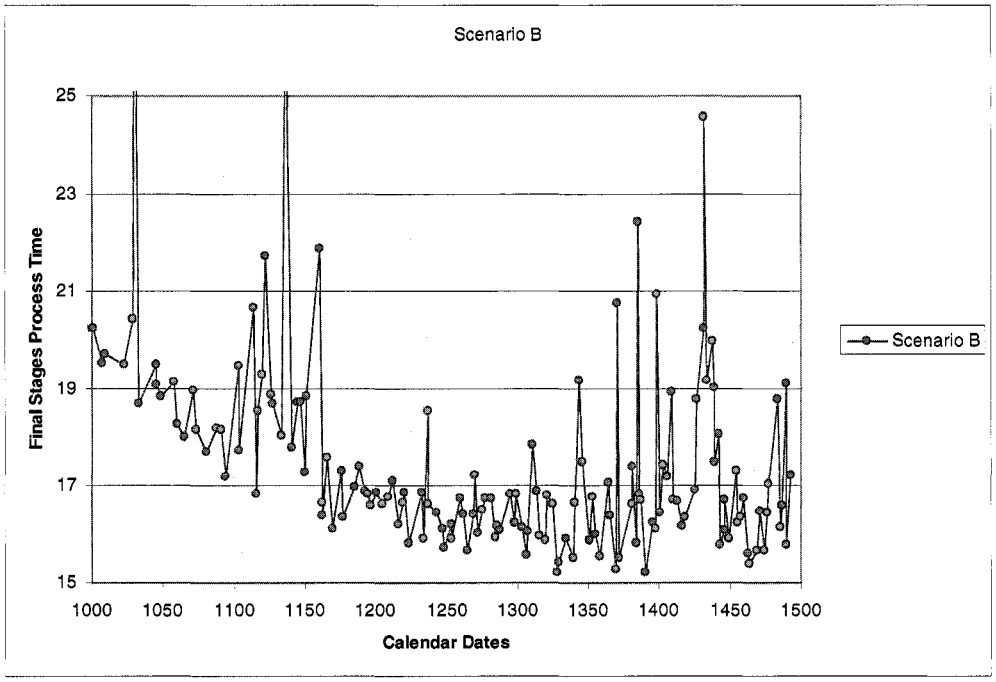


Figure 27 Case study 2: Detailed view of scenario B process times

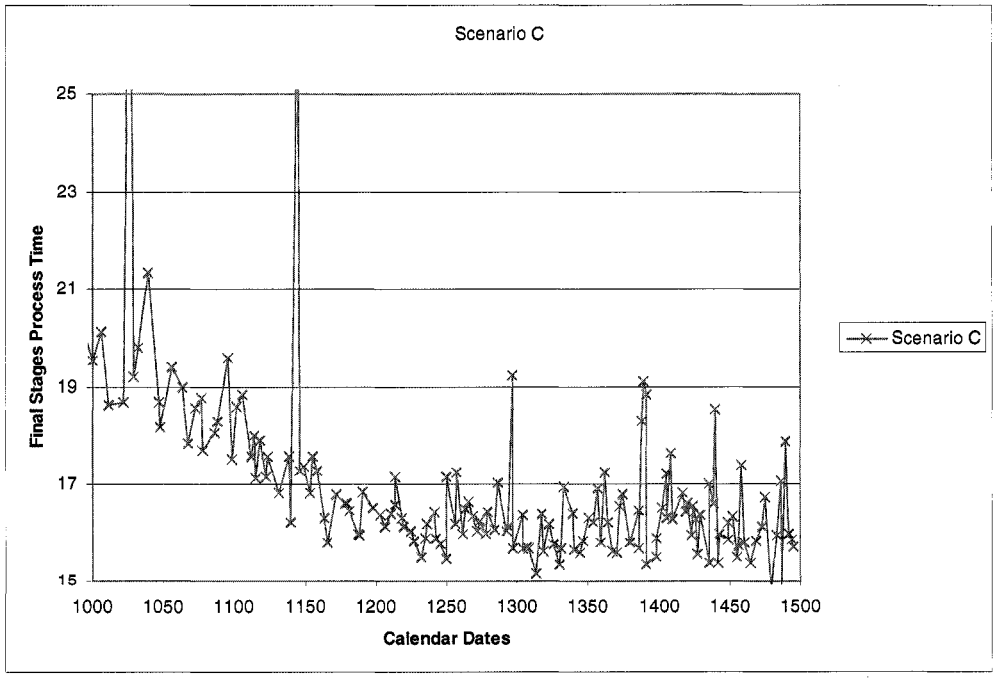


Figure 28 Case study 2: Detailed view of scenario C process times

Franke and Piller (2003) published the first in-depth empirical analysis of design process and outcome of "toolkits for user innovation and design" in the watch market. The study showed that what customers are willing to pay for their self-designed products exceeds that for the best selling standard watches of the same technical quality up to 200%. They addressed the process of designing a product by the actions of users and user perceptions. They stated that a question worth further pursuing is why users are willing to pay such an enormous price premium for their self-designed product. In figures 27 and 28, based on scenarios defined in table 24, scenario C has the least disruptions to the final integration processes while scenario B has the most. Discrete simulation modeling findings in this case study have also shown 150% to 200% process time increases are possible for demanding product customizations. The 150% to 200% process time increases from scenario C to scenario B happen around 1400 and 1450 calendar dates. Scenario A result lies between the other two scenarios as expected.

The second case study applies the DES method from sections 6.2 and 7.2 on an ASLC featured mass customized large scale production system. The ASLC effect is in the system as previous case study and hypotheses. Additionally, the mass customization decision-making part of the system characteristic in this case makes it different from the first case study and all of the hypotheses in chapters 5 and 7. The analytic statistical method is not applied in this case study, due to the complexity of the system combined with the efficiency of utilizing the DES methodology in an ASLC featured large scale production system.

Chapter 3 discusses large scale production system, chapter 4 introduces the ASLC, chapter 5 outlines mathematical presentations in modeling ASLC featured large scale production systems with several alternative hypotheses, chapter 6 depicts applications of the two ASLC methods, and chapter 7 shows how ASLC methods are applied, verified/validated, plus hypotheses analyses. Thanks to the validation and verification of the DES methodology, these two case studies in chapter 8 apply the DES method successfully to approach ASLC featured large scale production systems of different attributes. The next chapter concludes this research.

Chapter 9: Conclusions

9.1 The ASLC Model

Before the industrial revolution more than a century ago, some of the traditional craftsmanship could take a lifelong journey to master. Blacksmith, carpenter, and tailor are a few examples of the traditional craftsmanship professions. People took a long time following a master in their profession before they could produce a product independently. It is reasonable to speculate that it takes an apprentice a much longer time to produce the first article then subsequent ones. As an apprentice earns his/her way to become a master, the time it takes to produce an article of similar characteristics will be drastically reduced to a more stable and predicable time period. The learning curve effect played an important role in craftsman style production systems among our communities more than a century ago. The industrial revolution, followed by mass production in the US automobile industry (Womack et al. 1990) in the early twentieth century, transferred most craftsmanship type of industries to mass producing the same set of items as fast as possible on the same production line. In most mass production operations, performance is measured after all of the learning curve effects are over (see Section 4.3, page 43 - 45, figure 7 and 8). It is important to have synchronized processes in a high speed mass production system where identical products are made. One of the subsequently important production system restructures was in the aerospace industry during World War II when war planes needed to be produced with as short of a learning curve as possible (Wright 1936). This was because all planes produced during early stages of the learning curve (see Section 4.3) must join the service as soon as possible. The stochastic nature of process times in some production systems was not that noticeable more than half a century ago. Since all major operations were conducted on the same site during the inception of the learning curve, the asynchronous nature among different learning curves from different major production partners may not be noticeable either.

Nevertheless, recent trends of production systems of long lead-time long life-cycle complicated products are reaching toward more partners and suppliers globally. These complicated products are, but not limited to, commercial aircraft, ocean vessels, and high-rise apartment complexes. Suppliers and partners involved in production systems for

these complicated products are geographically located across continents worldwide. Components of these complicated products are generally expensive with very few made for testing and/or trial articles. Therefore, early production runs are important to the success of new products in such a production system. Since partners and suppliers are separate business entities in different countries, they have their unique production environments that contribute to their unique learning curve characteristics during early production runs. Global partners are having different lead times and process times depending on characteristics of their respective components and process capabilities. Major components are made by different partners from different regions. The production system needs all components to arrive at the final integration location as synchronously as possible. Because process lead times are asynchronous among all partners, some of the partners will have to start their process earlier and/or later than others. Moreover, processes of all partners are highly likely to have unequal levels of stochastic process time variations.

In today's large scale production system, process starting times are asynchronous and durations are stochastic among global partners during early stages of product launches. Process time of each component produced by a given partner is riding on a unique learning curve with a set of unique learning curve rates (see Section 4.1 on page 41, equation (11) and (12) on page 50). Therefore, learning curves among partners and suppliers in large scale production systems globally are asynchronous and stochastic in nature. This dissertation unveils the asynchronous stochastic learning curve (ASLC) effect in a large scale production system.

The major contribution of this dissertation is developing and testing the theory of ASLC. It also shows some potential applications of this new theory and identifies additional potential uses.

The value of the ASLC model is that it discloses otherwise unknown effects in a large scale production system. The ASLC model addresses the asynchronous nature of process lead-times because of differences among learning curves and the stochastic process durations in stages that are before production systems have reached their steady

states. The disclosed ASLC model and methods advance industrial engineering researchers and practitioners at large to research and/or to practice in these following areas:

- The ASLC theory and its effects in a large scale production system
- Application of the ASLC theory and methods in a large scale production system
- Production system control during early stages of complicated product launches
- Mass customization in an ASLC influenced production system
- Logistics in transporting large components with ASLC in the system
- Configuration and component ordering decision timings during early stages in an ASLC featured production system
- Postponement of engineering-to-order decision points when ASLC is in a system
- Change orders and unexpected event management in a system with ASLC
- Modeling of production fluctuations and determining production rate change opportunities in an ASLC featured global supply chain system

Another important contribution from the novel ASLC theory and model in this dissertation is elevating the awareness of the ASLC featured production systems. Researchers and practitioners can now address production system related phenomena with the consideration of the ASLC theory, and also model production systems using the developed ASLC model and methods. Knowing the ASLC exists in a large scale production system, the planning and execution of the production system during its early stages can be much improved over the traditional deterministic Gantt chart scheduling or similar methods. For example, decisions on mass customized configurations can now be postponed by using the verified and validated ASLC model and methods; events of various learning curves and characteristics can now be included in modeling global supply chain logistics. This dissertation promotes the understanding of the ASLC in a large scale production system with the ASLC framework, and provides a mathematic model, an analytic statistical method, a discrete event simulation method, and verification and validation of both methods.

The opportunity in researching the ASLC in a large scale production system is discovered through literature reviews in chapter two. Effects and stages of ASLC in a large scale production system are discussed in chapter three. Chapter four outlines elements of an ASLC featured system. Chapter five depicts the ASLC mathematical model in detail. Therefore, this dissertation concludes that the disclosed ASLC framework and model in a large scale production system is innovative and of value (Lu, 2007; Lu, Petersen, and Storch, 2007).

9.2 The ASLC Methodologies

Many production systems since the industrial revolution have been migrating first from Europe to North America, and today to Asia, partially thanks to modern transporters and efficiency in the global supply chain system. Today's large scale production systems are often across several continents worldwide. As mentioned earlier, global partners likely have different characteristics of ASLC effects in a production system that partners share development risks and production responsibilities. The risk of developing a brand new long life-cycle product, such as an aircraft, is often difficult to justify. If global partners, who also will be responsible for producing their designed components, can share the risk of the product development then the business case can be easier to justify. Furthermore, since customers of these types of large scale products are mostly pre-identified, all components produced are eventually going to the final integration site. Global partners hardly have the same rate of learning in their production operations. Partners have un-equal dynamic changing lead-times based on different learning curve rates.

For products that can be transported by standard cargo containers and/or air freighters, there is hardly any geographic boundary from the product concept design and production, all the way to the final product consuming market. For products that cannot be transported by any existing transportation means, production locations of major product components become an important factor. Inter-modal transportation logistics has to be specially arranged for larger and heavier than standard cargo container components – such as a section of an aircraft. Specialized inter-modal transportation arrangements exist only in a few regions globally. If a new product needs such special arrangements, its production sites will be limited – unless there is an alternative solution. The stated ASLC model and

method is capable of addressing logistics in a global production system of customized products with different decision gates (Lu and Storch 2006).

This dissertation contributes two methodologies in addressing the ASLC model. The analytic statistical method allows the use of the ASLC model through a series of manual calculations. The discrete event simulation (DES) method enables the use of the ASLC model through a computer simulation approach.

The analytic statistical method provides data processing steps that have visible data values in all steps. The DES method provides a more efficient way to model ASLC systems in computer, but some of the system data cannot be visible at the time of the data calculation. The DES method does have a data collection mechanism that outputs collected data electronically afterwards. Therefore, the analytic statistical method is useful in verifying and validating the DES method. Details of both methods have been discussed in chapter 6.

Both statistical and DES ASLC methodologies in this dissertation are able to be scaled to accommodate different levels of complexities in ASLC featured systems. For smaller ASLC featured systems, the statistical method can be applied to generate results faster. For larger and more complicated ASLC featured systems, the DES method can be more effective. The DES method requires at least one more software than the statistical method. Since the DES method needs a discrete event simulation engine to process stochastic process times only when an entity is passing through its associated processes.

Since the DES method is more efficient in handling more complicated and bigger production systems with ASLCs, it is important to ensure the DES method is verified and can generate validate results. Thus, this research provides an analytical methodology to address the verification and validation of the DES method. The verified and validated DES method can then be applied to analyze different scenarios in a large scale production system. The innovative, verified and validated ASLC DES method thus can be effectively applied to model:

- The complexity of logistics in a global enterprise,
- Stochastic learning curve featured early stages of new product launches with asynchronous process times among partners,
- Customized modules with various decision points from different customers, and
- Many other ASLC featured production system factors.

Therefore, this dissertation concludes that the stated ASLC featured large scale production system can be analyzed by either the analytic statistical or the DES methodologies.

9.3 The ASLC Applications

The ASLC model can be generally applied for systems with same characteristics besides large scale production system in the aerospace industry. Two application methods are discussed in this dissertation; practitioners can apply either the analytic statistical method or the discrete event simulation method. Since the discrete event simulation application of the ASLC is verified and validated, it can be applied to analyze hypotheses of various systems. The example in applying ASLC by using both methods in chapter 7 has three Engineering-To-Order (ETO) levels. The stated ASLC model and methods can be applied for systems with many times more ETO levels, such as in the area of mass customization.

Mass customization (MC) in a large scale production system can be one of the ASLC applications. Mass customization produces products that are tailored to the customers' special needs. MC products may be of high or low volume by the total final product count. However, managing the complexity within the MC production system effectively is more important. The general trend of recent human civilization has been having a subtle common phenomenon: people like matters customized to their needs. A century or two ago, most textile products were designed manually with specific people in mind prior to the execution of the production. A century or two later, most people sought mass produced textile products that fit them the best, while only a few could afford the privilege to enjoy custom made products that fit them perfectly. However, the shift from the mass production to the mass customized era is approaching our society in very noticeable fashion. The

production system that supported and benefited from mass production products is also changing to adapt to the mass customization methodology.

The null hypothesis (section 5.3) stated that same set of units of all components from different partners will arrive at the final integration location simultaneously. The synchronization of component unit arrivals in the production system would be perfectly ideal if all partners can perform per the null hypothesis. In many mass production systems when components are produced in the same location, the synchronization of component arrivals to their final integration location can be achieved. Mass produced soft drinks and beers are two examples that likely not to reject the null hypothesis. The large scale production system in this research involves more complicated processes of different learning curve characteristics and with global partners in different regions. Therefore, the first alternative hypothesis stated that same set of units of all components from different partners will NOT arrive at the final integration location simultaneously. Examples studied by using non-dimensionalized real data (Section 7.3) showed that the first alternative hypothesis rejects the null hypothesis in an ASLC featured large scale production system.

The second alternative hypothesis uses the same learning curve rate for all component partners while individual component process times are different among component partners. This alternative hypothesis is to see if the condition of the original null hypothesis can ever be achieved. Since component processes have different process times, at the early stages of the production, their component unit arrival times are not synchronized. However, since they improve over the same rate, they eventually have a much better synchronized component unit arrival performance as compared with the null hypothesis scenario (Figure 15, section 7.4).

The third alternative hypothesis sets all units of the same component with the same process time. Different components have their unique process times. There is no learning curve effect in any component partner's production process. There is no variation on unit process times per component. This hypothesis assumes the production system is performing without the asynchronous, stochastic, and learning curve effects. Therefore, the traditional Gantt chart practice shall be applicable in this hypothesis to schedule all unit

processes per their respective component level process time differences. This hypothesis can test the ASLC model for its ability to handle a deterministic scenario. The expected result is that all component units arrive at the final integration simultaneously every time all the time. The hypothesis testing result (Figure 16 in section 7.4) showed exactly as expected – all component units arrived at the final integration synchronously.

The forth alternative hypothesis tests the system with almost 100% learning curve rates for all components while component unit process times remain stochastic. This hypothesis examines the model for its ability to show some small fluctuations among component unit delivery times due to their stochastic process times, but with very fast learning. The fast learning shall be noticeable in the reduction of differences among component unit arrival times at the final integration. In most real world business cases, this alternative hypothesis can be practically realistic. The ASLC model successfully illustrates results of this alternative hypothesis with expected trends (Figure 17, section 7.4). There are some variations among component unit delivery times and the learning curve progression is very mild. As compared with the situation in the null hypothesis, the overall system performance in this alternative hypothesis is much better (Figure 17).

The null and the first alternative hypotheses are analyzed by applying the ASLC model and both statistical and DES methods. The second and fourth hypotheses illustrate different characteristics of ASLC featured production system. The third hypothesis presents a steady-state production system, which can be a special representation of an ASLC. The ASLC model and methods effectively applied to all four alternative hypotheses. Therefore, this dissertation concludes that the stated ASLC model and methods can be applied to ASLC featured large scale production system of different characteristics.

9.4 The ASLC Case Study in Large Scale Production System

Chapter 8 includes two case studies that have ASLCs in the system. One of the cases has mass customization decision timing influences in an ASLC featured large scale production system. The ASLC model in this dissertation can be further applied to the logistics part of a production system.

Figure 1 (page 28) in chapter 3 outlines a large scale production system framework, in which the ASLC model and methods are applied for various hypotheses. The application of the ASLC model and methods may only reside in one of the major events depicted in figure 1. Case study results show that the proposed ASLC model works in an ETO system via both statistical and DES methods. Moreover, in an integrated large scale production system, this result may represent only a segment of the total system. Figure 29 depicts a large-scale customized production system where, say, four major events are in the system:

- End-item logistics
- Large-component assembly
- Final product integration
- Final product Delivery

Very similar to figure 1, figure 29 shows major events take place sequentially throughout time, which progresses from left to right in the figure. Production system planning and scheduling goes from right to left, indicating the nature of the firm product delivery deadlines in the system. The manufacturing and supply chain logistics goes from left to right with the objective of meeting production schedule events while customization events may happen at all times throughout the production timeline. The Final Product Delivery box at the far right is a "push" type event where final products are being pushed to completion in order to meet the committed delivery dates. These delivery dates drive the master schedule in the Final Product Integration stage, which is also a "push" type process. The Final Product Integration can possibly be performed at multiple sites; although it is more practical and economical to have Final Product Integration processes conducted on the same site. Consequently, the major event beforehand, Large Component Assembly has to meet the initial conditions that support the final product integration. The Component Schedule that controls the Large Component Assembly is a mix of both push and pull processes. End Item Logistics can be a totally pull event based on their successors' processes in the Large Component Assembly event.

ASLC model and methods of this dissertation can be applied to any one of the four major events, especially in the large component assembly event. Since all major components need to be finished as close to the same demand time to support the final product integration but their starting times may vary largely due to the nature of ASLC in the system. The ASLC model and method can be applied to analyze schedule disruptions for the framework outlined in figure 29 in a large scale production system and in other systems of similar characteristics.

As more businesses and consumers demand individually tailored products that require ETOs in customized production systems, it is inevitable that the ASLC effects need to be understood and managed. This dissertation introduces the ASLC model, which contributes to the literature by its novel system view. Two case studies further illustrate the benefit of the ASLC model and methods. One of the significances of this ASLC model in addressing ETO related mass-customization practices is the ability to foresee potential product delivery performance (N) of component arrival times (f) at the final integration location. The product delivery performance can be treated as one of the system performance indicators.

This methodology of applying the ASLC model in an ETO or a mass customized large scale production system provides relative importance among process learning curve characteristics, process durations and starting times, and product finish rates among different levels of customization. While it may be a fine art to balance all of the mentioned factors in a large scale production system, this dissertation provides a validated and verified scientific approach to analyze the system. Nevertheless, practitioners shall always take caution in applying any analytical or scientific technique. This dissertation contributes to the community with means to apply the stated novel ASLC theory and model via case studies. These case studies further validate and support the stated ASLC theory and model.

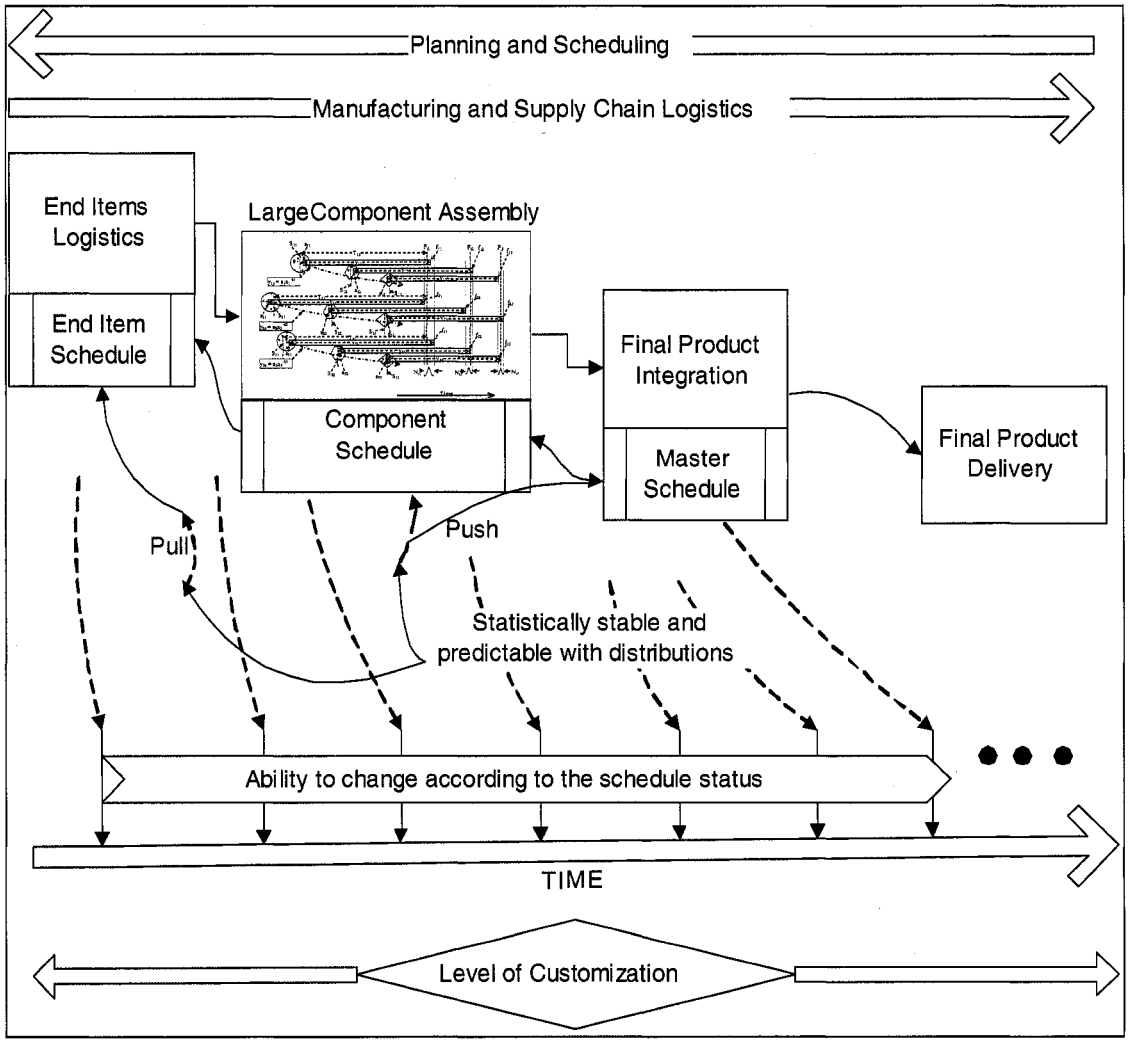


Figure 29 ASLC in a large scale production system

The next section addresses limitations and future research opportunities.

9.5 Future Research Opportunities

Limitations of this work represent ample future research opportunities. Levels of resources in each respective process are assumed as not a constraint in this study. It will be a worthwhile endeavor to further abilities in either schedule or resource prioritized system modeling. This research is schedule prioritized. If it is a resource prioritized study, then results would have levels of resources needed for any given customization and / or learning curve level per process. Moreover, it might be a worthwhile task to extend this

current research to other stages as stated in the system boundaries in the introduction, so that ASLC processes from customized product development to final product delivery may be better modeled. Cost per process may be an attribute that could have been added with correlation to the level of process customization per process. The cost attribute may enable higher level of system strategic decisions on optimizing costs of customization across processes with ASLC effects. This research varies processes times per component unit. In order to apply the ASLC model in the Final Product Integration event for mass customized products, it may be necessary to have an ability to model customized process times on items of different sequence numbers within each component. For example, $y_{1,1ETO}$ may have an ETO level of 8% while $y_{1,2ETO}$ has an ETO level of 15%. Therefore, the customization processes with added value can be modeled and deployed as late in the production system as possible.

This work also researches methods that enable customers to delay decision making at various defined time points with minimum production impact results. The ability to delay decision making is important especially for products that require large manufacturing system integration. Most products from such a large system have long lead times, in the scale of years. Local market situations, currency exchange rates, and the global economic climate are just a few factors that can influence final details of customized features from the customers' point-of-view. The later the implementation of customized features, the easier the configuration and integration of the product it is from the designers' point of view. Customized component suppliers naturally prefer as early of a notice as possible upon a commitment between the customer and the product integrator. Consequently, there exists a need to intelligently define possible key decision points at the product design stages for the designers and executed at the product integration stages for the customers and suppliers.

Fundamental objects considered in the area of decision gate zone definition research include flows of major components from individual suppliers to various integration sites, identification of customized configurations, possible integration processes and locations for customized configurations, components that require more design iterations for customizations, and customer preferred time frame of customized decision points. These

objects are to be organized in a series of discrete event simulation models of different ASLC at the component unit level, which is not covered in this dissertation.

ASLC and its respective logistics in a production system with variable critical paths is another future research opportunity. The variable critical path in a production system may be contributed by resource variations at the component unit process level, inter-dependencies among partners, partially finished inventories, component unit level customizations, variable demand rates at the component level, final product revision frequencies, concurrent engineering in design phases, and many other factors. This dissertation disclosed an applicable ASLC model and method in a large scale production system as a stepping stone to explore many future research opportunities.

Current state of the art addresses mass customized large scale global production system from many different angles, but none of the existing research encompasses issues in the whole system. For example, there are outstanding papers, such as in supply chain management (Beamon 1998, 1999, and 2001) and economic lot scheduling (Faaland and Schmitt 2002), that provided solid foundations for their respective research. This research provides opportunities to further advance in seeking relationships among more factors associated with the ASLC featured large scale production systems by applying discrete event simulation and a statistical methods.

There are other additional considerations in a large scale production system integration architecture and performance evaluation. Considerations may contain factors in:

1. Component ordering system

This consideration evaluates component ordering performance. The event of component ordering triggers the starting time of component unit production. Since most components in the system have different lead-times, the stochastic component ordering lead-times can be further analyzed to evaluate the overall system performance.

2. Inventory

This consideration evaluates inventory management performance. It has been well known that too much inventory is not a favorable condition in any system. However, this system does not have alternative production sources for most components. Additionally, there are logistics constraints in transporting many components. Thus, dynamic inventory management with respect to ASLCs of component processes may be further studied to evaluate the overall system performance.

3. Mass Customization

This consideration encompasses customization attributes across the overall product family. Since many major components have different levels of customization and their levels of customization impact many other system performance measures. As seen in the case studies, more research may be helpful to analyze different customization levels among components and within groups of component units.

4. Rate of change of final product

The final product configuration actually will not stay the same over a long period of time. New product improvements and derivatives are taking place while existing configurations are being produced in the system. Timing of new product introduction can be one of the important factors to consider in an ASLC featured large scale production system. Thus, more research in optimizing the rate of change of final products can be used to further evaluate the overall system performance.

5. Interim product service rate/due-date performance

The interim product is composed of most major structural components at the final integration site where the highest value added work is being executed. This stage in a production system traditionally receives the highest visibility. If the interim product is moving too fast through the final assembly line, there is a possibility of a disconnected supply chain. If it's moving too slow, then the inventory and work-in-process levels will likely increase. Thus, this consideration can use more studies.

6. Logistics

Logistics performance is critical in large scale production system integration. Especially since component production facilities are scattered globally. Surface inter modal transportation is needed in addition to dedicated and constrained air freighter shipping. Thus, the logistics related research in an ASLC system is worth further consideration.

7. Non-recurring design and investment

There are components of mass-customized products that do not need to be re-designed frequently. There are components that occasionally need to be re-designed along with some level of resource and capital investments to implement the design. This type of non-recurring design and investment could be referred to as new model introductions. Issues associated with these activities may be further considered to improve production systems with ASLCs.

8. Component manufacturing performance

Since components in a global production system are produced from partners worldwide, it is important for those involved partners to be able to manufacture their respective components efficiently and successfully. Thus, the consideration in capturing meaningful production performance measures in each individual production facility within the ASLC featured production system framework may worth further research.

How all of the above further research opportunities are combined and correlated and their relationship to the overall system performance have yet be studied and represented. Partners at various positions throughout the supply chain may have different individual priorities as compared to the final integrator. Utilizing the depicted ASLC model and methods as the starting point, further studies may help partners at different positions in the supply chain to evaluate their different priorities against the overall system performance measures.

There may be research opportunities in establishing algorithm to balance these eight considerations from partners' view and from the system integrator's view. For example, in order to simplify the logistics in the system, it might be intuitive to simplify levels of the customization with the longest possible lead-times. However, the customer may wish to finalize customization related orders as late as possible. Not all considerations in an ASLC featured large scale production system are orthogonal or parallel.

This dissertation discloses an ASLC model and methods in exercising the model. The ASLC model and methods can be a foundation that help researchers and practitioners to further researches in similar systems with regarding to component ordering, inventory, mass customization, final product changing rates, logistics, non-recurring design / investment, and component manufacturing performance.

This dissertation serves as a starting point in understanding asynchronous stochastic learning curve in a large production system. This dissertation concludes that it successfully contributes to the literature in these following novel items:

- The introduction of the ASLC concept
- The ASLC framework
- Mathematical presentation of the ASLC model
- A statistical method of the ASLC model
- A discrete event simulation method of the ASLC model
- A verification method between the statistical and the simulation methods
- Applications of the ASLC model in a large scale production system

BIBLIOGRAPHY

Agrawal, M., Kumaresh, T.V., and Mercer, G.A., 2001, "The False Promise of Mass Customization," The McKinsey Quarterly, No. 3, pp. 62 – 71.

Alfieri, A. and Brandimarte, P, 1997, "Object-Oriented Modeling and Simulation of Integrated Production/Distribution Systems," Computer Integrated Manufacturing Systems, Vol. 10, No. 4, pp. 261 – 266.

Anderson, E.G. Jr. and Parker, G.G., 2002, "The Effect of Learning on the Make/Buy Decision," Production and Operations Management, Vol. 11, No. 3, pp. 313 – 339.

Argote, L., and Epple, D., 1990, "Learning Curves in Manufacturing," Science Magazine, Vol. 247, pp. 920 – 924.

Bandivadekar, A.P., Kumar, V., Gunter, K.L., and Sutherland, J.W., 2004, "A Model for Material Flows and Economic Exchanges within the U.S. Automotive Life Cycle Chain," Journal of Manufacturing Systems, Vol. 23, No. 1, pp. 22 – 29.

Beamon, B.M., 1998, "Supply Chain Design and Analysis: Models and Methods," International Journal of Production Economics, Vol. 55, pp. 281 – 294.

Beamon, B.M., 1999, "Measuring Supply Chain Performance," International Journal of Operations and Production Management, Vol. 19, No. 3, pp. 275 – 292.

Beamon, B.M., 2004, "Simulation," Course Notes – University of Washington, pp. PSA 25 – 29.

Beamon, B.M. and Chen, V.C.P., 2001, "Performance Analysis of Conjoined Supply Chains," International Journal of Production Research, Vol. 39, No. 14, pp. 3195 – 3218.

Beamon, B.M. and Ware, T.M., 1998, "A Process Quality Model for the Analysis, Improvement, and Control of Supply Chain Systems," Logistics Information Management, Vol. 11, No. 2, pp. 105 – 113.

Berman, B., 2002, "Should your firm adopt a mass customization strategy?" Business Horizons, Vol. 45, No. 4, pp. 51 – 60.

Bigand, M., Korbaa, O., and Bourey, J.P., 2004, "Integration of FMS Performance Evaluation Models Using Patterns for an Information System Design," Computers & Industrial Engineering, Vol. 46, pp. 625 – 637.

Biskup, D., and Simons, D., 2004, "Common Due Date Scheduling With Autonomous and Induced Learning," European Journal of Operational Research, Vol. 159, No. 3, pp. 606 – 616.

Blackhurst, J., Wu, T., and O'Grady, P., 2004, "Network-Based Approach to Modeling Uncertainty in a Supply Chain," International Journal of Production Research, Vol. 42, No. 8, pp. 1639 – 1658.

Bosilj-Vuksic, V., Stemberger, M. I., Jaklic, J., and Kovacic, A., 2002, "Assessment of E-Business Transformation Using Simulation Modeling," Simulation, Vol. 78, No. 12, pp. 731 – 744.

Brailsford, S.C., Lattimer, V.A., Tarnaras, P., and Turnbull, J.C., 2004, "Emergency and On-Demand Health Care: Modeling a Large Complex System," Journal of the Operational Research Society, Vol. 55, pp. 34 - 42.

Browne, J., Harhen, J., and Shivnan, J., 1988, Production Management Systems, Addison-Wesley Wokingham, England.

Browning, T.R. and Eppinger, S.D., 2002, "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," IEEE Transactions on Engineering Management, Vol. 49, No. 4, pp. 428 – 442.

Caron, F. and Fiore, A., 1995, "'Engineer To Order' Companies: How to Integrate Manufacturing and Innovative Processes," International Journal of Project Management, Vol. 13, pp. 313 – 319.

Chen, L. and Jin, G., 2006, "Product Modeling for Multidisciplinary Collaborative Design," International Journal of Advanced Manufacturing Technology, Vol. 30, pp. 589 – 600.

Chen, S.J. and Lin, L., 2004, "Modeling Team Member Characteristics for the Formation of a Multifunctional Team in Concurrent Engineering," IEEE Transactions on Engineering Management, Vol. 51, No. 2, pp. 111 – 124.

Cheng, F.T., Yang, H.C., and Lin, J.Y., 2004, "Development of Holonic Information Coordination Systems with Failure-Recovery Considerations," IEEE Transactions on Automation Science and Engineering, Vol. 1, No. 1, pp. 58 – 72.

Choi, T.Y., Wu, Z., Ellram, L., and Koka, B.R., 2002, "Supplier-Supplier Relationships and Their Implications for Buyer-Supplier Relationships," IEEE Transactions on Engineering Management, Vol. 49, No. 2, pp. 119 – 130.

Coronado, A. E., Lyons, A. C., Kehoe, D. F., and Coleman, J., 2004, "Enabling Mass Customization: Extending Build-To-Order Concepts To Supply Chains," Production Planning & Control, Vol. 15, pp. 398 – 411.

Davis, S.M., 1987, "Future Perfect," New York: Addison Wesley.

Dekkers, R., 2006, "Engineering Management and the Order Entry Point," International Journal of Production Research, Vol. 44, pp. 4011 – 4025.

Demeester, L.L., and Qi, M., 2005, "Managing Learning Resources for Consecutive Product Generations," International Journal of Production Economics, Vol. 95, No. 2, pp, 265 – 283.

Eben-Chaime, M, Pliskin, N., and Sosna, D., 2004, "An Integrated Architecture for Simulation," Computers & Industrial Engineering, Vol. 46, pp. 159 – 170.

Everett, J.G. and Farghal, S., 1994, "Learning Curve Predictors for Construction Field Operations," Management Science, Vol. 120, No. 3, pp. 603 – 616.

Everett, J.G. and Farghal, S.H., 1997, "Data Representation for Predicting Performance with Learning Curves," Journal of Construction Engineering and Management, Vol. 123, pp. 46 – 52.

Faaland, B.H., Schmitt, T.G., and Arreola-Risa, A., 2004, "Economic Lot Scheduling with Lost Sales and Setup Times," IIE Transaction, Vol. 36, pp. 629 – 640.

Feitzinger, E. and Lee, H., 1997, "Mass Customization at Hewlett-Packard: The Power of Postponement," Harvard Business Review, Vol. 75, No. 1, pp. 116 – 121.

Fowler, J.W. and Rose, O., 2004, "Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems," Simulation, Vol. 80, No. 9, pp. 469 – 476.

Franke, N. and Piller, F., 2003, "Toolkits for User Innovation and Design: an Exploration of User Interaction and Value Creation in the Watch Market," Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization.

Gao, J.X., Aziz, H., Maropoulous, P.G., and Cheung, W.M., 2003, "Application of Product Data Management Technologies for Enterprise Integration," International Journal of Computer Integrated Manufacturing, Vol. 16, No. 7-8, pp. 491 – 500.

Garcia, J.S. and Dominguez, J.M.S., 2004, "MiiSD – Methodology of Integrated Information Systems Design," International Journal of Computer Integrated Manufacturing, Vol. 17, No. 6, pp. 493 – 503.

Gilmore, J.H. and Pine, B.J., 1997, "The Four Faces of Mass Customization," Harvard Business Review," Jan – Feb, pp. 91 – 101.

Gu, H. and Takahashi, H., 2000, "How bad may learning curves be?" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 10, pp. 1155 – 1167.

Hardgrave, B.C., and Johnson, R.A., 2003, "Toward an Information Systems Development Acceptance Model: The Case of Object-Oriented System Development," IEEE Transactions on Engineering Management, Vol. 50, No. 3, pp. 322 – 336.

Hartley, J.L., Lane, M.D., and Hong, Y., 2004, "An Exploration of the Adoption of E-Auctions in Supply Management," IEEE Transactions on Engineering Management, Vol. 51, No. 2, 153 – 161.

Herroelen, W. and Leus, R., 2004, "Robust and Reactive Project Scheduling: A Review and Classification of Procedures," International Journal of Production Research, Vol. 42, No. 8, pp. 1599 – 1620.

Hicks, C. and Braiden, P. M., 2000, "Computer-Aided Production Management Issues in the Egineer-To-Order Production of Complex Capital Goods Explored Using a Simulation Approach," International Journal of Production Research, Vol. 38, pp. 4783 – 4810.

Hicks, C., McGovern, T., and Earl, C.F., 2000, "Supply Chain Management: A Strategic Issue in Engineer To Order Manufacturing," International Journal of Production Economics, Vol. 65, pp. 179 – 190.

Hicks, C., McGovern, T., and Earl, C. F., 2001, "A Typology of UK Engineer-To-Order Companies," International Journal of Logistics: Research and Applications, Vol. 4, pp. 43 – 56.

Huang, C.Y. and Nof, S.Y., 2002, "Evaluation of Agent-Based Manufacturing Systems Based on a Parallel Simulator," Computers and Industrial Engineering, Vol. 43, pp. 529 – 552.

Hung, W.Y., Kucherenko, S., Samsatli, N.J., and Shah, N., 2004, "A Flexible and Generic Approach to Dynamic Modeling of Supply Chains," Journal of the Operational Research Society, Vol. 55, pp. 801 – 813.

Jaber, M.Y., and Guiffrida, A.L., 2004, "Learning Curves for Processes Generating Defects Requiring Reworks," European Journal of Operational Research, Vol. 159, No.3, pp. 663 – 672.

Jagstam, M. and Klingstam, P., 2002, "A Handbook for Integrating Discrete Event Simulation as an Aid in Conceptual Design of Manufacturing Systems," Proceedings of the Winter Simulation Conference, pp. 1940 – 1944.

Jeong, B. and Cho, H., 2006, "Feature Selection Techniques and Comparative Studies for Large-Scale Manufacturing Processes," International Journal of Advanced Manufacturing Technology, Vol. 28, pp. 1006 – 1011.

Jiao, J. and Tseng, M.M., 2004, "Customizability Analysis in Design for Mass Customization," Computer-Aided Design, Vol. 36, pp. 745 – 757.

Jiao, L. M., Khoo, L. P., and Chen, C.H., 2004, "An Intelligent Concurrent Design Task Planner for Manufacturing Systems," International Journal of Advanced Manufacturing Technology, Vol. 23, pp. 672 – 681.

Karcher, A. and Glander, M., 2003, "Global Distributed Engineering – Integrating Different Process Paradigms," Journal of Materials Processing Technology, Vol. 138, pp. 131 – 137.

Khouja, M., 2003, "Synchronization in Supply Chains: Implications for Design and Management," Journal of the Operational Research Society, Vol. 54, pp. 984 – 994.

Kirchner, S. and Marz, L., 2002, "Towards Self-Adaptive Production Systems: Modular Generic Simulation Models for Continuous Replanning and Reconfiguration," International Journal of Production Research, Vol. 40, No. 15, pp. 3627 – 3640.

Kock, N. and Davison, R., 2003, "Can Lean Media Support Knowledge Sharing? Investigating a Hidden Advantage of Process Improvement," IEEE Transactions on Engineering Management, Vol. 50, No. 2, pp. 151 – 163.

Kotak, D., Wu, S., Fleetwood, M., and Tamoto, H., 2003, "Agent-Based Holonic Design and Operations Environment for Distributed Manufacturing," Computer in Industry, Vol. 52, pp. 95 – 108.

Kurniawan, S.H., Tseng, M.M., and So, R.H.Y., 2003, "Consumer Decision-Making Process in Mass Customization," Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization.

Levin, N. and Globerson, S., 1993, "Generating Learning Curves for Individual Products from Aggregated Data," International Journal of Production Research, Vol. 31, No. 12, pp. 2807 – 2815.

Lin, H., Fan, Y., and Loiacono, E.T, 2004, "A Practical Scheduling Method Based on Workflow Management Technology," International Journal of Advanced Manufacturing Technology, Vol. 24, pp. 919 – 924.

Linton, J.D., 2003, "Facing the Challenges of Service Automation: An Enabler for E-Commerce and Productivity Gain in Traditional Services," IEEE Transactions on Engineering Management, Vol. 50, No. 4, pp. 478 – 484.

Little, D., Rollins, R., Peck, M., and Porter, J. K., 2000, "Integrated Planning and Scheduling in the Engineer-To-Order Sector," International Journal of Computer Integrated Manufacturing, Vol. 13, pp. 545 – 554.

Lou, P., Zhou, Z.D., Chen, Y.P., and Ai, W., 2004, "Study on Multi-Agent-Based Agile Supply Chain Management," International Journal of Advanced Manufacturing Technology, Vol. 23, pp. 197 – 203.

Lu, R., 2006, "Systems and Methods for Manufacturing a Product in a Pull and Push Manufacturing System and Associated Methods and Computer Program Products for Modeling the Same," United States Patent, US 6,983,189.

Lu, R., 2007, "Asynchronous Stochastic Learning Curve Effects in a Large Scale Production System," United States Patent pending, US Provisional Application No. 60/939,311.

Lu, R.F., Petersen, T.D., and Storch, R.L., 2007, "Asynchronous Stochastic Learning Curve Effects in Engineering-To-Order Customization Processes," International Journal of Production Research, submitted and accepted for publication.

Lu, R.F. and Storch, R.L., 2006, "Simulation Modeling of Customized Product Family Decision Points and Logistics," Proceedings of the IERC Conference, Orlando, FL, USA.

Lu, R.F. and Storch, R.L., 2005a, "Modeling of Supplier Event Management Influences in Large Manufacturing System Integration," Proceedings of APMS 2005 - IFIP 5.7 Conference on "Advances in Production Mgt Systems", NIST publishing, Washington DC, USA.

Lu, R.F. and Storch, R.L., 2005b, "Modeling of Customer Decision Point and Design Change Impact in Customized Large Manufacturing System Integration," Proceedings of the 3rd Interdisciplinary World Congress on Mass Customization and Personalization, Hong Kong, China.

MacCarthy, B., Brabazon, P.G., and Bramham, J., 2003, "Fundamental Modes of Operation For Mass Customization," International Journal of Production Economics, Vol. 85, pp. 289 – 304.

Montgomery, D.C., 2005, "Design and Analysis of Experiments," 6th Ed., New Jersey: John Wiley & Sons, Inc.

Mosheiov, G. and Sidney, J.B., 2003, "Scheduling with General Job-Dependent Learning Curves," European Journal of Operational Research, Vol. 147, No.3, pp. 665 – 670.

Mosheiov, G. and Sidney, J.B., 2005, "Note on Scheduling with General Learning Curves to Minimize the Number of Tardy Jobs," Journal of the Operational Research Society, Vol. 56, pp. 110 – 112.

Norfleet, D.A., 2005, "Loss of Learning in Disruption Claims," Cost Engineering, Vol. 47, No. 11, pp. 10 – 14.

Piller, F.T. and Stotko, C.M. 2002, "Mass Customization: Four Approaches to Deliver Customized Products and Services with Mass Production Efficiency," Engineering Management Conference IEMC '02, IEEE International, pp. 773 – 778.

Pine, J., Victor, B., and Boynton, A., 1993, "Making Mass Customization Work," Harvard Business Review, Vol. 71, No. 5, pp. 108 – 119.

Pope, E., 2001, "Now About This Assembly Line," CRAIN's Detroit Business, Vol.9, June 27.

Qiao, G., Lu, R.F., and McLean, C., 2006, "Flexible Manufacturing Systems for Mass Customisation Manufacturing," International Journal of Mass Customisation, Vol. 1, Issue 2/3, pp. 374 – 393.

Rahim, A.R.A. and Baksh, M.S.N., 2003, "The Need for a New Product Development Framework for Engineer-To-Order Products," European Journal of Innovation Management, Vol. 6, pp. 182 – 196.

Riis, J., Hansen, B.L., and Hvam, L, 2003, "Framework for Product Knowledge and Product Related Knowledge Which Supports Product Modeling for Mass Customization," Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization.

Rockwell Automation Technologies, 2007, "Arena Online Help," http://www.arenasimulation.com/support.

Roy, R. and Arunachalam, R., 2004, "Parallel Discrete Event Simulation Algorithm for Manufacturing Supply Chains," Journal of the Operational Research Society, Vol. 55, pp. 622 – 629.

Schruben, L.W. and Roeder, T.M., 2003, "Fast Simulation of Large-Scale Highly Congested Systems," Simulation, Vol. 79, No. 3.

Shafer, S.M., Nembhard, D.A., and Uzumeri, M.V., 2001, "The Effects of Worker Learning, Forgetting, and Heterogeneity on Assembly Line Productivity," Management Science, Vol. 47, No. 12, pp. 1639 – 1653.

Shah, R., Goldstein, S.M., and Ward, P.T., 2002, "Aligning Supply Chain Management Characteristics and Interorganizational Information System Types: An Exploratory Study," IEEE Transactions on Engineering Management, Vol. 49, No. 3, pp. 282 – 292.

Silveira, G.D., Borenstein, D., and Fogliatto, F.S., 2001, "Mass Customization: Literature Review and Research Directions," International Journal of Production Economics, Vol. 72, pp. 1 – 13.

Shunk, D. L., Kim, J. I., and Nam, H.Y., 2003, "The Application of an Integrated Enterprise Modeling Methodology – FIDO – to Supply Chain Integration Modeling," Computer & Industrial Engineering, Vol. 45, pp. 167 – 193.

Smith, J.S., 2003, "Survey on the Use of Simulation for Manufacturing System Design and Operation," Journal of Manufacturing Systems, Vol. 22, No. 2, pp. 157 – 171.

Stummer, C., and Heidenberger, K., 2003, "Interactive R&D Portfolio Analysis with Project Interdependencies and Time Profiles of Multiple Objectives," IEEE Transactions on Engineering Management, Vol. 50, No. 2, pp. 175 – 183.

Sugimura, N, Hino, R., and Moriwaki, T., 2001, "Integrated Process Planning and Scheduling in Holonic Manufacturing Systems," Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning, pp. 250 – 255.

Sugimura, N., Shrestha, R., and Inoue, J., 2003, "Integrated Process Planning and Scheduling in Holonic Manufacturing Systems – Optimization Based on Shop Time and Machining Cost," Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, pp. 36 – 41.

Teunter, R., van der Laan, E., and Vlachos, D., 2004, "Inventory Strategies for Systems with Fast Remanufacturing," Journal of the Operational Research Society, Vol. 55, pp. 475 – 484.

Thomas, H.R., Mathews, C.T., and Ward, J.G., 1986, "Learning Curve Models of Construction Productivity," Journal of Construction Engineering and Management, ASCE, Vol. 112, No. 2, pp. 245 – 258.

Tseng, M.M., Jiao, J., and Su, C. J., 1998, "Virtual Prototyping for Customized Product Development," Integrated Manufacturing Systems, No. 6, pp. 334 – 343, MCB University.

Tu, Q., Vonderembse, M.A., Ragu-Nathan, T.S., and Ragu-Nathan, B., 2004, "Measuring Modularity-Based Manufacturing Practices and Their Impact on Mass Customization Capability: A Customer-Driven Perspective," Decision Sciences, Vol. 35, No. 2, pp. 147 – 168.

Verbraeck, A. and Versteegt, C., 2001, "Logistic Control for Fully Automated Large Scale Freight Transport Systems; Event Based Control for the Underground Logistic System Schiphol," IEEE Intelligent Transportation Systems Conference Proceedings, Oakland, CA.

Villa, A., 2002, "Emerging Trends in Large-Scale Supply Chain Management," International Journal of Production Research, Vol. 40, No. 15, pp. 3487 – 3498.

Vits, J., Gelders, L., and Pintelon, L., 2006, "Production Process Changes: A Dynamic Programming Approach to Manage Effective Capacity and Experience," International Journal of Production Economics, Vol. 104, pp. 473 – 481.

Wang, G., Huang, S.H., and Dismukes, J.P., 2005, "Manufacturing Supply Chain Design and Evaluation," International Journal of Advanced Manufacturing Technology, Vol. 25, pp. 93 – 100.

Wang, Y. and Jiao, J., 2003, "Product Family Configuration Design Evaluation," Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization.

Wideman, M., 1994, "A Pragmatic Approach to Using Resource Loading, Production and Learning Curves on Construction Projects," Canadian Journal of Civil Engineering, Vol. 21, pp. 939 – 953.

Wikner, J., 2003, "Continuous-Time Dynamic Modeling of Variable Lead Times," International Journal of Production Research, Vol. 41, No. 12, pp. 2787 – 2798.

Womack, J.P., Jones, D.T., and Roos, D., 1990, "The Machine That Changed The World," New York: Rawson Associates.

Wright, T.P., 1936, "Factors Affecting the Cost of Airplanes," Journal of aeronautical Science, February, pp. 124 – 125.

Wu, M.C., and Sun, S.H., 2006, "A Project Scheduling and Staff Assignment Model Considering Learning Effect," International Journal of Advanced Manufacturing Technology, Vol. 28, pp. 1190 – 1195.

Xie, S.Q., Xu, X., and Tu, Y.L., 2005a, "A Reconfigurable Platform in Support of One-Of-A-Kind Product Development," International Journal of Production Research, Vol. 43, pp, 1889 – 1910.

Xie, H., Henderson, P., and Kernahan, M., 2005b, "Modeling and Solving Engineering Product Configuration Problems by Constraint Satisfaction," International Journal of Production Research, Vol. 43, pp. 4455 – 4469.

Xu, D., and Yan, H.S., 2006, "An Intelligent Estimation Method for Product Design Time," International Journal of Advanced Manufacturing Technology, Vol. 30, pp. 601 – 613.

Yoshimura, M., Izui, K., and Fujimi, Y., 2003, "Optimizing the Decision-Making Process for Large-Scale Design Problems According to Criteria Interrelationships," International Journal of Production Research, Vol. 41, No. 9, pp. 1987 – 2002.

Zhang, M., Chen, Y., and Tseng, M.M., 2003, "Process Planning for Mass Customization," Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization.

Zhang, W.Y., Tor, S.Y., and Britton, G.A., 2006, "Managing Modularity in Product Family Design with Functional Modeling," International Journal of Advanced Manufacturing Technology, Vol. 30, pp. 579 – 588.

Appendix A Source Data

Data source:

Data used in this research have been extracted from a set of airplane production schedules. This research proposal was a major part of a submitted journal paper which received release for public publication by The Boeing Company. This data set has 15% ETO. Three more sets of the same data structure were used in the research to reflect the 0%, 2%, and 8% ETOs.

Meanings of column headings:

m : The sequence number, which the learning curve flattens, of the ith component

N15 : The on-time performance of the xth unit of the ith component with 15% ETO.

f11, f21, f31 : The actual xth unit of the ith component arrival time at the final assembly

Fij : The targeted xth unit of the ith component arrival time at the final assembly

t11, t21, t31 : The actual total flow time of the xth unit of the ith component

T11, T21, T31 : The targeted total flow time of the xth unit of the ith component

S11, S21, S31 : The targeted xth unit of the ith component production starting time

s11, s21, s31 : The actual xth unit of the ith component production starting time

y11, y21, y31 : Time needed to process the x¬ith unit of the ith component in the system

y11ETO, y21ETO, y31ETO : Time needed to process the x¬ith unit of the ith component in the system with the ETO effect in the system

Finish Rate = time interval between two adjacent Fi,x's.

m	N15	f11	f21	f31	Fij	t11	t21	t31	T11	T21	T31
1	40.24	14112.52	14072.28	14100.98	14099	546.73	229.22	255.68	533.00	256.00	253.00
2	41.27	14110.82	14103.15	14144.42	14137	406.61	186.22	230.81	433.00	220.00	224.00
3	58.98	14190.55	14131.57	14176.24	14165	403.08	162.46	217.77	378.00	196.00	207.00
4	47.81	14188.98	14141.17	14171.33	14184	363.57	148.98	193.33	359.00	192.00	206.00
5	24.79	14182.74	14157.96	14180.65	14197	322.94	143.96	183.98	337.00	183.00	200.00
6	23.90	14193.45	14174.17	14198.07	14209	302.48	142.62	183.09	318.00	177.00	194.00
7	24.62	14216.75	14192.13	14215.72	14221	301.70	142.40	184.96	306.00	171.00	190.00
8	42.35	14236.16	14193.80	14228.66	14233	297.34	125.89	181.90	294.00	165.00	186.00
9	28.03	14243.06	14215.96	14243.99	14245	281.00	130.02	181.03	283.00	159.00	182.00
10	32.40	14239.77	14224.95	14257.36	14257	257.26	121.74	179.34	275.00	154.00	179.00
11	22.12	14251.87	14240.84	14262.95	14269	250.00	121.93	170.25	267.00	150.00	176.00
12	12.70	14252.12	14251.29	14263.99	14281	230.41	119.14	155.67	259.00	149.00	173.00
13	31.96	14289.88	14257.92	14275.81	14291	254.02	113.76	155.80	255.00	147.00	171.00
14	24.47	14294.15	14273.10	14297.57	14301	244.19	117.11	165.67	251.00	145.00	169.00
15	14.95	14302.05	14287.10	14300.36	14311	238.18	119.10	156.39	247.00	143.00	167.00
16	31.72	14287.03	14290.21	14318.76	14321	208.98	110.18	162.76	243.00	141.00	165.00
17	28.08	14320.59	14295.30	14323.38	14329	228.59	104.27	155.28	237.00	138.00	161.00
18	22.88	14306.95	14312.28	14329.83	14337	201.95	110.29	151.00	232.00	135.00	158.00
19	19.14	14335.60	14322.76	14341.90	14345	217.83	109.85	151.66	227.00	132.00	155.00
20	21.55	14328.59	14328.11	14349.67	14353	198.55	104.19	148.89	223.00	129.00	152.00
21	25.17	14337.36	14344.50	14362.53	14361	195.72	110.44	150.58	219.00	127.00	149.00
22	10.65	14362.79	14352.13	14360.46	14369	208.91	108.12	137.39	215.00	125.00	146.00
23	37.92	14350.03	14349.19	14387.11	14377	185.97	95.15	154.02	213.00	123.00	144.00
24	28.22	14362.17	14361.15	14389.37	14385	188.10	97.15	146.08	211.00	121.00	142.00
25	32.68	14365.00	14378.69	14397.67	14393	181.18	104.69	144.74	209.00	119.00	140.00
26	16.82	14382.64	14385.31	14399.46	14401	188.71	101.28	136.54	207.00	117.00	138.00
27	20.86	14403.53	14391.85	14412.71	14409	199.53	97.81	140.49	205.00	115.00	137.00
28	21.56	14403.00	14399.11	14420.67	14416	188.78	94.91	139.65	202.00	112.00	135.00
29	23.27	14417.72	14413.43	14436.70	14422	193.81	100.41	146.67	198.00	109.00	132.00
30	25.25	14411.00	14417.83	14436.25	14428	178.10	95.82	137.26	195.00	106.00	129.00
31	17.22	14433.08	14420.02	14437.24	14434	191.18	89.93	132.13	192.00	104.00	129.00
32	28.26	14428.10	14429.86	14456.36	14440	177.33	91.98	145.47	189.00	102.00	129.00
33	28.10	14431.59	14443.03	14459.69	14446	171.55	99.08	142.70	186.00	102.00	129.00
34	27.05	14436.51	14444.27	14463.57	14452	168.46	93.99	140.70	184.00	102.00	129.00
35	16.14	14454.73	14446.79	14462.94	14458	178.81	90.89	133.78	182.00	102.00	129.00
36	13.15	14456.89	14454.76	14467.91	14464	173.11	92.57	132.91	180.00	102.00	129.00
37	15.72	14451.17	14453.15	14466.89	14470	159.16	85.21	125.82	178.00	102.00	129.00
38	22.89	14479.34	14466.82	14489.71	14476	179.53	92.66	142.74	176.00	102.00	129.00
39	9.12	14473.02	14474.97	14482.14	14482	165.00	92.94	127.00	174.00	100.00	127.00
40	12.99	14488.00	14484.10	14497.09	14488	171.91	94.08	134.09	172.00	98.00	125.00
41	18.59	14497.28	14480.33	14498.92	14494	173.25	82.26	128.08	170.00	96.00	123.00
42	16.15	14493.12	14487.95	14504.10	14500	161.09	81.98	125.13	168.00	94.00	121.00
43	18.31	14510.93	14503.76	14522.06	14506	171.15	89.80	137.02	166.00	92.00	121.00
44	20.01	14508.06	14510.32	14528.07	14512	160.04	88.44	137.04	164.00	90.00	121.00

m	N15	f11	f21	f31	Fij	t11	t21	t31	T11	T21	T31
45	16.30	14512.18	14520.43	14528.48	14518	156.20	90.52	131.30	162.00	88.00	121.00
46	25.16	14512.92	14519.75	14538.07	14524	149.07	83.91	135.18	160.00	88.00	121.00
47	14.48	14529.93	14523.09	14537.58	14530	158.14	81.29	128.58	158.00	88.00	121.00
48	18.38	14541.95	14531.24	14549.63	14534	161.95	83.22	134.61	154.00	86.00	119.00
49	7.20	14548.81	14541.61	14545.00	14538	160.73	87.62	123.99	150.00	84.00	117.00
50	10.68	14544.46	14547.68	14555.14	14542	150.46	87.70	129.10	148.00	82.00	116.00
51	9.05	14546.32	14543.38	14552.43	14546	146.33	77.38	121.28	146.00	80.00	115.00
52	11.46	14567.84	14556.37	14558.89	14550	161.71	84.34	122.97	144.00	78.00	114.00
53	11.55	14560.99	14562.75	14572.53	14554	148.88	85.77	132.50	142.00	77.00	114.00
54	16.01	14576.00	14559.98	14570.09	14558	157.93	78.02	126.18	140.00	76.00	114.00
55	11.96	14575.94	14563.98	14568.82	14562	151.96	77.04	120.82	138.00	75.00	114.00
56	12.29	14571.39	14573.69	14583.68	14566	141.41	82.75	131.78	136.00	75.00	114.00
57	14.67	14589.17	14574.50	14583.30	14570	153.16	79.49	127.16	134.00	75.00	114.00
58	6.70	14590.88	14584.18	14587.38	14574	148.84	85.14	127.47	132.00	75.00	114.00
59	6.82	14592.32	14587.42	14594.24	14578	144.33	84.51	130.25	130.00	75.00	114.00
60	14.09	14596.06	14581.97	14584.57	14582	142.14	75.04	116.46	128.00	75.00	114.00
61	3.98	14594.83	14590.85	14593.07	14586	134.83	79.97	120.97	126.00	75.00	114.00
62	6.20	14598.92	14597.71	14592.71	14590	134.90	82.67	116.78	126.00	75.00	114.00
63	18.43	14614.29	14595.86	14603.48	14594	146.20	76.88	123.50	126.00	75.00	114.00
64	12.31	14606.66	14599.82	14612.12	14598	134.76	76.80	127.97	126.00	75.00	114.00
65	14.47	14621.06	14606.59	14611.17	14602	144.89	79.60	123.00	126.00	75.00	114.00
66	24.96	14629.86	14604.90	14616.70	14606	149.77	73.97	124.70	126.00	75.00	114.00
67	7.70	14619.02	14611.33	14616.68	14610	134.97	76.41	120.69	126.00	75.00	114.00
68	23.72	14636.87	14614.87	14613.15	14614	148.92	75.75	113.16	126.00	75.00	114.00
69	15.48	14632.78	14617.30	14625.62	14618	140.70	74.17	121.63	126.00	75.00	114.00
70	11.80	14632.98	14622.17	14621.18	14622	137.02	75.15	113.21	126.00	75.00	114.00
71	9.71	14634.39	14624.68	14625.05	14626	134.39	73.64	112.99	126.00	75.00	114.00
72	16.40	14647.68	14631.29	14637.03	14630	143.68	76.29	120.94	126.00	75.00	114.00
73	11.13	14643.96	14635.98	14632.83	14634	135.97	76.92	112.83	126.00	75.00	114.00
74	14.69	14653.20	14638.51	14647.19	14638	141.27	75.52	123.08	126.00	75.00	114.00
75	10.02	14654.07	14644.50	14654.52	14642	138.04	77.60	126.51	126.00	75.00	114.00
76	12.03	14646.21	14643.71	14655.74	14646	126.25	72.80	123.80	126.00	75.00	114.00
77	17.92	14666.44	14648.52	14652.98	14650	142.49	73.52	116.90	126.00	75.00	114.00
78	2.24	14653.05	14655.30	14653.47	14654	125.10	76.28	113.49	126.00	75.00	114.00
79	13.86	14664.26	14654.14	14668.00	14657	132.19	71.26	124.00	125.00	74.00	113.00
80	9.63	14668.71	14659.08	14668.06	14660	132.66	72.18	120.08	124.00	73.00	112.00
81	8.67	14669.69	14661.02	14664.26	14663	130.65	71.02	113.27	124.00	73.00	112.00
82	4.53	14672.36	14667.83	14671.00	14666	130.26	74.83	117.01	124.00	73.00	112.00
83	10.74	14671.06	14665.42	14676.16	14669	126.05	69.51	119.23	124.00	73.00	112.00
84	9.03	14681.99	14676.56	14672.96	14672	134.00	77.65	112.90	124.00	73.00	112.00
85	6.08	14681.48	14676.42	14682.51	14675	130.48	74.43	119.53	124.00	73.00	112.00
86	8.56	14680.74	14682.03	14689.30	14678	126.68	77.01	123.28	124.00	73.00	112.00
87	13.66	14693.74	14680.08	14686.64	14681	136.74	72.08	117.56	124.00	73.00	112.00
88	13.74	14693.96	14680.22	14687.76	14684	133.94	69.26	115.72	124.00	73.00	112.00

m	N15	f11	f21	f31	Fij	t11	t21	t31	T11	T21	T31
89	4.99	14684.43	14689.42	14686.63	14687	121.47	75.38	111.64	124.00	73.00	112.00
90	1.03	14686.67	14687.10	14687.70	14690	120.76	70.12	109.72	124.00	73.00	112.00
91	15.24	14704.03	14689.27	14704.51	14693	135.05	69.22	123.52	124.00	73.00	112.00
92	9.47	14706.16	14696.69	14698.54	14696	133.14	72.74	113.55	123.00	72.00	111.00
93	13.39	14696.45	14698.25	14709.84	14699	119.41	70.24	120.84	122.00	71.00	110.00
94	8.65	14709.53	14706.57	14715.22	14702	128.64	74.59	122.21	121.00	70.00	109.00
95	15.39	14703.66	14707.31	14719.05	14705	118.65	71.35	122.09	120.00	69.00	108.00
96	3.27	14705.55	14707.64	14708.83	14708	116.55	67.64	107.86	119.00	68.00	107.00
97	15.62	14716.69	14711.06	14726.68	14711	123.73	67.02	121.75	118.00	67.00	106.00
98	14.33	14722.14	14716.13	14730.46	14714	125.21	68.13	121.48	117.00	66.00	105.00
99	10.12	14727.07	14720.47	14730.59	14717	126.07	69.46	118.54	116.00	66.00	105.00
100	10.34	14732.12	14723.11	14721.78	14720	127.12	69.11	106.78	115.00	66.00	105.00

m	S11	S21	S31	s11	s21	s31
1	13566.00	13843.00	13846.00	13565.79	13843.06	13845.30
2	13704.00	13917.00	13913.00	13704.22	13916.93	13913.61
3	13787.00	13969.00	13958.00	13787.47	13969.11	13958.48
4	13825.00	13992.00	13978.00	13825.41	13992.19	13978.00
5	13860.00	14014.00	13997.00	13859.81	14014.00	13996.67
6	13891.00	14032.00	14015.00	13890.97	14031.55	14014.98
7	13915.00	14050.00	14031.00	13915.05	14049.72	14030.76
8	13939.00	14068.00	14047.00	13938.82	14067.92	14046.76
9	13962.00	14086.00	14063.00	13962.06	14085.94	14062.96
10	13982.00	14103.00	14078.00	13982.51	14103.21	14078.01
11	14002.00	14119.00	14093.00	14001.87	14118.91	14092.71
12	14022.00	14132.00	14108.00	14021.71	14132.15	14108.32
13	14036.00	14144.00	14120.00	14035.86	14144.16	14120.01
14	14050.00	14156.00	14132.00	14049.97	14155.99	14131.90
15	14064.00	14168.00	14144.00	14063.87	14168.01	14143.97
16	14078.00	14180.00	14156.00	14078.05	14180.04	14156.00
17	14092.00	14191.00	14168.00	14092.00	14191.03	14168.10
18	14105.00	14202.00	14179.00	14105.00	14201.99	14178.83
19	14118.00	14213.00	14190.00	14117.77	14212.92	14190.24
20	14130.00	14224.00	14201.00	14130.04	14223.92	14200.78
21	14142.00	14234.00	14212.00	14141.63	14234.06	14211.95
22	14154.00	14244.00	14223.00	14153.87	14244.01	14223.07
23	14164.00	14254.00	14233.00	14164.06	14254.04	14233.09
24	14174.00	14264.00	14243.00	14174.07	14264.00	14243.29
25	14184.00	14274.00	14253.00	14183.81	14274.00	14252.93
26	14194.00	14284.00	14263.00	14193.93	14284.03	14262.92
27	14204.00	14294.00	14272.00	14204.00	14294.05	14272.23
28	14214.00	14304.00	14281.00	14214.22	14304.20	14281.02

m	S11	S21	S31	s11	s21	s31
29	14224.00	14313.00	14290.00	14223.91	14313.03	14290.03
30	14233.00	14322.00	14299.00	14232.90	14322.01	14298.99
31	14242.00	14330.00	14305.00	14241.89	14330.09	14305.11
32	14251.00	14338.00	14311.00	14250.77	14337.87	14310.89
33	14260.00	14344.00	14317.00	14260.04	14343.95	14316.99
34	14268.00	14350.00	14323.00	14268.05	14350.28	14322.87
35	14276.00	14356.00	14329.00	14275.92	14355.90	14329.16
36	14284.00	14362.00	14335.00	14283.77	14362.19	14335.00
37	14292.00	14368.00	14341.00	14292.01	14367.94	14341.07
38	14300.00	14374.00	14347.00	14299.81	14374.15	14346.97
39	14308.00	14382.00	14355.00	14308.02	14382.03	14355.15
40	14316.00	14390.00	14363.00	14316.09	14390.02	14363.00
41	14324.00	14398.00	14371.00	14324.03	14398.07	14370.84
42	14332.00	14406.00	14379.00	14332.03	14405.97	14378.97
43	14340.00	14414.00	14385.00	14339.77	14413.96	14385.04
44	14348.00	14422.00	14391.00	14348.02	14421.88	14391.03
45	14356.00	14430.00	14397.00	14355.98	14429.91	14397.17
46	14364.00	14436.00	14403.00	14363.85	14435.84	14402.89
47	14372.00	14442.00	14409.00	14371.79	14441.80	14409.00
48	14380.00	14448.00	14415.00	14379.99	14448.02	14415.02
49	14388.00	14454.00	14421.00	14388.08	14454.00	14421.00
50	14394.00	14460.00	14426.00	14394.00	14459.98	14426.04
51	14400.00	14466.00	14431.00	14399.99	14466.00	14431.16
52	14406.00	14472.00	14436.00	14406.13	14472.04	14435.93
53	14412.00	14477.00	14440.00	14412.11	14476.97	14440.04
54	14418.00	14482.00	14444.00	14418.06	14481.96	14443.91
55	14424.00	14487.00	14448.00	14423.98	14486.94	14447.99
56	14430.00	14491.00	14452.00	14429.98	14490.95	14451.91
57	14436.00	14495.00	14456.00	14436.00	14495.01	14456.13
58	14442.00	14499.00	14460.00	14442.05	14499.04	14459.91
59	14448.00	14503.00	14464.00	14447.99	14502.92	14463.99
60	14454.00	14507.00	14468.00	14453.92	14506.93	14468.11
61	14460.00	14511.00	14472.00	14460.01	14510.89	14472.11
62	14464.00	14515.00	14476.00	14464.02	14515.04	14475.93
63	14468.00	14519.00	14480.00	14468.09	14518.98	14479.98
64	14472.00	14523.00	14484.00	14471.90	14523.02	14484.15
65	14476.00	14527.00	14488.00	14476.17	14527.00	14488.17
66	14480.00	14531.00	14492.00	14480.09	14530.93	14492.00
67	14484.00	14535.00	14496.00	14484.05	14534.92	14495.99
68	14488.00	14539.00	14500.00	14487.95	14539.11	14499.99
69	14492.00	14543.00	14504.00	14492.08	14543.13	14503.99
70	14496.00	14547.00	14508.00	14495.96	14547.01	14507.97
71	14500.00	14551.00	14512.00	14499.99	14551.04	14512.06
72	14504.00	14555.00	14516.00	14504.00	14555.00	14516.08

m	S11	S21	S31	s11	s21	s31
73	14508.00	14559.00	14520.00	14507.99	14559.06	14520.00
74	14512.00	14563.00	14524.00	14511.93	14562.99	14524.11
75	14516.00	14567.00	14528.00	14516.02	14566.90	14528.00
76	14520.00	14571.00	14532.00	14519.95	14570.91	14531.94
77	14524.00	14575.00	14536.00	14523.95	14574.99	14536.08
78	14528.00	14579.00	14540.00	14527.96	14579.02	14539.98
79	14532.00	14583.00	14544.00	14532.07	14582.89	14544.00
80	14536.00	14587.00	14548.00	14536.05	14586.89	14547.99
81	14539.00	14590.00	14551.00	14539.04	14590.01	14550.99
82	14542.00	14593.00	14554.00	14542.10	14593.00	14553.99
83	14545.00	14596.00	14557.00	14545.00	14595.91	14556.92
84	14548.00	14599.00	14560.00	14547.99	14598.91	14560.06
85	14551.00	14602.00	14563.00	14551.00	14602.00	14562.98
86	14554.00	14605.00	14566.00	14554.07	14605.02	14566.02
87	14557.00	14608.00	14569.00	14557.00	14608.00	14569.09
88	14560.00	14611.00	14572.00	14560.03	14610.96	14572.05
89	14563.00	14614.00	14575.00	14562.96	14614.04	14574.99
90	14566.00	14617.00	14578.00	14565.91	14616.98	14577.98
91	14569.00	14620.00	14581.00	14568.98	14620.05	14580.99
92	14573.00	14624.00	14585.00	14573.02	14623.96	14584.99
93	14577.00	14628.00	14589.00	14577.04	14628.01	14589.00
94	14581.00	14632.00	14593.00	14580.90	14631.98	14593.01
95	14585.00	14636.00	14597.00	14585.01	14635.96	14596.97
96	14589.00	14640.00	14601.00	14589.00	14640.00	14600.96
97	14593.00	14644.00	14605.00	14592.96	14644.04	14604.93
98	14597.00	14648.00	14609.00	14596.92	14648.00	14608.98
99	14601.00	14651.00	14612.00	14601.00	14651.01	14612.06
100	14605.00	14654.00	14615.00	14605.00	14654.00	14615.00

m	y11	y21	y31	y11ETO	y21ETO	y31ETO	Finish Rate
1	506.47	211.99	229.91	539.19	226.22	246.87	49
2	404.80	177.44	204.70	412.89	185.58	229.86	38
3	355.26	159.90	191.21	391.80	159.98	195.58	28
4	323.84	148.52	182.18	362.34	164.40	187.28	19
5	301.39	140.25	175.47	308.20	147.76	182.64	13
6	284.21	133.84	170.18	291.34	141.66	170.60	12
7	270.45	128.65	165.82	274.27	144.48	175.62	12
8	259.07	124.31	162.14	272.92	129.94	163.29	12
9	249.43	120.61	158.96	284.82	131.43	180.28	12
10	241.11	117.39	156.17	243.44	118.89	166.11	12
11	233.83	114.55	153.69	239.49	131.25	165.92	12
12	227.37	112.02	151.46	245.28	118.02	162.15	12

m	y11	y21	y31	y11ETO	y21ETO	y31ETO	Finish Rate
13	221.59	109.75	149.43	245.38	117.14	170.07	10
14	216.36	107.68	147.58	235.66	112.36	148.39	10
15	211.61	105.79	145.88	219.39	121.49	148.38	10
16	207.26	104.05	144.31	233.97	112.64	160.58	10
17	203.25	102.44	142.84	212.69	104.53	161.15	8
18	199.55	100.95	141.48	223.03	109.45	161.51	8
19	196.10	99.56	140.20	220.13	114.19	142.31	8
20	192.89	98.26	138.99	216.25	101.71	139.39	8
21	189.89	97.03	137.86	208.98	107.45	146.35	8
22	187.06	95.88	136.78	192.14	96.22	146.42	8
23	184.40	94.79	135.77	202.56	106.77	141.55	8
24	181.90	93.76	134.80	202.14	104.94	140.23	8
25	179.52	92.79	133.88	193.32	104.55	143.25	8
26	177.27	91.86	133.00	179.87	99.66	149.60	8
27	175.13	90.97	132.15	185.17	98.65	134.59	8
28	173.09	90.13	131.35	184.94	100.39	132.79	7
29	171.14	89.32	130.58	181.08	93.06	132.10	6
30	169.29	88.54	129.83	171.41	99.33	148.71	6
31	167.51	87.80	129.12	182.81	100.15	146.43	6
32	165.81	87.09	128.43	170.46	90.01	128.58	6
33	164.17	86.40	127.77	178.62	96.39	132.93	6
34	162.60	85.74	127.13	165.04	90.07	133.36	6
35	161.09	85.11	126.51	178.13	97.03	130.54	6
36	159.64	84.50	125.92	176.34	87.07	133.28	6
37	158.23	83.90	125.34	161.02	90.99	137.49	6
38	156.88	83.33	124.78	177.07	89.86	143.13	6
39	155.58	82.78	124.23	166.76	90.60	124.65	6
40	154.31	82.24	123.70	165.23	86.20	124.93	6
41	153.09	81.72	123.19	170.07	84.33	133.49	6
42	151.91	81.22	122.69	152.61	87.54	125.50	6
43	150.76	80.73	122.21	171.11	92.25	133.84	6
44	149.65	80.25	121.74	169.96	89.52	136.82	6
45	148.57	79.79	121.28	158.28	81.69	131.72	6
46	147.52	79.34	120.83	147.62	86.74	129.80	6
47	146.51	78.91	120.40	165.83	85.80	136.87	6
48	145.52	78.48	119.97	158.76	83.47	134.98	4
49	144.55	78.07	119.55	158.21	79.70	120.04	4
50	143.62	77.66	119.15	153.10	85.98	123.46	4
51	142.70	77.27	118.75	146.72	88.81	119.05	4
52	141.81	76.88	118.37	159.87	84.41	126.13	4
53	140.95	76.51	117.99	150.81	85.39	119.04	4
54	140.10	76.14	117.62	160.46	76.89	122.54	4
55	139.28	75.79	117.26	159.40	85.69	121.96	4
56	138.47	75.44	116.90	140.87	77.62	121.59	4

m	y11	y21	y31	y11ETO	y21ETO	y31ETO	Finish Rate
57	137.68	75.09	116.55	150.73	75.29	119.89	4
58	136.92	74.76	116.21	157.45	80.85	132.82	4
59	136.16	74.43	115.88	138.69	80.69	123.57	4
60	135.43	74.11	115.55	150.89	82.20	131.65	4
61	134.71	73.80	115.23	154.02	75.24	131.82	4
62	134.01	73.49	114.92	144.18	83.89	115.48	4
63	133.32	73.19	114.61	142.63	83.21	129.31	4
64	132.64	72.89	114.31	149.06	77.94	114.36	4
65	131.98	72.60	114.01	149.64	81.23	127.68	4
66	131.34	72.32	113.72	144.93	79.14	118.98	4
67	130.70	72.04	113.43	131.94	81.28	118.18	4
68	130.08	71.77	113.15	143.29	81.66	128.79	4
69	129.47	71.50	112.87	147.16	74.91	124.38	4
70	128.87	71.24	112.60	140.76	73.70	118.69	4
71	128.29	70.98	112.33	140.26	73.67	125.14	4
72	127.71	70.72	112.06	142.42	74.10	115.70	4
73	127.14	70.47	111.80	139.12	70.65	113.61	4
74	126.59	70.23	111.55	133.83	71.27	119.72	4
75	126.04	69.99	111.30	129.65	71.55	120.98	4
76	125.51	69.75	111.05	139.79	73.23	122.00	4
77	124.98	69.51	110.81	138.78	76.33	119.02	4
78	124.46	69.28	110.57	125.78	72.07	117.44	4
79	123.95	69.06	110.33	141.30	72.06	119.71	3
80	123.45	68.84	110.10	136.29	71.66	111.33	3
81	122.96	68.62	109.87	133.23	78.13	114.58	3
82	122.47	68.40	109.64	132.80	71.49	114.27	3
83	122.00	68.19	109.42	130.71	68.90	122.79	3
84	121.53	67.98	109.20	129.43	78.01	110.27	3
85	121.06	67.77	108.98	121.72	74.25	112.86	3
86	120.61	67.57	108.77	137.28	70.74	113.94	3
87	120.16	67.37	108.56	130.52	74.85	122.70	3
88	119.72	67.17	108.35	135.71	71.27	116.91	3
89	119.29	66.98	108.14	132.83	67.27	122.00	3
90	118.86	66.79	107.94	123.86	68.93	120.01	3
91	118.43	66.60	107.74	120.28	70.40	109.89	3
92	118.02	66.41	107.54	120.65	71.09	110.37	3
93	117.61	66.23	107.34	120.89	75.32	108.93	3
94	117.20	66.04	107.15	126.44	70.10	120.17	3
95	116.81	65.86	106.96	119.23	74.04	108.69	3
96	116.41	65.69	106.77	117.43	68.80	107.34	3
97	116.03	65.51	106.59	124.08	73.00	107.98	3
98	115.64	65.34	106.40	124.18	67.22	114.90	3
99	115.27	65.17	106.22	118.81	70.17	113.07	3
100	114.89	65.00	106.04	120.48	68.52	113.89	3

Appendix B Correlations among factors

Correlations: f11, Fi, t11, T_11, S_11, s11, y11, FinishRate

	f11	Fi	t11	T_11	S_11	s11	y11
Fi	0.999						
t11	-0.881	-0.889					
T_11	-0.928	-0.931	0.990				
S_11	0.991	0.993	-0.935	-0.968			
s11	0.991	0.993	-0.936	-0.968	1.000		
y11	-0.881	-0.889	1.000	0.990	-0.935	-0.936	
FinishRate	-0.745	-0.755	0.955	0.921	-0.820	-0.821	0.955

Cell Contents: Pearson correlation

Appendix C Simulation Source Codes

Discrete Event Simulation model source code.

These codes presents the DES model explained in the Section 7.2.

```
;
;
;     Model statements for module:  BasicProcess.Create 5 (S11_ Main Gear
Order)
;

39$           CREATE,
1,DaysToBaseTime(0.0),MainGear:DaysToBaseTime(1),1:NEXT(40$);

40$           ASSIGN:        S11_ Main Gear Order.NumberOut=S11_ Main Gear
Order.NumberOut + 1:NEXT(6$);


;
;
;     Model statements for module:  BasicProcess.Assign 15 (Assign CurveRate
MainGear)
;
6$            ASSIGN:        varSlopeMainGear=LOG(80/100)/LOG(2):
                             varMainGearLN1=10 *  * (LOG(varMainGearLN100) -
varSlopeMainGear*LOG(100)):NEXT(9$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 1 (Read Main Gear
Order Schedule)
;
9$            READ,          ASLC_A1_Data,RECORDSET(Recordset_s11):
                             MainGearTime:NEXT(10$);


;
;
;     Model statements for module:  BasicProcess.Separate 1 (Separate Main
Gear)
;
10$           DUPLICATE,     100 - 50:
                             1,45$,50:NEXT(44$);

44$           ASSIGN:        Separate Main Gear.NumberOut Orig=Separate Main
Gear.NumberOut Orig + 1:NEXT(11$);

45$           ASSIGN:        Separate Main Gear.NumberOut Dup=Separate Main
Gear.NumberOut Dup + 1:NEXT(9$);


;
```

```
;
;     Model statements for module:  AdvancedProcess.Delay 1 (Delay MainGear
Order)
;
11$           DELAY:          MainGearTime,,Other:NEXT(33$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 9 (Write s11 out)
;
33$           WRITE,          ASLC_A1_Data,RECORDSET(Recordset_s11output):
                              MainGearTime:NEXT(5$);


;
;
;     Model statements for module:  BasicProcess.Assign 14 (Assign Main Gear
Serial Number)
;
5$            ASSIGN:         varPartMainGear=varPartMainGear + 1:
                              attSNy11=varPartMainGear:NEXT(12$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 1 (Route MainGear
Order)
;
12$           ROUTE:          0.,StnMainGearOrder;


;
;
;     Model statements for module:  AdvancedTransfer.Station 1 (Main Gear Order
Arrival)
;

13$           STATION,        StnMainGearOrder;
48$           DELAY:          0.0,,VA:NEXT(7$);


;
;
;     Model statements for module:  BasicProcess.Process 13 (y11_Process)
;
7$            ASSIGN:         y11_Process.NumberIn=y11_Process.NumberIn + 1:
                              y11_Process.WIP=y11_Process.WIP+1;
78$           STACK,          1:Save:NEXT(52$);

52$           QUEUE,          y11_Process.Queue;
51$           SEIZE,          2,VA:
                              ResourceMainGear,1:NEXT(50$);

50$           DELAY:
                              MX( TRIA(varTriaMin*(10 *  * ((varSlopeMainGear *
LOG(attSNy11) )+log(varMainGearLN1))),(10 *  * ((varSlopeMainGear *
LOG(attSNy11) )+log(varMainGearLN1))),varTriaMax*(10 *  * ((varSlopeMainGear *
```

```
LOG(attSNy11) )+log(varMainGearLN1)))) ,
TRIA(varTriaMin*varMainGearLN100,varMainGearLN100,varTriaMax*varMainGearLN100))
,,
                              VA:NEXT(93$);

93$             ASSIGN:       y11_Process.WaitTime=y11_Process.WaitTime +
Diff.WaitTime;
57$             TALLY:        y11_Process.WaitTimePerEntity,Diff.WaitTime,1;
59$             TALLY:        y11_Process.TotalTimePerEntity,Diff.StartTime,1;
83$             ASSIGN:       y11_Process.VATime=y11_Process.VATime +
Diff.VATime;
84$             TALLY:        y11_Process.VATimePerEntity,Diff.VATime,1;
49$             RELEASE:      ResourceMainGear,1;
98$             STACK,        1:Destroy:NEXT(97$);

97$             ASSIGN:       y11_Process.NumberOut=y11_Process.NumberOut + 1:
                              y11_Process.WIP=y11_Process.WIP-1:NEXT(8$);


;
;
;     Model statements for module:  BasicProcess.Assign 16 (Show MainGear y11
Time)
;
8$              ASSIGN:       varProcMainGear=
                              MX( TRIA(varTriaMin*(10 *  * ((varSlopeMainGear *
LOG(attSNy11) )+log(varMainGearLN1))),(10 *  * ((varSlopeMainGear *
LOG(attSNy11) )+log(varMainGearLN1))),varTriaMax*(10 *  * ((varSlopeMainGear *
LOG(attSNy11) )+log(varMainGearLN1)))) ,
TRIA(varTriaMin*varMainGearLN100,varMainGearLN100,varTriaMax*varMainGearLN100))
                              :NEXT(36$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 12 (Write y11)
;
36$             WRITE,        ASLC_A1_Data,RECORDSET(Recordset_y11output):
                              varProcMainGear:NEXT(30$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 6 (Write f11)
;
30$             WRITE,        ASLC_A1_Data,RECORDSET(Recordset_f11):
                              TNOW:NEXT(1$);


;
;
;     Model statements for module:  BasicProcess.Dispose 1 (Dispose 1)
;
1$              ASSIGN:       Dispose 1.NumberOut=Dispose 1.NumberOut + 1;
100$            DISPOSE:      Yes;


;
```

```
;
;     Model statements for module:  BasicProcess.Create 6 (S31_Sec41 Order)
;

101$          CREATE,
1,DaysToBaseTime(0.0),Sec41:DaysToBaseTime(1),1:NEXT(102$);

102$          ASSIGN:        S31_Sec41 Order.NumberOut=S31_Sec41
Order.NumberOut + 1:NEXT(4$);


;
;
;     Model statements for module:  BasicProcess.Assign 8 (Assign CurveRate
Sec41)
;
4$            ASSIGN:        varSlopeSec41Fab=LOG(89/100)/LOG(2):
                             varSec41FabLN1=10 *  * (LOG(varSec41FabLN100) -
varSlopeSec41Fab*LOG(100)):NEXT(14$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 2 (Read Sec41
Order Schedule)
;
14$           READ,          ASLC_A1_Data,RECORDSET(Recordset_s31):
                             Sec41Time:NEXT(15$);


;
;
;     Model statements for module:  BasicProcess.Separate 2 (Separate Sec41)
;
15$           DUPLICATE,     100 - 50:
                             1,107$,50:NEXT(106$);

106$          ASSIGN:        Separate Sec41.NumberOut Orig=Separate
Sec41.NumberOut Orig + 1:NEXT(16$);

107$          ASSIGN:        Separate Sec41.NumberOut Dup=Separate
Sec41.NumberOut Dup + 1:NEXT(14$);


;
;
;     Model statements for module:  AdvancedProcess.Delay 2 (Delay Sec41 Order)
;
16$           DELAY:         Sec41Time,,Other:NEXT(35$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 11 (Write s31
out)
;
35$           WRITE,         ASLC_A1_Data,RECORDSET(Recordset_s31output):
                             Sec41Time:NEXT(2$);
```

```
;
;
;     Model statements for module:  BasicProcess.Assign 3 (Assign Sec41 Serial
Number)
;
2$            ASSIGN:        varPartSec41=varPartSec41 + 1:
                             attSNy31=varPartSec41:NEXT(17$);


;
;
;     Model statements for module:  AdvancedTransfer.Route 2 (Route Sec41
Order)
;
17$           ROUTE:         0.,StnSec41Order;


;
;
;     Model statements for module:  AdvancedTransfer.Station 2 (Sec41 Order
Arrival)
;

18$           STATION,       StnSec41Order;
110$          DELAY:         0.0,,VA:NEXT(0$);


;
;
;     Model statements for module:  BasicProcess.Process 5 (y31_Process)
;
0$            ASSIGN:        y31_Process.NumberIn=y31_Process.NumberIn + 1:
                             y31_Process.WIP=y31_Process.WIP+1;
140$          STACK,         1:Save:NEXT(114$);

114$          QUEUE,         y31_Process.Queue;
113$          SEIZE,         2,VA:
                             ResourceSec41Fab,1:NEXT(112$);

112$          DELAY:
                             MX( TRIA(varTriaMin* (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1))), (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1))),varTriaMax* (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1)))) ,  TRIA(varTriaMin*
varSec41FabLN100,varSec41FabLN100,varTriaMax* varSec41FabLN100)  ),,
                             VA:NEXT(155$);

155$          ASSIGN:        y31_Process.WaitTime=y31_Process.WaitTime +
Diff.WaitTime;
119$          TALLY:         y31_Process.WaitTimePerEntity,Diff.WaitTime,1;
121$          TALLY:         y31_Process.TotalTimePerEntity,Diff.StartTime,1;
145$          ASSIGN:        y31_Process.VATime=y31_Process.VATime +
Diff.VATime;
146$          TALLY:         y31_Process.VATimePerEntity,Diff.VATime,1;
111$          RELEASE:       ResourceSec41Fab,1;
160$          STACK,         1:Destroy:NEXT(159$);
```

```
159$          ASSIGN:        y31_Process.NumberOut=y31_Process.NumberOut + 1:
                             y31_Process.WIP=y31_Process.WIP-1:NEXT(29$);


;
;
;     Model statements for module:  BasicProcess.Assign 44 (Show Sec41 y31
Time)
;
29$           ASSIGN:        varProcSec41Fab=
                             MX( TRIA(varTriaMin* (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1))), (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1))),varTriaMax* (10 *  * ((varSlopeSec41Fab *
LOG(attSNy31) )+log(varSec41FabLN1)))) ,  TRIA(varTriaMin*
varSec41FabLN100,varSec41FabLN100,varTriaMax* varSec41FabLN100)  )
                             :NEXT(38$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 14 (Write y31)
;
38$           WRITE,         ASLC_A1_Data,RECORDSET(Recordset_y31output):
                             varProcSec41Fab:NEXT(32$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 8 (Write f31)
;
32$           WRITE,         ASLC_A1_Data,RECORDSET(Recordset_f31):
                             TNOW:NEXT(27$);


;
;
;     Model statements for module:  BasicProcess.Dispose 3 (Dispose 3)
;
27$           ASSIGN:        Dispose 3.NumberOut=Dispose 3.NumberOut + 1;
162$          DISPOSE:       Yes;


;
;
;     Model statements for module:  BasicProcess.Create 7 (S21_ Sec44 Order)
;

163$          CREATE,
1,DaysToBaseTime(0.0),Sec44:DaysToBaseTime(1),1:NEXT(164$);

164$          ASSIGN:        S21_ Sec44 Order.NumberOut=S21_ Sec44
Order.NumberOut + 1:NEXT(3$);


;
;
```

```
;     Model statements for module:  BasicProcess.Assign 6 (Assign CurveRate
Sec44)
;
3$            ASSIGN:        varSlopeSec44Fab=LOG(83.7/100)/LOG(2):
                             varSec44FabLN1=10 *  * (LOG(varSec44FabLN100) -
varSlopeSec44Fab*LOG(100)):NEXT(21$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 5 (Read Sec44
Order Schedule)
;
21$           READ,          ASLC_A1_Data,RECORDSET(Recordset_s21):
                             Sec44Time:NEXT(22$);


;
;
;     Model statements for module:  BasicProcess.Separate 5 (Separate Sec44)
;
22$           DUPLICATE,     100 - 50:
                             1,169$,50:NEXT(168$);

168$          ASSIGN:        Separate Sec44.NumberOut Orig=Separate
Sec44.NumberOut Orig + 1:NEXT(23$);

169$          ASSIGN:        Separate Sec44.NumberOut Dup=Separate
Sec44.NumberOut Dup + 1:NEXT(21$);


;
;
;     Model statements for module:  AdvancedProcess.Delay 5 (Delay Sec44 Order)
;
23$           DELAY:         Sec44Time,,Other:NEXT(34$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 10 (Write s21
out)
;
34$           WRITE,         ASLC_A1_Data,RECORDSET(Recordset_s21output):
                             Sec44Time:NEXT(19$);


;
;
;     Model statements for module:  BasicProcess.Assign 18 (Assign Sec44 Serial
Number)
;
19$           ASSIGN:        varPartSec44=varPartSec44 + 1:
                             attSNy21=varPartSec44:NEXT(24$);


;
;
```

```
;     Model statements for module:  AdvancedTransfer.Route 5 (Route Sec44
Order)
;
24$           ROUTE:          0.,StnSec44Order;


;
;
;     Model statements for module:  AdvancedTransfer.Station 5 (Sec44 Order
Arrival)
;

25$           STATION,        StnSec44Order;
172$          DELAY:          0.0,,VA:NEXT(20$);


;
;
;     Model statements for module:  BasicProcess.Process 17 (y21_Process)
;
20$           ASSIGN:         y21_Process.NumberIn=y21_Process.NumberIn + 1:
                              y21_Process.WIP=y21_Process.WIP+1;
202$          STACK,          1:Save:NEXT(176$);

176$          QUEUE,          y21_Process.Queue;
175$          SEIZE,          2,VA:
                              ResourceSec44Fab,1:NEXT(174$);

174$          DELAY:
                              MX( TRIA(varTriaMin* (10 *  * ((varSlopeSec44Fab *
LOG(attSNy21) )+log(varSec44FabLN1))), (10 *  * ((varSlopeSec44Fab *
LOG(attSNy21) )+log(varSec44FabLN1))),varTriaMax* (10 *  * ((varSlopeSec44Fab *
LOG(attSNy21) )+log(varSec44FabLN1)))) , TRIA(varTriaMin*
varSec44FabLN100,varSec44FabLN100,varTriaMax* varSec44FabLN100)),,
                              VA:NEXT(217$);

217$          ASSIGN:         y21_Process.WaitTime=y21_Process.WaitTime +
Diff.WaitTime;
181$          TALLY:          y21_Process.WaitTimePerEntity,Diff.WaitTime,1;
183$          TALLY:          y21_Process.TotalTimePerEntity,Diff.StartTime,1;
207$          ASSIGN:         y21_Process.VATime=y21_Process.VATime +
Diff.VATime;
208$          TALLY:          y21_Process.VATimePerEntity,Diff.VATime,1;
173$          RELEASE:        ResourceSec44Fab,1;
222$          STACK,          1:Destroy:NEXT(221$);

221$          ASSIGN:         y21_Process.NumberOut=y21_Process.NumberOut + 1:
                              y21_Process.WIP=y21_Process.WIP-1:NEXT(28$);


;
;
;     Model statements for module:  BasicProcess.Assign 43 (Show Sec44 y21
Time)
;
28$           ASSIGN:         varProcSec44Fab=
                              MX( TRIA(varTriaMin* (10 *  * ((varSlopeSec44Fab *
LOG(attSNy21) )+log(varSec44FabLN1))), (10 *  * ((varSlopeSec44Fab *
```

```
LOG(attSNy21) )+log(varSec44FabLN1))),varTriaMax* (10 *  * ((varSlopeSec44Fab *
LOG(attSNy21) )+log(varSec44FabLN1)))) , TRIA(varTriaMin*
varSec44FabLN100,varSec44FabLN100,varTriaMax* varSec44FabLN100))
                              :NEXT(37$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 13 (Write y21)
;
37$           WRITE,          ASLC_A1_Data,RECORDSET(Recordset_y21output):
                              varProcSec44Fab:NEXT(31$);


;
;
;     Model statements for module:  AdvancedProcess.ReadWrite 7 (Write f21)
;
31$           WRITE,          ASLC_A1_Data,RECORDSET(Recordset_f21):
                              TNOW:NEXT(26$);


;
;
;     Model statements for module:  BasicProcess.Dispose 2 (Dispose 2)
;
26$           ASSIGN:         Dispose 2.NumberOut=Dispose 2.NumberOut + 1;
224$          DISPOSE:        Yes;


;
;
;     Model statements for module:  AdvancedProcess.StateSet 1
(StateSetDisruption)
;
```

Appendix D Customized Process Simulation Example Codes

An example code for a customized process in the Arena simulation model.

ProcessOption1 = TRIA (0.85 *(attDelay1 + attDelay2 + varOption1Time) * varOption1, 1.00 * (attDelay1 + attDelay2 + varOption1Time) * varOption1, 1.15 * (attDelay1 + attDelay2 + varOption1Time) * varOption1)

Where (Scenario A):

varDelay1 = 24 hours

varDelay2 = 48 hours

varDealy3 = 120 hours

varOption1 = Discrete distribution of 25% = 1

varOption2 = Discrete distribution of 50% = 1

varOption3 = Discrete distribution of 25% = 1

0.85, 1.00, and 1.15: triangular distribution factors

Appendix E Simulation Code of Final Integration Processes

The process time of a final integration process in a customized decision model:

(MX (TRIA(0.9*(10**((varSlopeFinal * LOG(attSN)) + log(varFinalLN1))),

(10 ** ((varSlopeFinal * LOG(attSN)) + log(varFinalLN1))),

1.1 * (10 ** ((varSlopeFinal * LOG(attSN)) + log(varFinalLN1)))),

TRIA(0.9 * varFinalLN100, varFinalLN100, 1.1 * varFinalLN100)))

* (attDelta1 * attDisruption+1)

Where:

MX = the maximum of (the triangle distribution of the current improvement curve process time, the triangle distribution of the 100th serial number unit process time); assuming the improvement curve process time flattens at the 100th unit.

varSlopeFinal: the improvement curve ratio of the process

attSN: the serial number attribute attached to each component

varFinalLN1: the process time of the first unit on its given improvement curve.

varFinalLN100: the process time of the 100th unit on its given improvement curve.

attDelta1: the magnitude of a disruption

attDisruption: the distribution Boolean attribute, 0 or 1. It represents design change impacts and major customization events. attDelta1 and attDisruption combined represent one type of unexpected change that is injected into the manufacturing system.

Appendix F Null Hypothesis Results

Results for discrete event simulation verification using a statistical linear regression method.

m	N_DES	N_Stat	y11_DES	y11_stat	y21_DES	y21_stat	y31_DES	y31_stat
1	45.0663	42.8324	506.000	506.470	212.000	211.990	230.000	229.910
2	25.2227	24.2083	404.800	404.800	177.444	177.440	204.700	204.700
3	19.9807	21.4628	355.264	355.260	159.904	159.900	191.211	191.210
4	20.2845	20.0685	323.840	323.840	148.521	148.520	182.183	182.180
5	18.1518	17.705	301.392	301.390	140.252	140.250	175.475	175.470
6	19.2431	19.2904	284.212	284.210	133.839	133.840	170.178	170.180
7	17.8192	18.314	270.452	270.450	128.647	128.650	165.824	165.820
8	16.5384	16.5144	259.072	259.070	124.312	124.310	162.143	162.140
9	15.406	15.8563	249.433	249.430	120.609	120.610	158.964	158.960
10	13.0086	13.7235	241.114	241.110	117.391	117.390	156.173	156.170
11	13.306	12.754	233.828	233.830	114.554	114.550	153.690	153.690
12	15.1275	15.4624	227.369	227.370	112.024	112.020	151.458	151.460
13	15.6528	15.49	221.585	221.590	109.745	109.750	149.434	149.430
14	16.2009	15.9297	216.361	216.360	107.677	107.680	147.583	147.580
15	16.0724	15.9059	211.609	211.610	105.787	105.790	145.881	145.880
16	16.2593	16.207	207.258	207.260	104.049	104.050	144.307	144.310
17	17.4346	17.2803	203.252	203.250	102.442	102.440	142.844	142.840
18	17.677	17.7388	199.546	199.550	100.950	100.950	141.478	141.480
19	17.5972	17.5792	196.103	196.100	99.559	99.560	140.197	140.200
20	17.5866	17.6848	192.891	192.890	98.256	98.260	138.994	138.990
21	18.8508	18.8239	189.885	189.890	97.033	97.030	137.858	137.860
22	20.0169	19.8196	187.063	187.060	95.882	95.880	136.784	136.780

23	20.0955	20.3799	184.405	184.400	94.794	94.790	135.766	135.770
24	22.1525	21.9223	181.895	181.900	93.764	93.760	134.798	134.800
25	23.494	23.4472	179.521	179.520	92.786	92.790	133.876	133.880
26	24.7985	24.9039	177.268	177.270	91.857	91.860	132.996	133.000
27	24.9234	25.1549	175.127	175.130	90.971	90.970	132.155	132.150
28	24.9224	25.0446	173.089	173.090	90.126	90.130	131.349	131.350
29	25.6195	25.3109	171.145	171.140	89.318	89.320	130.577	130.580
30	26.603	26.5329	169.287	169.290	88.544	88.540	129.835	129.830
31	24.4125	24.5433	167.509	167.510	87.802	87.800	129.121	129.120
32	22.7027	22.4866	165.806	165.810	87.089	87.090	128.433	128.430
33	20.2325	20.8168	164.172	164.170	86.404	86.400	127.771	127.770
34	19.3228	19.343	162.601	162.600	85.744	85.740	127.131	127.130
35	18.3994	18.4949	161.091	161.090	85.108	85.110	126.513	126.510
36	17.3698	17.2024	159.637	159.640	84.495	84.500	125.915	125.920
37	16.1682	15.9373	158.235	158.230	83.903	83.900	125.336	125.340
38	14.7766	14.9111	156.882	156.880	83.331	83.330	124.776	124.780
39	15.6647	15.6018	155.576	155.580	82.777	82.780	124.232	124.230
40	16.5294	16.4112	154.313	154.310	82.241	82.240	123.704	123.700
41	17.0974	17.1348	153.091	153.090	81.721	81.720	123.192	123.190
42	18.0115	17.8883	151.908	151.910	81.217	81.220	122.694	122.690
43	16.3531	16.591	150.762	150.760	80.728	80.730	122.209	122.210
44	15.1279	15.0696	149.650	149.650	80.253	80.250	121.738	121.740
45	13.6051	13.6753	148.571	148.570	79.791	79.790	121.279	121.280
46	12.4381	12.3265	147.524	147.520	79.342	79.340	120.832	120.830
47	10.8548	11.0609	146.506	146.510	78.906	78.910	120.395	120.400
48	9.51	9.4228	145.516	145.520	78.480	78.480	119.970	119.970
49	8.4585	8.4673	144.554	144.550	78.066	78.070	119.555	119.550

50	7.5766	7.5115	143.617	143.620	77.662	77.660	119.150	119.150
51	6.9717	7.1969	142.704	142.700	77.268	77.270	118.754	118.750
52	6.4918	6.4272	141.815	141.810	76.884	76.880	118.366	118.370
53	4.9702	4.9325	140.948	140.950	76.509	76.510	117.988	117.990
54	3.6462	3.6112	140.102	140.100	76.143	76.140	117.618	117.620
55	2.4933	2.3362	139.277	139.280	75.785	75.790	117.256	117.260
56	2.5197	2.522	138.471	138.470	75.435	75.440	116.901	116.900
57	3.6354	3.5678	137.685	137.680	75.093	75.090	116.553	116.550
58	5.1454	4.9164	136.916	136.920	74.759	74.760	116.213	116.210
59	6.7249	6.6391	136.164	136.160	74.432	74.430	115.880	115.880
60	8.448	8.2607	135.430	135.430	74.111	74.110	115.553	115.550
61	9.9046	9.8654	134.711	134.710	73.797	73.800	115.232	115.230
62	9.5212	9.3964	134.008	134.010	73.490	73.490	114.917	114.920
63	9.1756	9.1322	133.319	133.320	73.189	73.190	114.609	114.610
64	8.6504	8.7559	132.645	132.640	72.893	72.890	114.306	114.310
65	8.4015	8.3665	131.984	131.980	72.604	72.600	114.008	114.010
66	7.9315	8.0673	131.337	131.340	72.320	72.320	113.716	113.720
67	7.6661	7.6678	130.703	130.700	72.041	72.040	113.429	113.430
68	7.3599	7.2612	130.081	130.080	71.768	71.770	113.147	113.150
69	6.9762	6.8238	129.471	129.470	71.499	71.500	112.869	112.870
70	6.748	6.6752	128.873	128.870	71.236	71.240	112.596	112.600
71	6.4414	6.1779	128.286	128.290	70.977	70.980	112.328	112.330
72	5.9378	6.1791	127.709	127.710	70.723	70.720	112.064	112.060
73	5.641	5.777	127.144	127.140	70.473	70.470	111.805	111.800
74	5.2916	5.2834	126.588	126.590	70.227	70.230	111.549	111.550
75	4.9426	5.0259	126.042	126.040	69.985	69.990	111.298	111.300
76	4.8086	4.7854	125.506	125.510	69.748	69.750	111.052	111.050

78	4.1038	4.1896	124.461	124.460	69.284	69.280	110.567	110.570
79	4.0524	3.8653	123.951	123.950	69.058	69.060	110.330	110.330
80	3.6773	3.6015	123.450	123.450	68.835	68.840	110.097	110.100
81	3.32	3.2701	122.958	122.960	68.616	68.620	109.867	109.870
82	3.1578	3.061	122.473	122.470	68.400	68.400	109.641	109.640
83	2.8426	2.6802	121.996	122.000	68.188	68.190	109.418	109.420
84	2.5809	2.543	121.526	121.530	67.979	67.980	109.197	109.200
85	2.3426	2.2266	121.064	121.060	67.772	67.770	108.980	108.980
86	2.2124	2.2514	120.609	120.610	67.569	67.570	108.766	108.770
87	2.1318	2.3161	120.161	120.160	67.369	67.370	108.555	108.560
88	2.1058	2.1589	119.720	119.720	67.172	67.170	108.347	108.350
89	2.1309	2.1275	119.285	119.290	66.977	66.980	108.141	108.140
90	2.1671	2.2691	118.857	118.860	66.785	66.790	107.938	107.940
91	2.0594	2.1768	118.435	118.430	66.596	66.600	107.738	107.740
92	2.0265	2.006	118.019	118.020	66.410	66.410	107.540	107.540
93	2.2033	2.0904	117.609	117.610	66.226	66.230	107.345	107.340
94	2.1173	2.127	117.205	117.200	66.044	66.040	107.152	107.150
95	2.1214	2.1567	116.806	116.810	65.865	65.860	106.962	106.960
96	2.5145	2.4464	116.413	116.410	65.688	65.690	106.773	106.770
97	2.5655	2.6079	116.025	116.030	65.513	65.510	106.588	106.590
98	2.6863	2.843	115.643	115.640	65.341	65.340	106.404	106.400
99	2.0244	2.1038	115.266	115.270	65.171	65.170	106.222	106.220
100	2.0100	2.1237	114.893	114.890	65.003	65.000	106.043	106.040

Appendix G The Second Alternative Hypothesis Results

Alternative Hypothesis 2 results.

m	N_DES Ha2	N_Stat	f11_output	f21_output	f31_output	Fij
1	203.8179	42.83238	14302.82	14214.58	14287.65	14099
2	78.2675	24.20831	14305.53	14227.26	14291.83	14137
3	66.6715	21.46283	14307.90	14241.23	14294.97	14165
4	57.6026	20.06852	14309.04	14251.44	14295.81	14184
5	54.3653	17.70498	14311.62	14257.26	14297.88	14197
6	50.6199	19.29035	14314.78	14264.16	14301.42	14209
7	43.9772	18.31400	14315.94	14271.97	14306.57	14221
8	40.6553	16.51436	14321.55	14280.89	14312.78	14233
9	38.8604	15.85632	14328.15	14289.29	14319.75	14245
10	36.4799	13.72355	14335.39	14298.91	14327.12	14257
11	34.2655	12.75396	14343.01	14308.74	14335.54	14269
12	32.8801	15.46243	14351.61	14318.73	14344.34	14281
13	30.9853	15.48999	14358.01	14327.03	14351.21	14291
14	29.9013	15.92965	14365.20	14335.30	14358.11	14301
15	28.3679	15.90589	14372.14	14343.77	14365.92	14311
16	27.3490	16.20697	14379.56	14352.21	14373.50	14321
17	26.1471	17.28026	14385.21	14359.06	14379.29	14329
18	25.1771	17.73884	14390.98	14365.80	14385.43	14337
19	23.9230	17.57921	14396.64	14372.72	14391.52	14345
20	23.4102	17.68482	14402.92	14379.51	14397.83	14353
21	21.9462	18.82392	14408.86	14386.92	14403.89	14361
22	21.6586	19.81958	14415.43	14393.77	14410.55	14369
23	20.7526	20.37991	14421.79	14401.03	14417.30	14377
24	19.9251	21.92227	14427.97	14408.05	14423.70	14385
25	19.3824	23.44722	14434.71	14415.33	14430.55	14393
26	19.3039	24.90394	14441.70	14422.40	14437.25	14401
27	18.2439	25.15488	14448.06	14429.82	14444.13	14409
28	17.6700	25.04457	14453.81	14436.14	14449.93	14416
29	17.0878	25.31088	14458.62	14441.53	14455.06	14422
30	16.6734	26.53293	14463.46	14446.79	14459.78	14428
31	16.1543	24.54328	14468.41	14452.26	14464.58	14434
32	15.5097	22.48664	14473.28	14457.77	14469.73	14440
33	14.6200	20.81676	14478.01	14463.39	14474.94	14446
34	14.3772	19.34303	14483.29	14468.91	14479.94	14452
35	14.3854	18.49487	14488.50	14474.12	14485.14	14458
36	13.7224	17.20244	14493.36	14479.64	14490.16	14464
37	13.4680	15.93735	14498.59	14485.12	14495.52	14470
38	12.7196	14.91107	14503.53	14490.81	14500.75	14476

m	N_DES Ha2	N_Stat	f11_output	f21_output	f31_output	Fij
39	12.5874	15.60179	14508.75	14496.17	14505.98	14482
40	12.1742	16.41119	14513.98	14501.80	14511.41	14488
41	11.6886	17.13482	14519.32	14507.63	14516.52	14494
42	11.5273	17.8883	14524.45	14512.92	14521.64	14500
43	11.0129	16.59099	14529.58	14518.56	14527.09	14506
44	10.7683	15.06965	14535.02	14524.25	14532.65	14512
45	10.3168	13.67535	14540.21	14529.89	14538.10	14518
46	10.1065	12.32645	14545.62	14535.51	14543.48	14524
47	9.9728	11.06087	14550.99	14541.02	14548.77	14530
48	9.6444	9.422752	14554.44	14544.80	14552.24	14534
49	9.1983	8.467279	14557.67	14548.47	14555.60	14538
50	8.7939	7.511465	14561.03	14552.24	14559.09	14542
51	8.6995	7.196947	14564.43	14555.73	14562.47	14546
52	8.5416	6.427215	14567.95	14559.41	14566.10	14550
53	8.1070	4.932479	14571.36	14563.25	14569.55	14554
54	7.8956	3.611158	14574.92	14567.03	14573.18	14558
55	7.6293	2.336237	14578.20	14570.57	14576.65	14562
56	7.3756	2.521985	14581.76	14574.39	14580.07	14566
57	7.0590	3.567827	14585.12	14578.06	14583.64	14570
58	7.1067	4.916361	14588.84	14581.73	14587.17	14574
59	6.6598	6.639136	14592.21	14585.55	14590.66	14578
60	6.4471	8.260711	14595.79	14589.35	14594.22	14582
61	6.2243	9.865378	14599.27	14593.04	14597.87	14586
62	5.9651	9.396388	14602.83	14596.86	14601.58	14590
63	6.0473	9.132166	14606.53	14600.48	14605.03	14594
64	5.5768	8.755876	14609.92	14604.35	14608.65	14598
65	5.4942	8.366531	14613.58	14608.09	14612.21	14602
66	5.2450	8.067348	14617.03	14611.78	14615.93	14606
67	5.0227	7.667783	14620.65	14615.63	14619.65	14610
68	4.7747	7.261238	14624.16	14619.39	14623.12	14614
69	4.6157	6.82376	14627.87	14623.26	14626.77	14618
70	4.5669	6.675229	14631.43	14626.86	14630.42	14622
71	4.4097	6.177904	14635.19	14630.78	14634.05	14626
72	3.9823	6.179117	14638.60	14634.61	14637.77	14630
73	3.8572	5.776978	14642.16	14638.30	14641.37	14634
74	3.6583	5.283398	14645.86	14642.21	14645.13	14638
75	3.5083	5.025909	14649.45	14645.94	14648.77	14642
76	3.4391	4.794004	14653.17	14649.73	14652.37	14646
77	3.1655	4.285303	14656.81	14653.64	14656.21	14650
78	3.0768	4.189551	14660.48	14657.40	14659.92	14654
79	2.8952	3.865347	14663.14	14660.24	14662.50	14657
80	2.7528	3.601482	14665.82	14663.07	14665.25	14660
81	2.6908	3.270065	14668.48	14665.79	14667.86	14663

m	N_DES Ha2	N_Stat	f11_output	f21_output	f31_output	Fij
82	2.4039	3.061024	14671.15	14668.74	14670.62	14666
83	2.3421	2.680218	14673.89	14671.55	14673.39	14669
84	2.0264	2.543045	14676.49	14674.46	14675.91	14672
85	1.9479	2.226555	14679.18	14677.23	14678.75	14675
86	1.7991	2.251352	14681.86	14680.06	14681.42	14678
87	1.6935	2.316056	14684.56	14682.87	14684.23	14681
88	1.5616	2.158869	14687.29	14685.73	14686.98	14684
89	1.3467	2.12745	14690.03	14688.68	14689.68	14687
90	1.2520	2.269123	14692.69	14691.44	14692.40	14690
91	1.1304	2.176841	14695.38	14694.25	14695.15	14693
92	0.9746	2.006044	14698.09	14697.11	14697.94	14696
93	0.8756	2.090377	14700.81	14699.94	14700.61	14699
94	0.6662	2.127026	14703.49	14702.83	14703.39	14702
95	0.5769	2.156749	14706.30	14705.72	14706.20	14705
96	0.5215	2.446432	14709.04	14708.52	14708.94	14708
97	0.3459	2.60793	14711.79	14711.44	14711.70	14711
98	0.2321	2.843014	14714.51	14714.27	14714.46	14714
99	0.0963	2.103801	14717.24	14717.15	14717.25	14717
100	0.1064	2.123715	14720.09	14720.02	14719.99	14720

Appendix H The Third Alternative Hypothesis Results

Alternative Hypothesis 3 results.

m	N_DES Ha3	N_Stat	f11_output	f21_output	f31_output	Fij
1	0.0000	42.8324	14099.00	14099.00	14099.00	14099
2	0.0000	24.2083	14104.00	14104.00	14104.00	14104
3	0.0000	21.4628	14109.00	14109.00	14109.00	14109
4	0.0000	20.0685	14114.00	14114.00	14114.00	14114
5	0.0000	17.7050	14119.00	14119.00	14119.00	14119
6	0.0000	19.2904	14124.00	14124.00	14124.00	14124
7	0.0000	18.3140	14129.00	14129.00	14129.00	14129
8	0.0000	16.5144	14134.00	14134.00	14134.00	14134
9	0.0000	15.8563	14139.00	14139.00	14139.00	14139
10	0.0000	13.7235	14144.00	14144.00	14144.00	14144
11	0.0000	12.7540	14149.00	14149.00	14149.00	14149
12	0.0000	15.4624	14154.00	14154.00	14154.00	14154
13	0.0000	15.4900	14159.00	14159.00	14159.00	14159
14	0.0000	15.9297	14164.00	14164.00	14164.00	14164
15	0.0000	15.9059	14169.00	14169.00	14169.00	14169
16	0.0000	16.2070	14174.00	14174.00	14174.00	14174
17	0.0000	17.2803	14179.00	14179.00	14179.00	14179
18	0.0000	17.7388	14184.00	14184.00	14184.00	14184
19	0.0000	17.5792	14189.00	14189.00	14189.00	14189
20	0.0000	17.6848	14194.00	14194.00	14194.00	14194
21	0.0000	18.8239	14199.00	14199.00	14199.00	14199
22	0.0000	19.8196	14204.00	14204.00	14204.00	14204
23	0.0000	20.3799	14209.00	14209.00	14209.00	14209
24	0.0000	21.9223	14214.00	14214.00	14214.00	14214
25	0.0000	23.4472	14219.00	14219.00	14219.00	14219
26	0.0000	24.9039	14224.00	14224.00	14224.00	14224
27	0.0000	25.1549	14229.00	14229.00	14229.00	14229
28	0.0000	25.0446	14234.00	14234.00	14234.00	14234
29	0.0000	25.3109	14239.00	14239.00	14239.00	14239
30	0.0000	26.5329	14244.00	14244.00	14244.00	14244
31	0.0000	24.5433	14249.00	14249.00	14249.00	14249
32	0.0000	22.4866	14254.00	14254.00	14254.00	14254
33	0.0000	20.8168	14259.00	14259.00	14259.00	14259
34	0.0000	19.3430	14264.00	14264.00	14264.00	14264
35	0.0000	18.4949	14269.00	14269.00	14269.00	14269
36	0.0000	17.2024	14274.00	14274.00	14274.00	14274
37	0.0000	15.9373	14279.00	14279.00	14279.00	14279
38	0.0000	14.9111	14284.00	14284.00	14284.00	14284

m	N_DES Ha3	N_Stat	f11_output	f21_output	f31_output	Fij
39	0.0000	15.6018	14289.00	14289.00	14289.00	14289
40	0.0000	16.4112	14294.00	14294.00	14294.00	14294
41	0.0000	17.1348	14299.00	14299.00	14299.00	14299
42	0.0000	17.8883	14304.00	14304.00	14304.00	14304
43	0.0000	16.5910	14309.00	14309.00	14309.00	14309
44	0.0000	15.0696	14314.00	14314.00	14314.00	14314
45	0.0000	13.6753	14319.00	14319.00	14319.00	14319
46	0.0000	12.3265	14324.00	14324.00	14324.00	14324
47	0.0000	11.0609	14329.00	14329.00	14329.00	14329
48	0.0000	9.4228	14334.00	14334.00	14334.00	14334
49	0.0000	8.4673	14339.00	14339.00	14339.00	14339
50	0.0000	7.5115	14344.00	14344.00	14344.00	14344
51	0.0000	7.1969	14349.00	14349.00	14349.00	14349
52	0.0000	6.4272	14354.00	14354.00	14354.00	14354
53	0.0000	4.9325	14359.00	14359.00	14359.00	14359
54	0.0000	3.6112	14364.00	14364.00	14364.00	14364
55	0.0000	2.3362	14369.00	14369.00	14369.00	14369
56	0.0000	2.5220	14374.00	14374.00	14374.00	14374
57	0.0000	3.5678	14379.00	14379.00	14379.00	14379
58	0.0000	4.9164	14384.00	14384.00	14384.00	14384
59	0.0000	6.6391	14389.00	14389.00	14389.00	14389
60	0.0000	8.2607	14394.00	14394.00	14394.00	14394
61	0.0000	9.8654	14399.00	14399.00	14399.00	14399
62	0.0000	9.3964	14404.00	14404.00	14404.00	14404
63	0.0000	9.1322	14409.00	14409.00	14409.00	14409
64	0.0000	8.7559	14414.00	14414.00	14414.00	14414
65	0.0000	8.3665	14419.00	14419.00	14419.00	14419
66	0.0000	8.0673	14424.00	14424.00	14424.00	14424
67	0.0000	7.6678	14429.00	14429.00	14429.00	14429
68	0.0000	7.2612	14434.00	14434.00	14434.00	14434
69	0.0000	6.8238	14439.00	14439.00	14439.00	14439
70	0.0000	6.6752	14444.00	14444.00	14444.00	14444
71	0.0000	6.1779	14449.00	14449.00	14449.00	14449
72	0.0000	6.1791	14454.00	14454.00	14454.00	14454
73	0.0000	5.7770	14459.00	14459.00	14459.00	14459
74	0.0000	5.2834	14464.00	14464.00	14464.00	14464
75	0.0000	5.0259	14469.00	14469.00	14469.00	14469
76	0.0000	4.7940	14474.00	14474.00	14474.00	14474
77	0.0000	4.2853	14479.00	14479.00	14479.00	14479
78	0.0000	4.1896	14484.00	14484.00	14484.00	14484
79	0.0000	3.8653	14489.00	14489.00	14489.00	14489
80	0.0000	3.6015	14494.00	14494.00	14494.00	14494
81	0.0000	3.2701	14499.00	14499.00	14499.00	14499

m	N_DES Ha3	N_Stat	f11_output	f21_output	f31_output	Fij
82	0.0000	3.0610	14504.00	14504.00	14504.00	14504
83	0.0000	2.6802	14509.00	14509.00	14509.00	14509
84	0.0000	2.5430	14514.00	14514.00	14514.00	14514
85	0.0000	2.2266	14519.00	14519.00	14519.00	14519
86	0.0000	2.2514	14524.00	14524.00	14524.00	14524
87	0.0000	2.3161	14529.00	14529.00	14529.00	14529
88	0.0000	2.1589	14534.00	14534.00	14534.00	14534
89	0.0000	2.1275	14539.00	14539.00	14539.00	14539
90	0.0000	2.2691	14544.00	14544.00	14544.00	14544
91	0.0000	2.1768	14549.00	14549.00	14549.00	14549
92	0.0000	2.0060	14554.00	14554.00	14554.00	14554
93	0.0000	2.0904	14559.00	14559.00	14559.00	14559
94	0.0000	2.1270	14564.00	14564.00	14564.00	14564
95	0.0000	2.1567	14569.00	14569.00	14569.00	14569
96	0.0000	2.4464	14574.00	14574.00	14574.00	14574
97	0.0000	2.6079	14579.00	14579.00	14579.00	14579
98	0.0000	2.8430	14584.00	14584.00	14584.00	14584
99	0.0000	2.1038	14589.00	14589.00	14589.00	14589
100	0.0000	2.1237	14594.00	14594.00	14594.00	14594

Appendix I The Forth Alternative Hypothesis Results

Alternative Hypothesis 4 results.

m	N_DES Ha4	N_Stat	f11_output	f21_output	f31_output	Fij
1	10.2343	42.8324	14097.26	14107.50	14103.29	14099
2	4.7460	24.2083	14138.43	14143.17	14142.87	14137
3	5.7337	21.4628	14165.26	14170.99	14170.26	14165
4	5.7961	20.0685	14183.99	14189.78	14188.76	14184
5	5.2348	17.7050	14196.99	14202.22	14200.90	14197
6	4.8702	19.2904	14208.97	14213.84	14212.23	14209
7	4.6946	18.3140	14221.00	14225.69	14224.43	14221
8	4.4473	16.5144	14232.82	14237.27	14236.53	14233
9	3.9194	15.8563	14245.23	14249.14	14247.80	14245
10	3.9471	13.7235	14257.17	14261.12	14259.81	14257
11	3.8135	12.7540	14268.79	14272.60	14271.71	14269
12	3.1636	15.4624	14281.27	14284.44	14283.80	14281
13	2.9253	15.4900	14291.43	14294.36	14293.59	14291
14	3.2628	15.9297	14300.93	14304.20	14303.33	14301
15	3.2207	15.9059	14310.89	14314.11	14313.13	14311
16	2.6056	16.2070	14321.43	14324.04	14323.04	14321
17	2.9549	17.2803	14329.09	14332.04	14331.22	14329
18	2.8652	17.7388	14337.11	14339.98	14339.43	14337
19	2.7521	17.5792	14344.99	14347.74	14346.85	14345
20	2.9050	17.6848	14352.76	14355.66	14354.84	14353
21	2.7813	18.8239	14360.94	14363.72	14362.97	14361
22	2.3908	19.8196	14369.10	14371.49	14370.90	14369
23	2.8501	20.3799	14376.84	14379.69	14378.99	14377
24	2.1343	21.9223	14385.10	14387.24	14386.73	14385
25	2.0519	23.4472	14393.07	14395.12	14394.75	14393
26	2.7505	24.9039	14400.79	14403.54	14402.71	14401
27	2.0970	25.1549	14409.11	14411.21	14410.44	14409
28	2.0077	25.0446	14416.00	14418.01	14417.44	14416
29	2.1275	25.3109	14421.94	14424.06	14423.64	14422
30	1.7983	26.5329	14428.13	14429.92	14429.49	14428
31	1.8729	24.5433	14434.03	14435.90	14435.51	14434
32	1.8470	22.4866	14439.97	14441.82	14441.41	14440
33	1.9533	20.8168	14446.07	14448.02	14447.39	14446
34	1.7967	19.3430	14451.98	14453.78	14453.31	14452
35	1.8280	18.4949	14457.88	14459.71	14459.30	14458
36	1.5503	17.2024	14463.92	14465.47	14465.47	14464
37	1.6012	15.9373	14469.99	14471.60	14471.25	14470
38	1.7600	14.9111	14476.04	14477.80	14477.28	14476

m	N_DES Ha4	N_Stat	f11_output	f21_output	f31_output	Fij
39	1.3639	15.6018	14481.92	14483.29	14483.15	14482
40	1.7084	16.4112	14488.04	14489.75	14489.28	14488
41	1.2353	17.1348	14494.26	14495.49	14495.12	14494
42	1.4992	17.8883	14499.95	14501.45	14501.05	14500
43	1.5591	16.5910	14505.96	14507.52	14507.08	14506
44	1.2657	15.0696	14512.00	14513.26	14512.98	14512
45	1.5021	13.6753	14517.97	14519.48	14518.81	14518
46	1.4503	12.3265	14523.97	14525.42	14524.93	14524
47	1.0205	11.0609	14530.21	14531.23	14530.96	14530
48	1.3263	9.4228	14533.99	14535.32	14534.96	14534
49	1.0686	8.4673	14538.00	14539.07	14538.96	14538
50	1.1789	7.5115	14542.02	14543.20	14542.84	14542
51	1.0487	7.1969	14545.98	14547.03	14546.91	14546
52	0.9716	6.4272	14549.98	14550.95	14550.87	14550
53	1.0336	4.9325	14554.07	14555.11	14554.78	14554
54	1.0390	3.6112	14557.99	14559.02	14558.79	14558
55	0.9602	2.3362	14562.01	14562.97	14562.75	14562
56	0.7809	2.5220	14566.06	14566.84	14566.77	14566
57	0.8694	3.5678	14570.00	14570.87	14570.75	14570
58	1.0121	4.9164	14573.85	14574.86	14574.62	14574
59	0.8421	6.6391	14578.10	14578.94	14578.68	14578
60	1.0017	8.2607	14581.90	14582.90	14582.67	14582
61	0.8223	9.8654	14585.98	14586.80	14586.70	14586
62	0.8190	9.3964	14589.95	14590.77	14590.60	14590
63	0.7405	9.1322	14593.94	14594.68	14594.61	14594
64	0.4676	8.7559	14598.12	14598.58	14598.56	14598
65	0.9507	8.3665	14601.92	14602.87	14602.47	14602
66	0.6122	8.0673	14606.05	14606.66	14606.56	14606
67	0.4789	7.6678	14610.11	14610.56	14610.59	14610
68	0.4697	7.2612	14614.16	14614.63	14614.38	14614
69	0.5459	6.8238	14618.05	14618.60	14618.56	14618
70	0.4298	6.6752	14622.13	14622.55	14622.56	14622
71	0.6905	6.1779	14626.00	14626.69	14626.35	14626
72	0.5690	6.1791	14629.94	14630.51	14630.43	14630
73	0.6399	5.7770	14633.92	14634.56	14634.33	14634
74	0.4159	5.2834	14637.94	14638.35	14638.36	14638
75	0.5592	5.0259	14641.91	14642.47	14642.37	14642
76	0.5112	4.7940	14646.06	14646.58	14646.18	14646
77	0.4568	4.2853	14650.00	14650.41	14650.45	14650
78	0.4718	4.1896	14653.98	14654.45	14654.27	14654
79	0.3635	3.8653	14657.01	14657.38	14657.23	14657
80	0.4161	3.6015	14659.96	14660.33	14660.38	14660
81	0.3229	3.2701	14663.00	14663.32	14663.22	14663

m	N_DES Ha4	N_Stat	f11_output	f21_output	f31_output	Fij
82	0.3185	3.0610	14666.05	14666.37	14666.24	14666
83	0.4019	2.6802	14668.91	14669.32	14669.23	14669
84	0.2997	2.5430	14672.01	14672.31	14672.22	14672
85	0.2734	2.2266	14674.94	14675.22	14675.21	14675
86	0.2431	2.2514	14677.99	14678.23	14678.12	14678
87	0.3064	2.3161	14680.88	14681.19	14681.17	14681
88	0.1880	2.1589	14684.00	14684.17	14684.18	14684
89	0.1881	2.1275	14687.05	14687.24	14687.07	14687
90	0.0668	2.2691	14690.10	14690.15	14690.09	14690
91	0.1027	2.1768	14692.99	14693.09	14693.08	14693
92	0.0847	2.0060	14696.00	14696.08	14696.06	14696
93	0.2345	2.0904	14698.89	14699.13	14699.12	14699
94	0.1264	2.1270	14702.00	14702.12	14702.05	14702
95	0.0665	2.1567	14705.01	14705.07	14705.08	14705
96	0.1384	2.4464	14708.01	14708.15	14708.04	14708
97	0.1666	2.6079	14710.94	14711.11	14711.04	14711
98	0.0930	2.8430	14713.93	14714.02	14714.02	14714
99	0.0222	2.1038	14716.95	14716.96	14716.97	14717
100	0.0617	2.1237	14720.02	14719.96	14720.00	14720

Appendix J The Second Case Study Simulation Codes

Case study two DES codes:

```
;
;
;   Model statements for module:  BasicProcess.Create 1 (Nose Gear Order)
;
189$        CREATE,       1,DaysToBaseTime(0.0),NoseGear:DaysToBaseTime(1),1:NEXT(190$);

190$        ASSIGN:       Nose Gear Order.NumberOut=Nose Gear Order.NumberOut + 1:NEXT(17$);


;
;
;   Model statements for module:  BasicProcess.Assign 4 (Assign CurveRate NostGear)
;
17$         ASSIGN:       varSlopeNoseGearForge=LOG(varCurveRate(80)/100)/LOG(2):
                          varNoseGearForgeLN1=10        *             *        (LOG(varNoseGearForgeLN100)       -
varSlopeNoseGearForge*LOG(100)):
                          varSlopeNoseGearMach=LOG(varCurveRate(80)/100)/LOG(2):
                          varNoseGearMachLN1=10         *             *        (LOG(varNoseGearMachLN100)        -
varSlopeNoseGearMach*LOG(100)):
                          varSlopeNoseGearBuildUp=LOG(varCurveRate(80)/100)/LOG(2):
                          varNoseGearBuildUpLN1=10      *             *        (LOG(varNoseGearBuildUpLN100)     -
varSlopeNoseGearBuildUp*LOG(100)):
                          varSlopePreIntegration=LOG(varCurveRate(88)/100)/LOG(2):
                          varPreIntegrationLN1=10       *             *        (LOG(varPreIntegrationLN100)      -
varSlopePreIntegration*LOG(100)):
                          varPreInteSec49LN1=10         *             *        (LOG(varPreInteSec49LN100)        -
varSlopePreIntegration*LOG(100)):NEXT(47$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 4 (Read Nose Gear Order Schedule)
;
47$         READ,         MDaySchedule787,RECORDSET(RecordsetNLG):
                          NoseGearTime:NEXT(48$);


;
;
;   Model statements for module:  BasicProcess.Separate 4 (Separate Nose Gear)
;
48$         DUPLICATE,    100 - 50:
                          1,195$,50:NEXT(194$);

194$        ASSIGN:       Separate Nose Gear.NumberOut Orig=Separate Nose Gear.NumberOut Orig +
1:NEXT(49$);

195$        ASSIGN:       Separate Nose Gear.NumberOut Dup=Separate Nose Gear.NumberOut Dup +
1:NEXT(47$);
```

```
;
;
;   Model statements for module:  AdvancedProcess.Delay 4 (Delay Nose Gear Order)
;
49$         DELAY:        NoseGearTime,,Other:NEXT(13$);


;
;
;   Model statements for module:  BasicProcess.Assign 1 (Assign Nose Gear Serial Number)
;
13$         ASSIGN:       varPartNoseGear=varPartNoseGear + 1:
                          attSN=varPartNoseGear:NEXT(50$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 4 (Route Nose Gear Order)
;
50$         ROUTE:        0.,StnNoseGearOrder;


;
;
;   Model statements for module:  BasicProcess.Create 2 (Sec43 Order)
;
196$        CREATE,       1,DaysToBaseTime(0.0),Sec43:DaysToBaseTime(1),1:NEXT(197$);

197$        ASSIGN:       Sec43 Order.NumberOut=Sec43 Order.NumberOut + 1:NEXT(52$);


;
;
;   Model statements for module:  BasicProcess.Assign 17 (Assign CurveRate Sec43)
;
52$         ASSIGN:       varSlopeSec43AsmMinor=LOG(varCurveRate(82)/100)/LOG(2):
                          varSec43AsmMinorLN1=10        *             *        (LOG(varSec43AsmMinorLN100)       -
varSlopeSec43AsmMinor*LOG(100)):
                          varSlopeSec43AsmMajor=LOG(varCurveRate(85)/100)/LOG(2):
                          varSec43AsmMajorLN1=10        *             *        (LOG(varSec43AsmMajorLN100)       -
varSlopeSec43AsmMajor*LOG(100)):
                          varSlopeSec43Fab=LOG(varCurveRate(84)/100)/LOG(2):
                          varSec43FabLN1=10             *             *             (LOG(varSec43FabLN100)       -
varSlopeSec43Fab*LOG(100)):NEXT(40$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 3 (Read Sec43 Order Schedule)
;
40$         READ,         MDaySchedule787,RECORDSET(RecordsetSec43):
                          Sec43Time:NEXT(41$);
```

```
;
;
;   Model statements for module:  BasicProcess.Separate 3 (Separate Sec43)
;
41$         DUPLICATE,    100 - 50:
                  1,202$,50:NEXT(201$);

201$        ASSIGN:       Separate Sec43.NumberOut Orig=Separate Sec43.NumberOut Orig + 1:NEXT(42$);

202$        ASSIGN:       Separate Sec43.NumberOut Dup=Separate Sec43.NumberOut Dup + 1:NEXT(40$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 3 (Delay Sec43 Order)
;
42$         DELAY:        Sec43Time,,Other:NEXT(14$);


;
;
;   Model statements for module:  BasicProcess.Assign 2 (Assign Sec43 Serial Number)
;
14$         ASSIGN:       varPartSec43=varPartSec43 + 1:
                  attSN=varPartSec43:NEXT(43$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 3 (Route Sec43 Order)
;
43$         ROUTE:        0.,StnSec43Order;


;
;
;   Model statements for module:  BasicProcess.Create 5 (Main Gear Order)
;

203$        CREATE,       1,DaysToBaseTime(0.0),MainGear:DaysToBaseTime(1),1:NEXT(204$);

204$        ASSIGN:       Main Gear Order.NumberOut=Main Gear Order.NumberOut + 1:NEXT(25$);


;
;
;   Model statements for module:  BasicProcess.Assign 15 (Assign CurveRate MainGear)
;
25$         ASSIGN:       varSlopeMainGear=LOG(varCurveRate(80)/100)/LOG(2):
                  varMainGearLN1=10 * * (LOG(varMainGearLN100) - varSlopeMainGear*LOG(100)):
                  varSlopeMainGearMach=LOG(varCurveRate(80)/100)/LOG(2):
                  varMainGearMachLN1=10       *       *       (LOG(varMainGearMachLN100)       -
varSlopeMainGearMach*LOG(100)):
                  varSlopeMainGearBuildUp=LOG(varCurveRate(80)/100)/LOG(2):
                  varMainGearBuildUpLN1=10       *       *       (LOG(varMainGearBuildUpLN100)       -
varSlopeMainGearBuildUp*LOG(100))
                  :NEXT(28$);
```

```
;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 1 (Read Main Gear Order Schedule)
;
28$         READ,       MDaySchedule787,RECORDSET(RecordsetMLG):
                    MainGearTime:NEXT(29$);


;
;
;   Model statements for module:  BasicProcess.Separate 1 (Separate Main Gear)
;
29$         DUPLICATE,    100 - 50:
                    1,209$,50:NEXT(208$);

208$            ASSIGN:         Separate Main Gear.NumberOut Orig=Separate Main Gear.NumberOut Orig +
1:NEXT(30$);

209$            ASSIGN:         Separate Main Gear.NumberOut Dup=Separate Main Gear.NumberOut Dup +
1:NEXT(28$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 1 (Delay MainGear Order)
;
30$         DELAY:      MainGearTime,,Other:NEXT(24$);


;
;
;   Model statements for module:  BasicProcess.Assign 14 (Assign Main Gear Serial Number)
;
24$         ASSIGN:     varPartMainGear=varPartMainGear + 1:
                    attSN=varPartMainGear:NEXT(31$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 1 (Route MainGear Order)
;
31$         ROUTE:      0.,StnMainGearOrder;


;
;
;   Model statements for module:  AdvancedTransfer.Station 1 (Main Gear Order Arrival)
;
32$         STATION,    StnMainGearOrder;
212$        DELAY:      0.0,,VA:NEXT(167$);


;
;
```

```
;   Model statements for module:  BasicProcess.Assign 41 (Assign Decision 1)
;
167$        ASSIGN:      varDecision1=DISC(0.25, 1, 1.0, 0):
                         attDelay1=varDelay1*varDecision1:NEXT(26$);


;
;
;   Model statements for module:  BasicProcess.Process 13 (Main Landing Gear Forging)
;
26$         ASSIGN:      Main Landing Gear Forging.NumberIn=Main Landing Gear Forging.NumberIn + 1:
                         Main Landing Gear Forging.WIP=Main Landing Gear Forging.WIP+1;
242$        STACK,       1:Save:NEXT(216$);

216$        QUEUE,       Main Landing Gear Forging.Queue;
215$        SEIZE,       2,VA:
                         ResourceMainGear,1:NEXT(214$);

214$        DELAY:
                         MX( TRIA(0.85*(10 *  * ((varSlopeMainGear * LOG(attSN) )+log(varMainGearLN1))),(10 *  *
((varSlopeMainGear * LOG(attSN) )+log(varMainGearLN1))),1.15*(10 *  * ((varSlopeMainGear * LOG(attSN)
)+log(varMainGearLN1)))) , TRIA(0.85*varMainGearLN100,varMainGearLN100,1.15*varMainGearLN100)),,
                         VA:NEXT(257$);

257$          ASSIGN:        Main Landing Gear Forging.WaitTime=Main Landing Gear Forging.WaitTime +
Diff.WaitTime;
221$        TALLY:       Main Landing Gear Forging.WaitTimePerEntity,Diff.WaitTime,1;
223$        TALLY:       Main Landing Gear Forging.TotalTimePerEntity,Diff.StartTime,1;
247$          ASSIGN:        Main Landing Gear Forging.VATime=Main Landing Gear Forging.VATime +
Diff.VATime;
248$        TALLY:       Main Landing Gear Forging.VATimePerEntity,Diff.VATime,1;
213$        RELEASE:     ResourceMainGear,1;
262$        STACK,       1:Destroy:NEXT(261$);

261$        ASSIGN:      Main Landing Gear Forging.NumberOut=Main Landing Gear Forging.NumberOut + 1:
                         Main Landing Gear Forging.WIP=Main Landing Gear Forging.WIP-1:NEXT(33$);


;
;
;   Model statements for module:  BasicProcess.Process 14 (Main Landing Gear Machining)
;
33$         ASSIGN:      Main Landing Gear Machining.NumberIn=Main Landing Gear Machining.NumberIn +
1:
                         Main Landing Gear Machining.WIP=Main Landing Gear Machining.WIP+1;
293$        STACK,       1:Save:NEXT(267$);

267$        QUEUE,       Main Landing Gear Machining.Queue;
266$        SEIZE,       2,VA:
                         ResourceMainGearMach,1:NEXT(265$);

265$        DELAY:
                         MX(      TRIA(0.85*(10      *          *      ((varSlopeMainGearMach      *      LOG(attSN)
)+log(varMainGearMachLN1))),(10      *          *      ((varSlopeMainGearMach      *      LOG(attSN)
)+log(varMainGearMachLN1))),1.15*(10      *          *      ((varSlopeMainGearMach      *      LOG(attSN)
)+log(varMainGearMachLN1))))                                                                    ,
TRIA(0.85*varMainGearMachLN100,varMainGearMachLN100,1.15*varMainGearMachLN100)),,
```

```
                VA:NEXT(308$);

308$        ASSIGN:      Main Landing Gear Machining.WaitTime=Main Landing Gear Machining.WaitTime +
Diff.WaitTime;
272$        TALLY:       Main Landing Gear Machining.WaitTimePerEntity,Diff.WaitTime,1;
274$        TALLY:       Main Landing Gear Machining.TotalTimePerEntity,Diff.StartTime,1;
298$          ASSIGN:       Main Landing Gear Machining.VATime=Main Landing Gear Machining.VATime +
Diff.VATime;
299$        TALLY:       Main Landing Gear Machining.VATimePerEntity,Diff.VATime,1;
264$        RELEASE:     ResourceMainGearMach,1;
313$        STACK,       1:Destroy:NEXT(312$);

312$                ASSIGN:              Main Landing Gear Machining.NumberOut=Main Landing Gear
Machining.NumberOut + 1:
                Main Landing Gear Machining.WIP=Main Landing Gear Machining.WIP-1:NEXT(27$);


;
;
;   Model statements for module:  BasicProcess.Assign 16 (Show MainGear Mach Time)
;
27$         ASSIGN:      varProcMainGearMach=
                MX((10  *    * ((varSlopeMainGearMach  *  LOG(attSN)  )+log(varMainGearMachLN1))),
varMainGearMachLN100)
                :NEXT(34$);


;
;
;   Model statements for module:  BasicProcess.Process 15 (Main Landing Gear BuildUp)
;
34$         ASSIGN:      Main Landing Gear BuildUp.NumberIn=Main Landing Gear BuildUp.NumberIn + 1:
                Main Landing Gear BuildUp.WIP=Main Landing Gear BuildUp.WIP+1;
344$        STACK,       1:Save:NEXT(318$);

318$        QUEUE,       Main Landing Gear BuildUp.Queue;
317$        SEIZE,       2,VA:
                ResourceMainGearBuildUp,1:NEXT(316$);

316$        DELAY:
                MX(     TRIA(0.85*(10     *          *     ((varSlopeMainGearBuildUp     *     LOG(attSN)
)+log(varMainGearBuildUpLN1))),(10     *          *     ((varSlopeMainGearBuildUp     *     LOG(attSN)
)+log(varMainGearBuildUpLN1))),1.15*(10     *          *     ((varSlopeMainGearBuildUp     *     LOG(attSN)
)+log(varMainGearBuildUpLN1))))                                                                    ,
TRIA(0.85*varMainGearBuildUpLN100,varMainGearBuildUpLN100,1.15*varMainGearBuildUpLN100)),,
                VA:NEXT(359$);

359$          ASSIGN:       Main Landing Gear BuildUp.WaitTime=Main Landing Gear BuildUp.WaitTime +
Diff.WaitTime;
323$        TALLY:       Main Landing Gear BuildUp.WaitTimePerEntity,Diff.WaitTime,1;
325$        TALLY:       Main Landing Gear BuildUp.TotalTimePerEntity,Diff.StartTime,1;
349$            ASSIGN:          Main Landing Gear BuildUp.VATime=Main Landing Gear BuildUp.VATime +
Diff.VATime;
350$        TALLY:       Main Landing Gear BuildUp.VATimePerEntity,Diff.VATime,1;
315$        RELEASE:     ResourceMainGearBuildUp,1;
364$        STACK,       1:Destroy:NEXT(363$);
```

```
363$        ASSIGN:       Main Landing Gear BuildUp.NumberOut=Main Landing Gear BuildUp.NumberOut +
1:
                  Main Landing Gear BuildUp.WIP=Main Landing Gear BuildUp.WIP-1:NEXT(112$);


;
;
;   Model statements for module:  BasicProcess.Process 111 (Ship Main Gear)
;
112$        ASSIGN:       Ship Main Gear.NumberIn=Ship Main Gear.NumberIn + 1:
                  Ship Main Gear.WIP=Ship Main Gear.WIP+1;
395$        STACK,        1:Save:NEXT(369$);

369$        QUEUE,        Ship Main Gear.Queue;
368$        SEIZE,        2,NVA:
                  ResourceMainGearShipper,1:NEXT(367$);

367$        DELAY:        5,,NVA:NEXT(410$);

410$        ASSIGN:       Ship Main Gear.WaitTime=Ship Main Gear.WaitTime + Diff.WaitTime;
374$        TALLY:        Ship Main Gear.WaitTimePerEntity,Diff.WaitTime,1;
376$        TALLY:        Ship Main Gear.TotalTimePerEntity,Diff.StartTime,1;
400$        ASSIGN:       Ship Main Gear.NVATime=Ship Main Gear.NVATime + Diff.NVATime;
401$        TALLY:        Ship Main Gear.NVATimePerEntity,Diff.NVATime,1;
366$        RELEASE:      ResourceMainGearShipper,1;
415$        STACK,        1:Destroy:NEXT(414$);

414$        ASSIGN:       Ship Main Gear.NumberOut=Ship Main Gear.NumberOut + 1:
                  Ship Main Gear.WIP=Ship Main Gear.WIP-1:NEXT(166$);


;
;
;   Model statements for module:  BasicProcess.Assign 40 (Assign Decision 2)
;
166$        ASSIGN:       varDecision2=DISC(0.50, 1, 1.0, 0):
                  attDelay2=varDelay2*varDecision2:NEXT(16$);


;
;
;   Model statements for module:  BasicProcess.Batch 2 (Batch for Final Assembly)
;
16$         QUEUE,        Batch for Final Assembly.Queue;
417$        GROUP,        attSN,Permanent:5,Last:NEXT(418$);

418$            ASSIGN:           Batch for Final Assembly.NumberOut=Batch for Final Assembly.NumberOut +
1:NEXT(168$);


;
;
;   Model statements for module:  BasicProcess.Assign 42 (Assign Decision 3)
;
168$        ASSIGN:       varDecision3=DISC(0.25, 1, 1.0, 0):
                  attDelay3=varDelay3*varDecision3:NEXT(165$);
```

```
;
;
;   Model statements for module:  BasicProcess.Assign 39 (Assign Option Percentage)
;
165$        ASSIGN:       varOption1=DISC(0.36, 1, 1.0, 0):
                          varOption2=DISC(0.60, 1, 1.0, 0):
                          varOption3=DISC(0.25, 1, 1.0, 0):
                          varOption4=DISC(0.60, 1, 1.0, 0):
                          varOption5=DISC(0.80, 1, 1.0, 0):
                          varOption6=DISC(0.70, 1, 1.0, 0):
                          varOption7=DISC(0.55, 1, 1.0, 0):
                          varOption8=DISC(0.55, 1, 1.0, 0):
                          varOption9=DISC(0.23, 1, 1.0, 0):
                          varOption10=DISC(0.40, 1, 1.0, 0):
                          varOptionAverage=
                          (varOption1 + varOption2 + varOption3 + varOption4 + varOption5 + varOption6 +
varOption7 + varOption8 + varOption9 + varOption10)/10
                          :NEXT(159$);


;
;
;   Model statements for module:  BasicProcess.Decide 1 (Decide Disruption)
;
159$        BRANCH,       1:
                          If,varTriggerLN1 == attSN || varTriggerLN2 ==attSN || varTriggerLN3 == attSN,419$,Yes:
                          Else,420$,Yes;
419$              ASSIGN:           Decide Disruption.NumberOut True=Decide Disruption.NumberOut True +
1:NEXT(161$);

420$              ASSIGN:           Decide Disruption.NumberOut False=Decide Disruption.NumberOut False +
1:NEXT(160$);


;
;
;   Model statements for module:  BasicProcess.Decide 2 (Decide 1st Disruption)
;
161$        BRANCH,       1:
                          If,attNewTrigger==0,421$,Yes:
                          Else,422$,Yes;
421$          ASSIGN:         Decide 1st Disruption.NumberOut True=Decide 1st Disruption.NumberOut True +
1:NEXT(158$);

422$          ASSIGN:         Decide 1st Disruption.NumberOut False=Decide 1st Disruption.NumberOut False +
1:NEXT(162$);


;
;
;   Model statements for module:  BasicProcess.Assign 35 (Assign Yes Disruption)
;
158$        ASSIGN:       attDisruption=1:
                          attDuration=varDuration1:
                          attDelta1=varDelta1 * ( attDuration/varDuration1 ):
                          attDuration=MX((attDuration-1), 0):
```

```
                attNewTrigger=1:
                attFAstart=TNOW:NEXT(9$);


;
;
;   Model statements for module:  BasicProcess.Process 10 (Final Assembly)
;
9$          ASSIGN:     Final Assembly.NumberIn=Final Assembly.NumberIn + 1:
                Final Assembly.WIP=Final Assembly.WIP+1;
452$        STACK,      1:Save:NEXT(426$);

426$        QUEUE,      Final Assembly.Queue;
425$        SEIZE,      2,VA:
                RésourceFinalAssembly,1:NEXT(424$);

424$        DELAY:
                (MX(    TRIA(0.85   *   (10   *       *   ((varSlopeFinalAssembly   *   LOG(attSN)
)+log(varFinalAssemblyLN1))),(10      *           *       ((varSlopeFinalAssembly       *   LOG(attSN)
)+log(varFinalAssemblyLN1))),1.15   *   (10   *       *   ((varSlopeFinalAssembly   *   LOG(attSN)
)+log(varFinalAssemblyLN1))))                                                                   ;
TRIA(0.85*varFinalAssemblyLN100,varFinalAssemblyLN100,1.15*varFinalAssemblyLN100)           ))   *
(attDelta1*attDisruption+1),,
                VA:NEXT(467$);

467$        ASSIGN:     Final Assembly.WaitTime=Final Assembly.WaitTime + Diff.WaitTime;
431$        TALLY:      Final Assembly.WaitTimePerEntity,Diff.WaitTime,1;
433$        TALLY:      Final Assembly.TotalTimePerEntity,Diff.StartTime,1;
457$        ASSIGN:     Final Assembly.VATime=Final Assembly.VATime + Diff.VATime;
458$        TALLY:      Final Assembly.VATimePerEntity,Diff.VATime,1;
423$        RELEASE:    ResourceFinalAssembly,1;
472$        STACK,      1:Destroy:NEXT(471$);

471$        ASSIGN:     Final Assembly.NumberOut=Final Assembly.NumberOut + 1:
                Final Assembly.WIP=Final Assembly.WIP-1:NEXT(22$);


;
;
;   Model statements for module:  BasicProcess.Assign 12 (Assign FA Status Report)
;
22$         ASSIGN:     varProcFinalAssembly=
                (MX(    TRIA(0.85   *   (10   *       *   ((varSlopeFinalAssembly   *   LOG(attSN)
)+log(varFinalAssemblyLN1))),(10      *           *       ((varSlopeFinalAssembly       *   LOG(attSN)
)+log(varFinalAssemblyLN1))),1.15   *   (10   *       *   ((varSlopeFinalAssembly   *   LOG(attSN)
)+log(varFinalAssemblyLN1))))                                                                   ;
TRIA(0.85*varFinalAssemblyLN100,varFinalAssemblyLN100,1.15*varFinalAssemblyLN100)           ))   *
(attDelta1*attDisruption+1):
                varDelta1Status=attDelta1:
                varNewTriggerStatus=attNewTrigger:NEXT(163$);


;
;
;   Model statements for module:  BasicProcess.Separate 14 (SeparateOption1)
;
163$        DUPLICATE,  100 - 50:
```

```
                1,476$,50:NEXT(475$);

475$            ASSIGN:         SeparateOption1.NumberOut Orig=SeparateOption1.NumberOut Orig +
1:NEXT(164$);

476$            ASSIGN:         SeparateOption1.NumberOut Dup=SeparateOption1.NumberOut Dup +
1:NEXT(170$);


;
;
;   Model statements for module:  BasicProcess.Process 112 (ProcessOpt1)
;
164$        ASSIGN:     ProcessOpt1.NumberIn=ProcessOpt1.NumberIn + 1:
                ProcessOpt1.WIP=ProcessOpt1.WIP+1;
506$        STACK,      1:Save:NEXT(480$);

480$        QUEUE,      ProcessOpt1.Queue;
479$        SEIZE,      2,VA:
                ResourceOptions,1:NEXT(478$);

478$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+varOpt1Time)*varOption1,
1*(attDelay1+attDelay2+varOpt1Time)*varOption1, 1.15*(attDelay1+attDelay2+varOpt1Time)*varOption1)),,
                VA:NEXT(521$);

521$        ASSIGN:     ProcessOpt1.WaitTime=ProcessOpt1.WaitTime + Diff.WaitTime;
485$        TALLY:      ProcessOpt1.WaitTimePerEntity,Diff.WaitTime,1;
487$        TALLY:      ProcessOpt1.TotalTimePerEntity,Diff.StartTime,1;
511$        ASSIGN:     ProcessOpt1.VATime=ProcessOpt1.VATime + Diff.VATime;
512$        TALLY:      ProcessOpt1.VATimePerEntity,Diff.VATime,1;
477$        RELEASE:    ResourceOptions,1;
526$        STACK,      1:Destroy:NEXT(525$);

525$        ASSIGN:     ProcessOpt1.NumberOut=ProcessOpt1.NumberOut + 1:
                ProcessOpt1.WIP=ProcessOpt1.WIP-1:NEXT(169$);


;
;
;   Model statements for module:  BasicProcess.Batch 9 (Batch Options)
;
169$        QUEUE,      Batch Options.Queue;
528$        GROUP,      attSN,Permanent:5,Last:NEXT(529$);

529$        ASSIGN:     Batch Options.NumberOut=Batch Options.NumberOut + 1:NEXT(10$);


;
;
;   Model statements for module:  BasicProcess.Process 11 (Final Paint)
;
10$         ASSIGN:     Final Paint.NumberIn=Final Paint.NumberIn + 1:
                Final Paint.WIP=Final Paint.WIP+1;
559$        STACK,      1:Save:NEXT(533$);

533$        QUEUE,      Final Paint.Queue;
```

```
532$        SEIZE,       2,VA:
                ResourceFinalPaint,1:NEXT(531$);

531$        DELAY:
                MX( TRIA(0.85* (10 *  * ((varSlopeFinalPaint * LOG(attSN) )+log(varFinalPaintLN1))), (10 *  *
((varSlopeFinalPaint * LOG(attSN) )+log(varFinalPaintLN1))),1.15* (10 *  * ((varSlopeFinalPaint * LOG(attSN)
)+log(varFinalPaintLN1)))) , TRIA(0.85* varFinalPaintLN100,varFinalPaintLN100,1.15* varFinalPaintLN100)),,
                VA:NEXT(574$);

574$        ASSIGN:      Final Paint.WaitTime=Final Paint.WaitTime + Diff.WaitTime;
538$        TALLY:       Final Paint.WaitTimePerEntity,Diff.WaitTime,1;
540$        TALLY:       Final Paint.TotalTimePerEntity,Diff.StartTime,1;
564$        ASSIGN:      Final Paint.VATime=Final Paint.VATime + Diff.VATime;
565$        TALLY:       Final Paint.VATimePerEntity,Diff.VATime,1;
530$        RELEASE:     ResourceFinalPaint,1;
579$        STACK,       1:Destroy:NEXT(578$);

578$        ASSIGN:      Final Paint.NumberOut=Final Paint.NumberOut + 1:
                Final Paint.WIP=Final Paint.WIP-1:NEXT(187$);


;
;
;   Model statements for module:  BasicProcess.Separate 23 (SeparateFinalOptions)
;
187$        DUPLICATE,    100 - 50:
                1,583$,50:NEXT(582$);

582$        ASSIGN:      SeparateFinalOptions.NumberOut Orig=SeparateFinalOptions.NumberOut Orig +
1:NEXT(11$);

583$        ASSIGN:      SeparateFinalOptions.NumberOut Dup=SeparateFinalOptions.NumberOut Dup +
1:NEXT(177$);


;
;
;   Model statements for module:  BasicProcess.Process 12 (Final Field)
;
11$         ASSIGN:      Final Field.NumberIn=Final Field.NumberIn + 1:
                Final Field.WIP=Final Field.WIP+1;
613$        STACK,       1:Save:NEXT(587$);

587$        QUEUE,       Final Field.Queue;
586$        SEIZE,       2,VA:
                ResourceFinalField,1:NEXT(585$);

585$        DELAY:
                MX( TRIA(0.85* (10 *  * ((varSlopeFinalField * LOG(attSN) )+log(varFinalFieldLN1))), (10 *  *
((varSlopeFinalField * LOG(attSN) )+log(varFinalFieldLN1))),1.15* (10 *  * ((varSlopeFinalField * LOG(attSN)
)+log(varFinalFieldLN1)))) , TRIA(0.85* varFinalFieldLN100,varFinalFieldLN100,1.15* varFinalFieldLN100)),,
                VA:NEXT(628$);

628$        ASSIGN:      Final Field.WaitTime=Final Field.WaitTime + Diff.WaitTime;
592$        TALLY:       Final Field.WaitTimePerEntity,Diff.WaitTime,1;
594$        TALLY:       Final Field.TotalTimePerEntity,Diff.StartTime,1;
618$        ASSIGN:      Final Field.VATime=Final Field.VATime + Diff.VATime;
```

```
619$        TALLY:       Final Field.VATimePerEntity,Diff.VATime,1;
584$        RELEASE:       ResourceFinalField,1;
633$        STACK,       1:Destroy:NEXT(632$);

632$        ASSIGN:      Final Field.NumberOut=Final Field.NumberOut + 1:
                         Final Field.WIP=Final Field.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Batch 10 (Batch Options Again)
;
185$        QUEUE,       Batch Options Again.Queue;
635$        GROUP,       attSN,Permanent:6,Last:NEXT(636$);

636$        ASSIGN:      Batch Options Again.NumberOut=Batch Options Again.NumberOut + 1:NEXT(188$);


;
;
;   Model statements for module:  BasicProcess.Record 1 (RecordFATime)
;
188$        TALLY:       TotalFATime,TNOW - attFAstart,1:NEXT(12$);


;
;
;   Model statements for module:  BasicProcess.Dispose 1 (Dispose 1)
;
12$         ASSIGN:      Dispose 1.NumberOut=Dispose 1.NumberOut + 1;
637$        DISPOSE:       Yes;


;
;
;   Model statements for module:  BasicProcess.Separate 19 (SeparateOpt6)
;
177$        DUPLICATE,     100 - 50:
                         1,640$,50:NEXT(639$);

639$        ASSIGN:      SeparateOpt6.NumberOut Orig=SeparateOpt6.NumberOut Orig + 1:NEXT(181$);

640$        ASSIGN:      SeparateOpt6.NumberOut Dup=SeparateOpt6.NumberOut Dup + 1:NEXT(178$);


;
;
;   Model statements for module:  BasicProcess.Process 117 (ProcessOpt6)
;
181$        ASSIGN:      ProcessOpt6.NumberIn=ProcessOpt6.NumberIn + 1:
                         ProcessOpt6.WIP=ProcessOpt6.WIP+1;
670$        STACK,       1:Save:NEXT(644$);

644$        QUEUE,       ProcessOpt6.Queue;
643$        SEIZE,       2,VA:
                         ResourceOptions,1:NEXT(642$);
```

```
642$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+attDelay3+varOpt6Time)*varOption6,
1*(attDelay1+attDelay2+attDelay3+varOpt6Time)*varOption6,
1.15*(attDelay1+attDelay2+attDelay3+varOpt6Time)*varOption6)),,
                VA:NEXT(685$);

685$        ASSIGN:     ProcessOpt6.WaitTime=ProcessOpt6.WaitTime + Diff.WaitTime;
649$        TALLY:      ProcessOpt6.WaitTimePerEntity,Diff.WaitTime,1;
651$        TALLY:      ProcessOpt6.TotalTimePerEntity,Diff.StartTime,1;
675$        ASSIGN:     ProcessOpt6.VATime=ProcessOpt6.VATime + Diff.VATime;
676$        TALLY:      ProcessOpt6.VATimePerEntity,Diff.VATime,1;
641$        RELEASE:    ResourceOptions,1;
690$        STACK,      1:Destroy:NEXT(689$);

689$        ASSIGN:     ProcessOpt6.NumberOut=ProcessOpt6.NumberOut + 1:
                ProcessOpt6.WIP=ProcessOpt6.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Separate 20 (SeparateOpt7)
;
178$        DUPLICATE,    100 - 50:
                1,694$,50:NEXT(693$);

693$        ASSIGN:     SeparateOpt7.NumberOut Orig=SeparateOpt7.NumberOut Orig + 1:NEXT(182$);

694$        ASSIGN:     SeparateOpt7.NumberOut Dup=SeparateOpt7.NumberOut Dup + 1:NEXT(179$);


;
;
;   Model statements for module:  BasicProcess.Process 118 (ProcessOpt7)
;
182$        ASSIGN:     ProcessOpt7.NumberIn=ProcessOpt7.NumberIn + 1:
                ProcessOpt7.WIP=ProcessOpt7.WIP+1;
724$        STACK,      1:Save:NEXT(698$);

698$        QUEUE,      ProcessOpt7.Queue;
697$        SEIZE,      2,VA:
                ResourceOptions,1:NEXT(696$);

696$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+attDelay3+varOpt7Time)*varOption7,
1*(attDelay1+attDelay2+attDelay3+varOpt7Time)*varOption7,
1.15*(attDelay1+attDelay2+attDelay3+varOpt7Time)*varOption7)),,
                VA:NEXT(739$);

739$        ASSIGN:     ProcessOpt7.WaitTime=ProcessOpt7.WaitTime + Diff.WaitTime;
703$        TALLY:      ProcessOpt7.WaitTimePerEntity,Diff.WaitTime,1;
705$        TALLY:      ProcessOpt7.TotalTimePerEntity,Diff.StartTime,1;
729$        ASSIGN:     ProcessOpt7.VATime=ProcessOpt7.VATime + Diff.VATime;
730$        TALLY:      ProcessOpt7.VATimePerEntity,Diff.VATime,1;
695$        RELEASE:    ResourceOptions,1;
744$        STACK,      1:Destroy:NEXT(743$);

743$        ASSIGN:     ProcessOpt7.NumberOut=ProcessOpt7.NumberOut + 1:
```

```
                ProcessOpt7.WIP=ProcessOpt7.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Separate 21 (SeparateOpt8)
;
179$        DUPLICATE,    100 - 50:
                1,748$,50:NEXT(747$);

747$        ASSIGN:       SeparateOpt8.NumberOut Orig=SeparateOpt8.NumberOut Orig + 1:NEXT(183$);

748$        ASSIGN:       SeparateOpt8.NumberOut Dup=SeparateOpt8.NumberOut Dup + 1:NEXT(180$);


;
;
;   Model statements for module:  BasicProcess.Process 119 (ProcessOpt8)
;
183$        ASSIGN:       ProcessOpt8.NumberIn=ProcessOpt8.NumberIn + 1:
                ProcessOpt8.WIP=ProcessOpt8.WIP+1;
778$        STACK,        1:Save:NEXT(752$);

752$        QUEUE,        ProcessOpt8.Queue;
751$        SEIZE,        2,VA:
                ResourceOptions,1:NEXT(750$);

750$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+attDelay3+varOpt8Time)*varOption8,
1*(attDelay1+attDelay2+attDelay3+varOpt8Time)*varOption8,
1.15*(attDelay1+attDelay2+attDelay3+varOpt8Time)*varOption8)),,
                VA:NEXT(793$);

793$        ASSIGN:       ProcessOpt8.WaitTime=ProcessOpt8.WaitTime + Diff.WaitTime;
757$        TALLY:        ProcessOpt8.WaitTimePerEntity,Diff.WaitTime,1;
759$        TALLY:        ProcessOpt8.TotalTimePerEntity,Diff.StartTime,1;
783$        ASSIGN:       ProcessOpt8.VATime=ProcessOpt8.VATime + Diff.VATime;
784$        TALLY:        ProcessOpt8.VATimePerEntity,Diff.VATime,1;
749$        RELEASE:      ResourceOptions,1;
798$        STACK,        1:Destroy:NEXT(797$);

797$        ASSIGN:       ProcessOpt8.NumberOut=ProcessOpt8.NumberOut + 1:
                ProcessOpt8.WIP=ProcessOpt8.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Separate 22 (SeparateOpt9)
;
180$        DUPLICATE,    100 - 50:
                1,802$,50:NEXT(801$);

801$        ASSIGN:       SeparateOpt9.NumberOut Orig=SeparateOpt9.NumberOut Orig + 1:NEXT(184$);

802$        ASSIGN:       SeparateOpt9.NumberOut Dup=SeparateOpt9.NumberOut Dup + 1:NEXT(186$);
```

```
;
;
;   Model statements for module:  BasicProcess.Process 120 (ProcessOpt9)
;
184$        ASSIGN:      ProcessOpt9.NumberIn=ProcessOpt9.NumberIn + 1:
                         ProcessOpt9.WIP=ProcessOpt9.WIP+1;
832$        STACK,       1:Save:NEXT(806$);

806$        QUEUE,       ProcessOpt9.Queue;
805$        SEIZE,       2,VA:
                         ResourceOptions,1:NEXT(804$);

804$        DELAY:
                         HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+attDelay3+varOpt9Time)*varOption9,
1*(attDelay1+attDelay2+attDelay3+varOpt9Time)*varOption9,
1.15*(attDelay1+attDelay2+attDelay3+varOpt9Time)*varOption9)),,
                         VA:NEXT(847$);

847$        ASSIGN:      ProcessOpt9.WaitTime=ProcessOpt9.WaitTime + Diff.WaitTime;
811$        TALLY:       ProcessOpt9.WaitTimePerEntity,Diff.WaitTime,1;
813$        TALLY:       ProcessOpt9.TotalTimePerEntity,Diff.StartTime,1;
837$        ASSIGN:      ProcessOpt9.VATime=ProcessOpt9.VATime + Diff.VATime;
838$        TALLY:       ProcessOpt9.VATimePerEntity,Diff.VATime,1;
803$        RELEASE:     ResourceOptions,1;
852$        STACK,       1:Destroy:NEXT(851$);

851$        ASSIGN:      ProcessOpt9.NumberOut=ProcessOpt9.NumberOut + 1:
                         ProcessOpt9.WIP=ProcessOpt9.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Process 121 (ProcessOpt10)
;
186$        ASSIGN:      ProcessOpt10.NumberIn=ProcessOpt10.NumberIn + 1:
                         ProcessOpt10.WIP=ProcessOpt10.WIP+1;
883$        STACK,       1:Save:NEXT(857$);

857$        QUEUE,       ProcessOpt10.Queue;
856$        SEIZE,       2,VA:
                         ResourceOptions,1:NEXT(855$);

855$        DELAY:
                         HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+attDelay3+varOpt10Time)*varOption10,
1*(attDelay1+attDelay2+attDelay3+varOpt10Time)*varOption10,
1.15*(attDelay1+attDelay2+attDelay3+varOpt10Time)*varOption10)),,
                         VA:NEXT(898$);

898$        ASSIGN:      ProcessOpt10.WaitTime=ProcessOpt10.WaitTime + Diff.WaitTime;
862$        TALLY:       ProcessOpt10.WaitTimePerEntity,Diff.WaitTime,1;
864$        TALLY:       ProcessOpt10.TotalTimePerEntity,Diff.StartTime,1;
888$        ASSIGN:      ProcessOpt10.VATime=ProcessOpt10.VATime + Diff.VATime;
889$        TALLY:       ProcessOpt10.VATimePerEntity,Diff.VATime,1;
854$        RELEASE:     ResourceOptions,1;
903$        STACK,       1:Destroy:NEXT(902$);

902$        ASSIGN:      ProcessOpt10.NumberOut=ProcessOpt10.NumberOut + 1:
```

```
                ProcessOpt10.WIP=ProcessOpt10.WIP-1:NEXT(185$);


;
;
;   Model statements for module:  BasicProcess.Separate 15 (SeparateOption2)
;
170$        DUPLICATE,    100 - 50:
                1,907$,50:NEXT(906$);

906$            ASSIGN:         SeparateOption2.NumberOut Orig=SeparateOption2.NumberOut Orig +
1:NEXT(171$);

907$            ASSIGN:         SeparateOption2.NumberOut Dup=SeparateOption2.NumberOut Dup +
1:NEXT(172$);


;
;
;   Model statements for module:  BasicProcess.Process 113 (ProcessOpt2)
;
171$        ASSIGN:     ProcessOpt2.NumberIn=ProcessOpt2.NumberIn + 1:
                ProcessOpt2.WIP=ProcessOpt2.WIP+1;
937$        STACK,      1:Save:NEXT(911$);

911$        QUEUE,      ProcessOpt2.Queue;
910$        SEIZE,      2,VA:
                ResourceOptions,1:NEXT(909$);

909$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+varOpt2Time)*varOption2,
1*(attDelay1+attDelay2+varOpt2Time)*varOption2, 1.15*(attDelay1+attDelay2+varOpt2Time)*varOption2)),,
                VA:NEXT(952$);

952$        ASSIGN:     ProcessOpt2.WaitTime=ProcessOpt2.WaitTime + Diff.WaitTime;
916$        TALLY:      ProcessOpt2.WaitTimePerEntity,Diff.WaitTime,1;
918$        TALLY:      ProcessOpt2.TotalTimePerEntity,Diff.StartTime,1;
942$        ASSIGN:     ProcessOpt2.VATime=ProcessOpt2.VATime + Diff.VATime;
943$        TALLY:      ProcessOpt2.VATimePerEntity,Diff.VATime,1;
908$        RELEASE:    ResourceOptions,1;
957$        STACK,      1:Destroy:NEXT(956$);

956$        ASSIGN:     ProcessOpt2.NumberOut=ProcessOpt2.NumberOut + 1:
                ProcessOpt2.WIP=ProcessOpt2.WIP-1:NEXT(169$);


;
;
;   Model statements for module:  BasicProcess.Separate 16 (SeparateOption3)
;
172$        DUPLICATE,    100 - 50:
                1,961$,50:NEXT(960$);

960$            ASSIGN:         SeparateOption3.NumberOut Orig=SeparateOption3.NumberOut Orig +
1:NEXT(174$);
```

```
961$            ASSIGN:         SeparateOption3.NumberOut Dup=SeparateOption3.NumberOut Dup +
1:NEXT(173$);


;
;
;   Model statements for module:  BasicProcess.Process 114 (ProcessOpt3)
;
174$        ASSIGN:     ProcessOpt3.NumberIn=ProcessOpt3.NumberIn + 1:
                ProcessOpt3.WIP=ProcessOpt3.WIP+1;
991$        STACK,      1:Save:NEXT(965$);

965$        QUEUE,      ProcessOpt3.Queue;
964$        SEIZE,      2,VA:
                ResourceOptions,1:NEXT(963$);

963$        DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+varOpt3Time)*varOption3,
1*(attDelay1+attDelay2+varOpt3Time)*varOption3, 1.15*(attDelay1+attDelay2+varOpt3Time)*varOption3)),,
                VA:NEXT(1006$);

1006$       ASSIGN:     ProcessOpt3.WaitTime=ProcessOpt3.WaitTime + Diff.WaitTime;
970$        TALLY:      ProcessOpt3.WaitTimePerEntity,Diff.WaitTime,1;
972$        TALLY:      ProcessOpt3.TotalTimePerEntity,Diff.StartTime,1;
996$        ASSIGN:     ProcessOpt3.VATime=ProcessOpt3.VATime + Diff.VATime;
997$        TALLY:      ProcessOpt3.VATimePerEntity,Diff.VATime,1;
962$        RELEASE:    ResourceOptions,1;
1011$       STACK,      1:Destroy:NEXT(1010$);

1010$       ASSIGN:     ProcessOpt3.NumberOut=ProcessOpt3.NumberOut + 1:
                ProcessOpt3.WIP=ProcessOpt3.WIP-1:NEXT(169$);


;
;
;   Model statements for module:  BasicProcess.Separate 17 (SeparateOption4)
;
173$        DUPLICATE,  100 - 50:
                1,1015$,50:NEXT(1014$);

1014$           ASSIGN:         SeparateOption4.NumberOut Orig=SeparateOption4.NumberOut Orig +
1:NEXT(175$);

1015$           ASSIGN:         SeparateOption4.NumberOut Dup=SeparateOption4.NumberOut Dup +
1:NEXT(176$);


;
;
;   Model statements for module:  BasicProcess.Process 115 (ProcessOpt4)
;
175$        ASSIGN:     ProcessOpt4.NumberIn=ProcessOpt4.NumberIn + 1:
                ProcessOpt4.WIP=ProcessOpt4.WIP+1;
1045$       STACK,      1:Save:NEXT(1019$);

1019$       QUEUE,      ProcessOpt4.Queue;
1018$       SEIZE,      2,VA:
```

```
                ResourceOptions,1:NEXT(1017$);

1017$       DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+varOpt4Time)*varOption4,
1*(attDelay1+attDelay2+varOpt4Time)*varOption4, 1.15*(attDelay1+attDelay2+varOpt4Time)*varOption4)),,
                VA:NEXT(1060$);

1060$       ASSIGN:     ProcessOpt4.WaitTime=ProcessOpt4.WaitTime + Diff.WaitTime;
1024$       TALLY:      ProcessOpt4.WaitTimePerEntity,Diff.WaitTime,1;
1026$       TALLY:      ProcessOpt4.TotalTimePerEntity,Diff.StartTime,1;
1050$       ASSIGN:     ProcessOpt4.VATime=ProcessOpt4.VATime + Diff.VATime;
1051$       TALLY:      ProcessOpt4.VATimePerEntity,Diff.VATime,1;
1016$       RELEASE:    ResourceOptions,1;
1065$       STACK,      1:Destroy:NEXT(1064$);

1064$       ASSIGN:     ProcessOpt4.NumberOut=ProcessOpt4.NumberOut + 1:
                ProcessOpt4.WIP=ProcessOpt4.WIP-1:NEXT(169$);


;
;
;   Model statements for module:  BasicProcess.Process 116 (ProcessOpt5)
;
176$        ASSIGN:     ProcessOpt5.NumberIn=ProcessOpt5.NumberIn + 1:
                ProcessOpt5.WIP=ProcessOpt5.WIP+1;
1096$       STACK,      1:Save:NEXT(1070$);

1070$       QUEUE,      ProcessOpt5.Queue;
1069$       SEIZE,      2,VA:
                ResourceOptions,1:NEXT(1068$);

1068$       DELAY:
                HoursToBaseTime(TRIA(0.85*(attDelay1+attDelay2+varOpt5Time)*varOption5,
1*(attDelay1+attDelay2+varOpt5Time)*varOption5, 1.15*(attDelay1+attDelay2+varOpt5Time)*varOption5)),,
                VA:NEXT(1111$);

1111$       ASSIGN:     ProcessOpt5.WaitTime=ProcessOpt5.WaitTime + Diff.WaitTime;
1075$       TALLY:      ProcessOpt5.WaitTimePerEntity,Diff.WaitTime,1;
1077$       TALLY:      ProcessOpt5.TotalTimePerEntity,Diff.StartTime,1;
1101$       ASSIGN:     ProcessOpt5.VATime=ProcessOpt5.VATime + Diff.VATime;
1102$       TALLY:      ProcessOpt5.VATimePerEntity,Diff.VATime,1;
1067$       RELEASE:    ResourceOptions,1;
1116$       STACK,      1:Destroy:NEXT(1115$);

1115$       ASSIGN:     ProcessOpt5.NumberOut=ProcessOpt5.NumberOut + 1:
                ProcessOpt5.WIP=ProcessOpt5.WIP-1:NEXT(169$);


;
;
;   Model statements for module:  BasicProcess.Assign 38 (Assign Cont Disruption)
;
162$        ASSIGN:     attDisruption=1:
                attDelta1=varDelta1 *  ( attDuration/varDuration1 ):
                attDuration=MX((attDuration-1), 0):
                attNewTrigger=attDuration:
                attFAstart=TNOW:NEXT(9$);
```

```
;
;
;   Model statements for module:  BasicProcess.Assign 36 (Assign No Disruption)
;
160$        ASSIGN:      attDisruption=0:
                         attFAstart=TNOW:NEXT(9$);


;
;
;   Model statements for module:  BasicProcess.Create 6 (Sec41 Order)
;
1118$       CREATE,      1,DaysToBaseTime(0.0),Sec41:DaysToBaseTime(1),1:NEXT(1119$);

1119$       ASSIGN:      Sec41 Order.NumberOut=Sec41 Order.NumberOut + 1:NEXT(19$);


;
;
;   Model statements for module:  BasicProcess.Assign 8 (Assign CurveRate Sec41)
;
19$         ASSIGN:      varSlopeSec41Fab=LOG(varCurveRate(94)/100)/LOG(2):
                         varSec41FabLN1=10 ** (LOG(varSec41FabLN100) - varSlopeSec41Fab*LOG(100)):
                         varSlopeSec41Asm=LOG(varCurveRate(85)/100)/LOG(2):
                         varSec41AsmLN1=10 ** (LOG(varSec41AsmLN100) - varSlopeSec41Asm*LOG(100)):
                         varSlopeFinalAssembly=LOG(varCurveRate(78)/100)/LOG(2):
                         varFinalAssemblyLN1=10 * * (LOG(varFinalAssemblyLN100) -
varSlopeFinalAssembly*LOG(100)):
                         varFinalAsmAfterEngineLN1=
                         10 ** (LOG(varFinalAsmAfterEngineLN100) - varSlopeFinalAssembly*LOG(100)):
                         varSlopeFinalPaint=LOG(varCurveRate(93)/100)/LOG(2):
                         varFinalPaintLN1=10 ** (LOG(varFinalPaintLN100) - varSlopeFinalPaint*LOG(100)):
                         varSlopeFinalField=LOG(varCurveRate(78)/100)/LOG(2):
                         varFinalFieldLN1=10 ** (LOG(varFinalFieldLN100) - varSlopeFinalField*LOG(100)):
                         varSlopeSec41NG=LOG(varCurveRate(85)/100)/LOG(2):
                         varSec41NGLN1=10 * * (LOG(varSec41NGLN100) -
varSlopeSec41NG*LOG(100)):NEXT(35$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 2 (Read Sec41 Order Schedule)
;
35$         READ,        MDaySchedule787,RECORDSET(RecordsetSec41):
                         Sec41Time:NEXT(36$);


;
;
;   Model statements for module:  BasicProcess.Separate 2 (Separate Sec41)
;
36$         DUPLICATE,   100 - 50:
                         1,1124$,50:NEXT(1123$);
```

```
1123$       ASSIGN:      Separate Sec41.NumberOut Orig=Separate Sec41.NumberOut Orig + 1:NEXT(37$);

1124$       ASSIGN:      Separate Sec41.NumberOut Dup=Separate Sec41.NumberOut Dup + 1:NEXT(35$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 2 (Delay Sec41 Order)
;
37$         DELAY:       Sec41Time,,Other:NEXT(15$);


;
;
;   Model statements for module:  BasicProcess.Assign 3 (Assign Sec41 Serial Number)
;
15$         ASSIGN:      varPartSec41=varPartSec41 + 1:
                         attSN=varPartSec41:NEXT(38$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 2 (Route Sec41 Order)
;
38$         ROUTE:       0.,StnSec41Order;


;
;
;   Model statements for module:  AdvancedTransfer.Station 2 (Sec41 Order Arrival)
;

39$         STATION,     StnSec41Order;
1127$       DELAY:       0.0,,VA:NEXT(4$);


;
;
;   Model statements for module:  BasicProcess.Process 5 (Sec 41 Fab)
;
4$          ASSIGN:      Sec 41 Fab.NumberIn=Sec 41 Fab.NumberIn + 1:
                         Sec 41 Fab.WIP=Sec 41 Fab.WIP+1;
1157$       STACK,       1:Save:NEXT(1131$);

1131$       QUEUE,       Sec 41 Fab.Queue;
1130$       SEIZE,       2,VA:
                         ResourceSec41Fab,1:NEXT(1129$);

1129$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec41Fab * LOG(attSN) )+log(varSec41FabLN1))), (10 *
*  ((varSlopeSec41Fab  *  LOG(attSN)  )+log(varSec41FabLN1))),1.15*  (10  *    *  ((varSlopeSec41Fab  *
LOG(attSN)   )+log(varSec41FabLN1))))   ,      TRIA(0.85*   varSec41FabLN100,varSec41FabLN100,1.15*
varSec41FabLN100) ),,
                         VA:NEXT(1172$);

1172$       ASSIGN:      Sec 41 Fab.WaitTime=Sec 41 Fab.WaitTime + Diff.WaitTime;
1136$       TALLY:       Sec 41 Fab.WaitTimePerEntity,Diff.WaitTime,1;
```

```
1138$       TALLY:       Sec 41 Fab.TotalTimePerEntity,Diff.StartTime,1;
1162$       ASSIGN:       Sec 41 Fab.VATime=Sec 41 Fab.VATime + Diff.VATime;
1163$       TALLY:       Sec 41 Fab.VATimePerEntity,Diff.VATime,1;
1128$       RELEASE:       ResourceSec41Fab,1;
1177$       STACK,       1:Destroy:NEXT(1176$);

1176$       ASSIGN:       Sec 41 Fab.NumberOut=Sec 41 Fab.NumberOut + 1:
                  Sec 41 Fab.WIP=Sec 41 Fab.WIP-1:NEXT(5$);


;
;
;   Model statements for module:  BasicProcess.Process 6 (Sec 41 Assembly)
;
5$          ASSIGN:       Sec 41 Assembly.NumberIn=Sec 41 Assembly.NumberIn + 1:
                  Sec 41 Assembly.WIP=Sec 41 Assembly.WIP+1;
1208$       STACK,       1:Save:NEXT(1182$);

1182$       QUEUE,       Sec 41 Assembly.Queue;
1181$       SEIZE,       2,VA:
                  ResourceSec41Asm,1:NEXT(1180$);

1180$       DELAY:
                  MX( TRIA(0.85* (10 *  * ((varSlopeSec41Asm * LOG(attSN) )+log(varSec41AsmLN1))), (10 *
* ((varSlopeSec41Asm  *  LOG(attSN)  )+log(varSec41AsmLN1))),1.15*  (10  *    * ((varSlopeSec41Asm  *
LOG(attSN)   )+log(varSec41AsmLN1))))   ,   TRIA(0.85*   varSec41AsmLN100,varSec41AsmLN100,1.15*
varSec41AsmLN100)),,
                  VA:NEXT(1223$);

1223$       ASSIGN:       Sec 41 Assembly.WaitTime=Sec 41 Assembly.WaitTime + Diff.WaitTime;
1187$       TALLY:       Sec 41 Assembly.WaitTimePerEntity,Diff.WaitTime,1;
1189$       TALLY:       Sec 41 Assembly.TotalTimePerEntity,Diff.StartTime,1;
1213$       ASSIGN:       Sec 41 Assembly.VATime=Sec 41 Assembly.VATime + Diff.VATime;
1214$       TALLY:       Sec 41 Assembly.VATimePerEntity,Diff.VATime,1;
1179$       RELEASE:       ResourceSec41Asm,1;
1228$       STACK,       1:Destroy:NEXT(1227$);

1227$       ASSIGN:       Sec 41 Assembly.NumberOut=Sec 41 Assembly.NumberOut + 1:
                  Sec 41 Assembly.WIP=Sec 41 Assembly.WIP-1:NEXT(45$);


;
;
;   Model statements for module:  BasicProcess.Batch 4 (Batch Sec41 NoseGear)
;
45$         QUEUE,       Batch Sec41 NoseGear.Queue;
1230$       GROUP,       attSN,Permanent:2,Last:NEXT(1231$);

1231$          ASSIGN:          Batch  Sec41  NoseGear.NumberOut=Batch  Sec41  NoseGear.NumberOut  +
1:NEXT(46$);


;
;
;   Model statements for module:  BasicProcess.Process 16 (Sec41 and NoseGear)
;
46$         ASSIGN:       Sec41 and NoseGear.NumberIn=Sec41 and NoseGear.NumberIn + 1:
```

```
                Sec41 and NoseGear.WIP=Sec41 and NoseGear.WIP+1;
1261$       STACK,      1:Save:NEXT(1235$);

1235$       QUEUE,      Sec41 and NoseGear.Queue;
1234$       SEIZE,      2,VA:
                ResourceSec41NG,1:NEXT(1233$);

1233$       DELAY:
                MX(TRIA(0.85* (10 *  * ((varSlopeSec41NG * LOG(attSN) )+log(varSec41NGLN1))), (10 *  *
((varSlopeSec41NG * LOG(attSN) )+log(varSec41NGLN1))),1.15* (10 *  * ((varSlopeSec41NG * LOG(attSN)
)+log(varSec41NGLN1)))) , TRIA(0.85* varSec41NGLN100,varSec41NGLN100,1.15* varSec41NGLN100)),,
                VA:NEXT(1276$);

1276$       ASSIGN:     Sec41 and NoseGear.WaitTime=Sec41 and NoseGear.WaitTime + Diff.WaitTime;
1240$       TALLY:      Sec41 and NoseGear.WaitTimePerEntity,Diff.WaitTime,1;
1242$       TALLY:      Sec41 and NoseGear.TotalTimePerEntity,Diff.StartTime,1;
1266$       ASSIGN:     Sec41 and NoseGear.VATime=Sec41 and NoseGear.VATime + Diff.VATime;
1267$       TALLY:      Sec41 and NoseGear.VATimePerEntity,Diff.VATime,1;
1232$       RELEASE:    ResourceSec41NG,1;
1281$       STACK,      1:Destroy:NEXT(1280$);

1280$       ASSIGN:     Sec41 and NoseGear.NumberOut=Sec41 and NoseGear.NumberOut + 1:
                Sec41 and NoseGear.WIP=Sec41 and NoseGear.WIP-1:NEXT(110$);


;
;
;   Model statements for module:  BasicProcess.Process 109 (Ship Sec 41)
;
110$        ASSIGN:     Ship Sec 41.NumberIn=Ship Sec 41.NumberIn + 1:
                Ship Sec 41.WIP=Ship Sec 41.WIP+1;
1312$       STACK,      1:Save:NEXT(1286$);

1286$       QUEUE,      Ship Sec 41.Queue;
1285$       SEIZE,      2,NVA:
                ResourceSec41Shipper,1:NEXT(1284$);

1284$       DELAY:      3,,NVA:NEXT(1327$);

1327$       ASSIGN:     Ship Sec 41.WaitTime=Ship Sec 41.WaitTime + Diff.WaitTime;
1291$       TALLY:      Ship Sec 41.WaitTimePerEntity,Diff.WaitTime,1;
1293$       TALLY:      Ship Sec 41.TotalTimePerEntity,Diff.StartTime,1;
1317$       ASSIGN:     Ship Sec 41.NVATime=Ship Sec 41.NVATime + Diff.NVATime;
1318$       TALLY:      Ship Sec 41.NVATimePerEntity,Diff.NVATime,1;
1283$       RELEASE:    ResourceSec41Shipper,1;
1332$       STACK,      1:Destroy:NEXT(1331$);

1331$       ASSIGN:     Ship Sec 41.NumberOut=Ship Sec 41.NumberOut + 1:
                Ship Sec 41.WIP=Ship Sec 41.WIP-1:NEXT(16$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 3 (Sec43 Order Arrival)
;

44$         STATION,    StnSec43Order;
```

```
1336$       DELAY:       0.0,,VA:NEXT(1$);


;
;
;   Model statements for module:  BasicProcess.Process 2 (Sec 43 Fab)
;
1$          ASSIGN:      Sec 43 Fab.NumberIn=Sec 43 Fab.NumberIn + 1:
                         Sec 43 Fab.WIP=Sec 43 Fab.WIP+1;
1366$       STACK,       1:Save:NEXT(1340$);

1340$       QUEUE,       Sec 43 Fab.Queue;
1339$       SEIZE,       2,VA:
                         ResourceSec43Fab,1:NEXT(1338$);

1338$       DELAY:
                         MX( TRIA(0.85*(10 *  * ((varSlopeSec43Fab * LOG(attSN) )+log(varSec43FabLN1))),(10 *  *
((varSlopeSec43Fab * LOG(attSN) )+log(varSec43FabLN1))),1.15*(10 *  * ((varSlopeSec43Fab * LOG(attSN)
)+log(varSec43FabLN1)))) , TRIA(0.85*varSec43FabLN100,varSec43FabLN100,1.15*varSec43FabLN100)),,
                         VA:NEXT(1381$);

1381$       ASSIGN:      Sec 43 Fab.WaitTime=Sec 43 Fab.WaitTime + Diff.WaitTime;
1345$       TALLY:       Sec 43 Fab.WaitTimePerEntity,Diff.WaitTime,1;
1347$       TALLY:       Sec 43 Fab.TotalTimePerEntity,Diff.StartTime,1;
1371$       ASSIGN:      Sec 43 Fab.VATime=Sec 43 Fab.VATime + Diff.VATime;
1372$       TALLY:       Sec 43 Fab.VATimePerEntity,Diff.VATime,1;
1337$       RELEASE:     ResourceSec43Fab,1;
1386$       STACK,       1:Destroy:NEXT(1385$);

1385$       ASSIGN:      Sec 43 Fab.NumberOut=Sec 43 Fab.NumberOut + 1:
                         Sec 43 Fab.WIP=Sec 43 Fab.WIP-1:NEXT(21$);


;
;
;   Model statements for module:  BasicProcess.Assign 10 (Show ProcSec43 Fab Time)
;
21$         ASSIGN:      varProcSec43Fab=
                         MX( TRIA(0.85*(10 *  * ((varSlopeSec43Fab * LOG(attSN) )+log(varSec43FabLN1))),(10 *  *
((varSlopeSec43Fab * LOG(attSN) )+log(varSec43FabLN1))),1.15*(10 *  * ((varSlopeSec43Fab * LOG(attSN)
)+log(varSec43FabLN1)))) , TRIA(0.85*varSec43FabLN100,varSec43FabLN100,1.15*varSec43FabLN100))
                         :NEXT(2$);


;
;
;   Model statements for module:  BasicProcess.Process 3 (Sec 43 Asm Minor)
;
2$          ASSIGN:      Sec 43 Asm Minor.NumberIn=Sec 43 Asm Minor.NumberIn + 1:
                         Sec 43 Asm Minor.WIP=Sec 43 Asm Minor.WIP+1;
1417$       STACK,       1:Save:NEXT(1391$);

1391$       QUEUE,       Sec 43 Asm Minor.Queue;
1390$       SEIZE,       2,VA:
                         ResourceSec43AsmMinor,1:NEXT(1389$);

1389$       DELAY:
```

```
                MX(     TRIA(0.85*(10     *          *     ((varSlopeSec43AsmMinor     *     LOG(attSN)
)+log(varSec43AsmMinorLN1))),(10     *          *     ((varSlopeSec43AsmMinor     *     LOG(attSN)
)+log(varSec43AsmMinorLN1))),1.15*(10     *          *     ((varSlopeSec43AsmMinor     *     LOG(attSN)
)+log(varSec43AsmMinorLN1))))                                                                    ,
TRIA(0.85*varSec43AsmMinorLN100,varSec43AsmMinorLN100,1.15*varSec43AsmMinorLN100)),,
                VA:NEXT(1432$);

1432$     ASSIGN:     Sec 43 Asm Minor.WaitTime=Sec 43 Asm Minor.WaitTime + Diff.WaitTime;
1396$     TALLY:      Sec 43 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
1398$     TALLY:      Sec 43 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
1422$     ASSIGN:     Sec 43 Asm Minor.VATime=Sec 43 Asm Minor.VATime + Diff.VATime;
1423$     TALLY:      Sec 43 Asm Minor.VATimePerEntity,Diff.VATime,1;
1388$     RELEASE:    ResourceSec43AsmMinor,1;
1437$     STACK,      1:Destroy:NEXT(1436$);

1436$     ASSIGN:     Sec 43 Asm Minor.NumberOut=Sec 43 Asm Minor.NumberOut + 1:
                Sec 43 Asm Minor.WIP=Sec 43 Asm Minor.WIP-1:NEXT(3$);


;
;
;   Model statements for module:  BasicProcess.Process 4 (Sec 43 Asm Major)
;
3$        ASSIGN:     Sec 43 Asm Major.NumberIn=Sec 43 Asm Major.NumberIn + 1:
                Sec 43 Asm Major.WIP=Sec 43 Asm Major.WIP+1;
1468$     STACK,      1:Save:NEXT(1442$);

1442$     QUEUE,      Sec 43 Asm Major.Queue;
1441$     SEIZE,      2,VA:
                ResourceSec43AsmMajor,1:NEXT(1440$);

1440$     DELAY:
                MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec43AsmMajor     *     LOG(attSN)
)+log(varSec43AsmMajorLN1))),     (10     *          *     ((varSlopeSec43AsmMajor     *     LOG(attSN)
)+log(varSec43AsmMajorLN1))),1.15*     (10     *          *     ((varSlopeSec43AsmMajor     *     LOG(attSN)
)+log(varSec43AsmMajorLN1))))     ,     TRIA(0.85*     varSec43AsmMajorLN100,varSec43AsmMajorLN100,1.15*
varSec43AsmMajorLN100)),,
                VA:NEXT(1483$);

1483$     ASSIGN:     Sec 43 Asm Major.WaitTime=Sec 43 Asm Major.WaitTime + Diff.WaitTime;
1447$     TALLY:      Sec 43 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
1449$     TALLY:      Sec 43 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
1473$     ASSIGN:     Sec 43 Asm Major.VATime=Sec 43 Asm Major.VATime + Diff.VATime;
1474$     TALLY:      Sec 43 Asm Major.VATimePerEntity,Diff.VATime,1;
1439$     RELEASE:    ResourceSec43AsmMajor,1;
1488$     STACK,      1:Destroy:NEXT(1487$);

1487$     ASSIGN:     Sec 43 Asm Major.NumberOut=Sec 43 Asm Major.NumberOut + 1:
                Sec 43 Asm Major.WIP=Sec 43 Asm Major.WIP-1:NEXT(105$);


;
;
;   Model statements for module:  BasicProcess.Process 104 (Ship Sec 43)
;
105$      ASSIGN:     Ship Sec 43.NumberIn=Ship Sec 43.NumberIn + 1:
                Ship Sec 43.WIP=Ship Sec 43.WIP+1;
```

```
1519$       STACK,       1:Save:NEXT(1493$);

1493$       QUEUE,       Ship Sec 43.Queue;
1492$       SEIZE,       2,NVA:
                ResourceSec43Shipper,1:NEXT(1491$);

1491$       DELAY:       3,,NVA:NEXT(1534$);

1534$       ASSIGN:      Ship Sec 43.WaitTime=Ship Sec 43.WaitTime + Diff.WaitTime;
1498$       TALLY:       Ship Sec 43.WaitTimePerEntity,Diff.WaitTime,1;
1500$       TALLY:       Ship Sec 43.TotalTimePerEntity,Diff.StartTime,1;
1524$       ASSIGN:      Ship Sec 43.NVATime=Ship Sec 43.NVATime + Diff.NVATime;
1525$       TALLY:       Ship Sec 43.NVATimePerEntity,Diff.NVATime,1;
1490$       RELEASE:     ResourceSec43Shipper,1;
1539$       STACK,       1:Destroy:NEXT(1538$);

1538$       ASSIGN:      Ship Sec 43.NumberOut=Ship Sec 43.NumberOut + 1:
                Ship Sec 43.WIP=Ship Sec 43.WIP-1:NEXT(20$);


;
;
;   Model statements for module:  BasicProcess.Batch 3 (Batch for Pre Integration)
;
20$         QUEUE,       Batch for Pre Integration.Queue;
1541$       GROUP,       attSN,Permanent:4,Last:NEXT(1542$);

1542$           ASSIGN:          Batch for Pre Integration.NumberOut=Batch for Pre Integration.NumberOut +
1:NEXT(8$);


;
;
;   Model statements for module:  BasicProcess.Process 9 (Pre Integration)
;
8$          ASSIGN:      Pre Integration.NumberIn=Pre Integration.NumberIn + 1:
                Pre Integration.WIP=Pre Integration.WIP+1;
1572$       STACK,       1:Save:NEXT(1546$);

1546$       QUEUE,       Pre Integration.Queue;
1545$       SEIZE,       2,VA:
                ResourcePreIntegration,1:NEXT(1544$);

1544$       DELAY:
                MX(     TRIA(0.85*     (10     *          *     ((varSlopePreIntegration     *     LOG(attSN)
)+log(varPreIntegrationLN1))),     (10     *          *     ((varSlopePreIntegration     *     LOG(attSN)
)+log(varPreIntegrationLN1))),1.15*     (10     *          *     ((varSlopePreIntegration     *     LOG(attSN)
)+log(varPreIntegrationLN1))))     ,     TRIA(0.85*     varPreIntegrationLN100,varPreIntegrationLN100,1.15*
varPreIntegrationLN100)),,
                VA:NEXT(1587$);

1587$       ASSIGN:      Pre Integration.WaitTime=Pre Integration.WaitTime + Diff.WaitTime;
1551$       TALLY:       Pre Integration.WaitTimePerEntity,Diff.WaitTime,1;
1553$       TALLY:       Pre Integration.TotalTimePerEntity,Diff.StartTime,1;
1577$       ASSIGN:      Pre Integration.VATime=Pre Integration.VATime + Diff.VATime;
1578$       TALLY:       Pre Integration.VATimePerEntity,Diff.VATime,1;
1543$       RELEASE:     ResourcePreIntegration,1;
```

```
1592$       STACK,       1:Destroy:NEXT(1591$);

1591$       ASSIGN:      Pre Integration.NumberOut=Pre Integration.NumberOut + 1:
                         Pre Integration.WIP=Pre Integration.WIP-1:NEXT(78$);


;
;
;   Model statements for module: BasicProcess.Batch 6 (Sec49 to Pre Integration)
;
78$         QUEUE,       Sec49 to Pre Integration.Queue;
1594$       GROUP,       attSN,Permanent:2,Last:NEXT(1595$);

1595$       ASSIGN:      Sec49 to Pre Integration.NumberOut=Sec49 to Pre Integration.NumberOut +
1:NEXT(79$);


;
;
;   Model statements for module: BasicProcess.Process 33 (Pre Integration and Sec49)
;
79$         ASSIGN:      Pre Integration and Sec49.NumberIn=Pre Integration and Sec49.NumberIn + 1:
                         Pre Integration and Sec49.WIP=Pre Integration and Sec49.WIP+1;
1625$       STACK,       1:Save:NEXT(1599$);

1599$       QUEUE,       Pre Integration and Sec49.Queue;
1598$       SEIZE,       2,VA:
                         ResourcePreInteSec49,1:NEXT(1597$);

1597$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopePreIntegration * LOG(attSN) )+log(varPreInteSec49LN1))),
(10  *      *  ((varSlopePreIntegration  *  LOG(attSN)  )+log(varPreInteSec49LN1))),1.15*  (10  *      *
((varSlopePreIntegration      *      LOG(attSN)      )+log(varPreInteSec49LN1))))      ,      TRIA(0.85*
varPreInteSec49LN100,varPreInteSec49LN100,1.15*varPreInteSec49LN100)),,
                         VA:NEXT(1640$);

1640$       ASSIGN:      Pre Integration and Sec49.WaitTime=Pre Integration and Sec49.WaitTime +
Diff.WaitTime;
1604$       TALLY:       Pre Integration and Sec49.WaitTimePerEntity,Diff.WaitTime,1;
1606$       TALLY:       Pre Integration and Sec49.TotalTimePerEntity,Diff.StartTime,1;
1630$       ASSIGN:      Pre Integration and Sec49.VATime=Pre Integration and Sec49.VATime +
Diff.VATime;
1631$       TALLY:       Pre Integration and Sec49.VATimePerEntity,Diff.VATime,1;
1596$       RELEASE:     ResourcePreInteSec49,1;
1645$       STACK,       1:Destroy:NEXT(1644$);

1644$       ASSIGN:      Pre Integration and Sec49.NumberOut=Pre Integration and Sec49.NumberOut + 1:
                         Pre Integration and Sec49.WIP=Pre Integration and Sec49.WIP-1:NEXT(23$);


;
;
;   Model statements for module: BasicProcess.Assign 13 (Assign 13)
;
23$         ASSIGN:      varProcPreIntegration=
                         MX((10  *      *  ((varSlopePreIntegration  *  LOG(attSN)  )+log(varPreIntegrationLN1))),
varPreIntegrationLN100)
```

```
                    :NEXT(108$);


;
;
;   Model statements for module:  BasicProcess.Process 107 (Ship Pre Inte Fuselage)
;
108$        ASSIGN:      Ship Pre Inte Fuselage.NumberIn=Ship Pre Inte Fuselage.NumberIn + 1:
                         Ship Pre Inte Fuselage.WIP=Ship Pre Inte Fuselage.WIP+1;
1676$       STACK,       1:Save:NEXT(1650$);

1650$       QUEUE,       Ship Pre Inte Fuselage.Queue;
1649$       SEIZE,       2,NVA:
                         ResourcePreInteShipper,1:NEXT(1648$);

1648$       DELAY:       3,,NVA:NEXT(1691$);

1691$       ASSIGN:      Ship Pre Inte Fuselage.WaitTime=Ship Pre Inte Fuselage.WaitTime + Diff.WaitTime;
1655$       TALLY:       Ship Pre Inte Fuselage.WaitTimePerEntity,Diff.WaitTime,1;
1657$       TALLY:       Ship Pre Inte Fuselage.TotalTimePerEntity,Diff.StartTime,1;
1681$       ASSIGN:      Ship Pre Inte Fuselage.NVATime=Ship Pre Inte Fuselage.NVATime + Diff.NVATime;
1682$       TALLY:       Ship Pre Inte Fuselage.NVATimePerEntity,Diff.NVATime,1;
1647$       RELEASE:     ResourcePreInteShipper,1;
1696$       STACK,       1:Destroy:NEXT(1695$);

1695$       ASSIGN:      Ship Pre Inte Fuselage.NumberOut=Ship Pre Inte Fuselage.NumberOut + 1:
                         Ship Pre Inte Fuselage.WIP=Ship Pre Inte Fuselage.WIP-1:NEXT(16$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 4 (Nose Gear Order Arrival)
;

51$         STATION,     StnNoseGearOrder;
1700$       DELAY:       0.0,,VA:NEXT(0$);


;
;
;   Model statements for module:  BasicProcess.Process 1 (Nose Gear Forging)
;
0$          ASSIGN:      Nose Gear Forging.NumberIn=Nose Gear Forging.NumberIn + 1:
                         Nose Gear Forging.WIP=Nose Gear Forging.WIP+1;
1730$       STACK,       1:Save:NEXT(1704$);

1704$       QUEUE,       Nose Gear Forging.Queue;
1703$       SEIZE,       2,VA:
                         ResourceNoseGearForge,1:NEXT(1702$);

1702$       DELAY:
                         MX(          TRIA(0.85*(10     *          *     ((varSlopeNoseGearForge     *     LOG(attSN)
)+log(varNoseGearForgeLN1))),(10      *          *      ((varSlopeNoseGearForge      *      LOG(attSN)
)+log(varNoseGearForgeLN1))),1.15*(10      *          *      ((varSlopeNoseGearForge      *      LOG(attSN)
)+log(varNoseGearForgeLN1))))                                                                          ,
TRIA(0.85*varNoseGearForgeLN100,varNoseGearForgeLN100,1.15*varNoseGearForgeLN100)),,
                         VA:NEXT(1745$);
```

```
1745$       ASSIGN:      Nose Gear Forging.WaitTime=Nose Gear Forging.WaitTime + Diff.WaitTime;
1709$       TALLY:       Nose Gear Forging.WaitTimePerEntity,Diff.WaitTime,1;
1711$       TALLY:       Nose Gear Forging.TotalTimePerEntity,Diff.StartTime,1;
1735$       ASSIGN:      Nose Gear Forging.VATime=Nose Gear Forging.VATime + Diff.VATime;
1736$       TALLY:       Nose Gear Forging.VATimePerEntity,Diff.VATime,1;
1701$       RELEASE:     ResourceNoseGearForge,1;
1750$       STACK,       1:Destroy:NEXT(1749$);

1749$       ASSIGN:      Nose Gear Forging.NumberOut=Nose Gear Forging.NumberOut + 1:
                         Nose Gear Forging.WIP=Nose Gear Forging.WIP-1:NEXT(7$);


;
;
;   Model statements for module:  BasicProcess.Process 8 (Nose Gear Machining)
;
7$          ASSIGN:      Nose Gear Machining.NumberIn=Nose Gear Machining.NumberIn + 1:
                         Nose Gear Machining.WIP=Nose Gear Machining.WIP+1;
1781$       STACK,       1:Save:NEXT(1755$);

1755$       QUEUE,       Nose Gear Machining.Queue;
1754$       SEIZE,       2,VA:
                         ResourceNoseGearMach,1:NEXT(1753$);

1753$       DELAY:
                         MX(      TRIA(0.85*(10      *          *      ((varSlopeNoseGearMach      *      LOG(attSN)
)+log(varNoseGearMachLN1))),(10      *          *      ((varSlopeNoseGearMach      *      LOG(attSN)
)+log(varNoseGearMachLN1))),1.15*(10      *          *      ((varSlopeNoseGearMach      *      LOG(attSN)
)+log(varNoseGearMachLN1))))                                                                    ,
TRIA(0.85*varNoseGearMachLN100,varNoseGearMachLN100,1.15*varNoseGearMachLN100)),,
                         VA:NEXT(1796$);

1796$       ASSIGN:      Nose Gear Machining.WaitTime=Nose Gear Machining.WaitTime + Diff.WaitTime;
1760$       TALLY:       Nose Gear Machining.WaitTimePerEntity,Diff.WaitTime,1;
1762$       TALLY:       Nose Gear Machining.TotalTimePerEntity,Diff.StartTime,1;
1786$       ASSIGN:      Nose Gear Machining.VATime=Nose Gear Machining.VATime + Diff.VATime;
1787$       TALLY:       Nose Gear Machining.VATimePerEntity,Diff.VATime,1;
1752$       RELEASE:     ResourceNoseGearMach,1;
1801$       STACK,       1:Destroy:NEXT(1800$);

1800$       ASSIGN:      Nose Gear Machining.NumberOut=Nose Gear Machining.NumberOut + 1:
                         Nose Gear Machining.WIP=Nose Gear Machining.WIP-1:NEXT(6$);


;
;
;   bring level of customization into here as individual variables
;
;   Model statements for module:  BasicProcess.Process 7 (Nose Gear Build Up)
;
6$          ASSIGN:      Nose Gear Build Up.NumberIn=Nose Gear Build Up.NumberIn + 1:
                         Nose Gear Build Up.WIP=Nose Gear Build Up.WIP+1;
1832$       STACK,       1:Save:NEXT(1806$);

1806$       QUEUE,       Nose Gear Build Up.Queue;
1805$       SEIZE,       2,VA:
```

```
                ResourceNoseGearBuildUp,1:NEXT(1804$);

1804$       DELAY:
                MX(     TRIA(0.85*     (10     *          *     ((varSlopeNoseGearBuildUp     *     LOG(attSN)
)+log(varNoseGearBuildUpLN1))),     (10     *          *     ((varSlopeNoseGearBuildUp     *     LOG(attSN)
)+log(varNoseGearBuildUpLN1))),1.15*     (10     *          *     ((varSlopeNoseGearBuildUp     *     LOG(attSN)
)+log(varNoseGearBuildUpLN1)))), TRIA(0.85* varNoseGearBuildUpLN100,varNoseGearBuildUpLN100,1.15*
varNoseGearBuildUpLN100)),,
                VA:NEXT(1847$);

1847$       ASSIGN:     Nose Gear Build Up.WaitTime=Nose Gear Build Up.WaitTime + Diff.WaitTime;
1811$       TALLY:      Nose Gear Build Up.WaitTimePerEntity,Diff.WaitTime,1;
1813$       TALLY:      Nose Gear Build Up.TotalTimePerEntity,Diff.StartTime,1;
1837$       ASSIGN:     Nose Gear Build Up.VATime=Nose Gear Build Up.VATime + Diff.VATime;
1838$       TALLY:      Nose Gear Build Up.VATimePerEntity,Diff.VATime,1;
1803$       RELEASE:    ResourceNoseGearBuildUp,1;
1852$       STACK,      1:Destroy:NEXT(1851$);

1851$       ASSIGN:     Nose Gear Build Up.NumberOut=Nose Gear Build Up.NumberOut + 1:
                Nose Gear Build Up.WIP=Nose Gear Build Up.WIP-1:NEXT(111$);


;
;
;   Model statements for module:  BasicProcess.Process 110 (Ship Nose Gear)
;
111$        ASSIGN:     Ship Nose Gear.NumberIn=Ship Nose Gear.NumberIn + 1:
                Ship Nose Gear.WIP=Ship Nose Gear.WIP+1;
1883$       STACK,      1:Save:NEXT(1857$);

1857$       QUEUE,      Ship Nose Gear.Queue;
1856$       SEIZE,      2,NVA:
                ResourceNoseGearShipper,1:NEXT(1855$);

1855$       DELAY:      5,,NVA:NEXT(1898$);

1898$       ASSIGN:     Ship Nose Gear.WaitTime=Ship Nose Gear.WaitTime + Diff.WaitTime;
1862$       TALLY:      Ship Nose Gear.WaitTimePerEntity,Diff.WaitTime,1;
1864$       TALLY:      Ship Nose Gear.TotalTimePerEntity,Diff.StartTime,1;
1888$       ASSIGN:     Ship Nose Gear.NVATime=Ship Nose Gear.NVATime + Diff.NVATime;
1889$       TALLY:      Ship Nose Gear.NVATimePerEntity,Diff.NVATime,1;
1854$       RELEASE:    ResourceNoseGearShipper,1;
1903$       STACK,      1:Destroy:NEXT(1902$);

1902$       ASSIGN:     Ship Nose Gear.NumberOut=Ship Nose Gear.NumberOut + 1:
                Ship Nose Gear.WIP=Ship Nose Gear.WIP-1:NEXT(45$);


;
;
;   Model statements for module:  BasicProcess.Create 7 (Sec44 Order)
;

1905$       CREATE,     1,DaysToBaseTime(0.0),Sec44:DaysToBaseTime(1),1:NEXT(1906$);

1906$       ASSIGN:     Sec44 Order.NumberOut=Sec44 Order.NumberOut + 1:NEXT(18$);
```

```
;
;
;   Model statements for module:  BasicProcess.Assign 6 (Assign CurveRate Sec44)
;
18$         ASSIGN:       varSlopeSec44AsmMinor=LOG(varCurveRate(76)/100)/LOG(2):
                          varSec44AsmMinorLN1=10        *              *        (LOG(varSec44AsmMinorLN100)       -
varSlopeSec44AsmMinor*LOG(100)):
                          varSlopeSec44AsmMajor=LOG(varCurveRate(83)/100)/LOG(2):
                          varSec44AsmMajorLN1=10        *              *        (LOG(varSec44AsmMajorLN100)       -
varSlopeSec44AsmMajor*LOG(100)):
                          varSlopeSec44Fab=LOG(varCurveRate(86)/100)/LOG(2):
                          varSec44FabLN1=10             *              *          (LOG(varSec44FabLN100)          -
varSlopeSec44Fab*LOG(100)):NEXT(113$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 5 (Read Sec44 Order Schedule)
;
113$        READ,         MDaySchedule787,RECORDSET(RecordsetSec44):
                          Sec44Time:NEXT(114$);


;
;
;   Model statements for module:  BasicProcess.Separate 5 (Separate Sec44)
;
114$        DUPLICATE,    100 - 50:
                          1,1911$,50:NEXT(1910$);

1910$               ASSIGN:             Separate Sec44.NumberOut Orig=Separate Sec44.NumberOut Orig +
1:NEXT(115$);

1911$               ASSIGN:             Separate Sec44.NumberOut Dup=Separate Sec44.NumberOut Dup +
1:NEXT(113$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 5 (Delay Sec44 Order)
;
115$        DELAY:        Sec44Time,,Other:NEXT(53$);


;
;
;   Model statements for module:  BasicProcess.Assign 18 (Assign Sec44 Serial Number)
;
53$         ASSIGN:       varPartSec44=varPartSec44 + 1:
                          attSN=varPartSec44:NEXT(116$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 5 (Route Sec44 Order)
;
```

```
116$        ROUTE:        0.,StnSec44Order;

;
;
;    Model statements for module:  BasicProcess.Create 8 (Sec46 Order)
;
1912$       CREATE,       1,DaysToBaseTime(0.0),Sec46:DaysToBaseTime(1),1:NEXT(1913$);

1913$       ASSIGN:       Sec46 Order.NumberOut=Sec46 Order.NumberOut + 1:NEXT(57$);


;
;
;    Model statements for module:  BasicProcess.Assign 19 (Assign CurveRate Sec46)
;
57$         ASSIGN:       varSlopeSec46AsmMinor=LOG(varCurveRate(85)/100)/LOG(2):
                          varSec46AsmMinorLN1=10        *             *        (LOG(varSec46AsmMinorLN100)       -
varSlopeSec46AsmMinor*LOG(100)):
                          varSlopeSec46AsmMajor=LOG(varCurveRate(84)/100)/LOG(2):
                          varSec46AsmMajorLN1=10        *             *        (LOG(varSec46AsmMajorLN100)       -
varSlopeSec46AsmMajor*LOG(100)):
                          varSlopeSec46Fab=LOG(varCurveRate(88)/100)/LOG(2):
                          varSec46FabLN1=10             *             *          (LOG(varSec46FabLN100)          -
varSlopeSec46Fab*LOG(100)):NEXT(118$);


;
;
;    Model statements for module:  AdvancedProcess.ReadWrite 6 (Read Sec46 Order Schedule)
;
118$        READ,         MDaySchedule787,RECORDSET(RecordsetSec46):
                          Sec46Time:NEXT(119$);


;
;
;    Model statements for module:  BasicProcess.Separate 6 (Separate Sec46)
;
119$        DUPLICATE,    100 - 50:
                          1,1918$,50:NEXT(1917$);

1917$       ASSIGN:       Separate Sec46.NumberOut Orig=Separate Sec46.NumberOut Orig +
1:NEXT(120$);

1918$       ASSIGN:       Separate Sec46.NumberOut Dup=Separate Sec46.NumberOut Dup +
1:NEXT(118$);


;
;
;    Model statements for module:  AdvancedProcess.Delay 6 (Delay Sec46 Order)
;
120$        DELAY:        Sec46Time,,Other:NEXT(58$);
```

```
;
;
;   Model statements for module:  BasicProcess.Assign 20 (Assign Sec46 Serial Number)
;
58$         ASSIGN:       varPartSec46=varPartSec46 + 1:
                          attSN=varPartSec46:NEXT(121$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 6 (Route Sec46 Order)
;
121$        ROUTE:        0.,StnSec46Order;


;
;
;   Model statements for module:  BasicProcess.Create 9 (Sec11 Order)
;

1919$       CREATE,       1,DaysToBaseTime(0.0),Sec11:DaysToBaseTime(1),1:NEXT(1920$);

1920$       ASSIGN:       Sec11 Order.NumberOut=Sec11 Order.NumberOut + 1:NEXT(62$);


;
;
;   Model statements for module:  BasicProcess.Assign 21 (Assign CurveRate Sec11)
;
62$         ASSIGN:       varSlopeSec11AsmMinor=LOG(varCurveRate(70)/100)/LOG(2):
                          varSec11AsmMinorLN1=10        *             *        (LOG(varSec11AsmMinorLN100)       -
varSlopeSec11AsmMinor*LOG(100)):
                          varSlopeSec11AsmMajor=LOG(varCurveRate(71)/100)/LOG(2):
                          varSec11AsmMajorLN1=10        *             *        (LOG(varSec11AsmMajorLN100)       -
varSlopeSec11AsmMajor*LOG(100)):
                          varSlopeSec11Fab=LOG(varCurveRate(92)/100)/LOG(2):
                          varSec11FabLN1=10 *  * (LOG(varSec11FabLN100) - varSlopeSec11Fab*LOG(100)):
                          varSlopeSec1145Join=LOG(varCurveRate(77)/100)/LOG(2):
                          varSec1145JoinLN1=10          *             *         (LOG(varSec1145JoinLN100)        -
varSlopeSec1145Join*LOG(100)):NEXT(123$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 7 (Read Sec11 Order Schedule)
;
123$        READ,         MDaySchedule787,RECORDSET(RecordsetSec11):
                          Sec11Time:NEXT(126$);


;
;
;   Model statements for module:  BasicProcess.Separate 7 (Separate Sec11)
;
126$        DUPLICATE,    100 - 50:
                          1,1925$,50:NEXT(1924$);
```

```
1924$           ASSIGN:         Separate Sec11.NumberOut Orig=Separate Sec11.NumberOut Orig +
1:NEXT(129$);

1925$           ASSIGN:         Separate Sec11.NumberOut Dup=Separate Sec11.NumberOut Dup +
1:NEXT(123$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 7 (Delay Sec11 Order)
;
129$        DELAY:      Sec11Time,,Other:NEXT(63$);


;
;
;   Model statements for module:  BasicProcess.Assign 22 (Assign Sec11 Serial Number)
;
63$         ASSIGN:     varPartSec11=varPartSec11 + 1:
                        attSN=varPartSec11:NEXT(132$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 7 (Route Sec11 Order)
;
132$        ROUTE:      0.,StnSec11Order;


;
;
;   Model statements for module:  BasicProcess.Create 10 (Sec45 Order)
;

1926$       CREATE,     1,DaysToBaseTime(0.0),Sec45:DaysToBaseTime(1),1:NEXT(1927$);

1927$       ASSIGN:     Sec45 Order.NumberOut=Sec45 Order.NumberOut + 1:NEXT(69$);


;
;
;   Model statements for module:  BasicProcess.Assign 23 (Assign CurveRate Sec45)
;
69$         ASSIGN:     varSlopeSec45AsmMinor=LOG(varCurveRate(80)/100)/LOG(2):
                        varSec45AsmMinorLN1=10      *           *       (LOG(varSec45AsmMinorLN100)     -
varSlopeSec45AsmMinor*LOG(100)):
                        varSlopeSec45Fab=LOG(varCurveRate(90)/100)/LOG(2):
                        varSec45FabLN1=10           *           *       (LOG(varSec45FabLN100)          -
varSlopeSec45Fab*LOG(100)):NEXT(124$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 8 (Read Sec45 Order Schedule)
;
124$        READ,       MDaySchedule787,RECORDSET(RecordsetSec45):
                        Sec45Time:NEXT(127$);
```

```
;
;
;   Model statements for module:  BasicProcess.Separate 8 (Separate Sec45)
;
127$        DUPLICATE,    100 - 50:
                1,1932$,50:NEXT(1931$);

1931$             ASSIGN:          Separate Sec45.NumberOut Orig=Separate Sec45.NumberOut Orig +
1:NEXT(130$);

1932$             ASSIGN:          Separate Sec45.NumberOut Dup=Separate Sec45.NumberOut Dup +
1:NEXT(124$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 8 (Delay Sec45 Order)
;
130$        DELAY:       Sec45Time,,Other:NEXT(70$);


;
;
;   Model statements for module:  BasicProcess.Assign 24 (Assign Sec45 Serial Number)
;
70$         ASSIGN:      varPartSec45=varPartSec45 + 1:
                attSN=varPartSec45:NEXT(133$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 8 (Route Sec45 Order)
;
133$        ROUTE:       0.,StnSec45Order;


;
;
;   Model statements for module:  BasicProcess.Create 11 (Sec49 Order)
;

1933$       CREATE,      1,DaysToBaseTime(0.0),Sec49:DaysToBaseTime(1),1:NEXT(1934$);

1934$       ASSIGN:      Sec49 Order.NumberOut=Sec49 Order.NumberOut + 1:NEXT(73$);


;
;
;   Model statements for module:  BasicProcess.Assign 25 (Assign CurveRate Sec49)
;
73$         ASSIGN:      varSlopeSec49AsmMinor=LOG(varCurveRate(83)/100)/LOG(2):
                varSec49AsmMinorLN1=10       *            *       (LOG(varSec49AsmMinorLN100)       -
varSlopeSec49AsmMinor*LOG(100)):
                varSlopeSec49AsmMajor=LOG(varCurveRate(81)/100)/LOG(2):
```

```
                varSec49AsmMajorLN1=10      *          *      (LOG(varSec49AsmMajorLN100)      -
varSlopeSec49AsmMajor*LOG(100)):
                varSlopeSec49Fab=LOG(varCurveRate(90)/100)/LOG(2):
                varSec49FabLN1=10      *          *      (LOG(varSec49FabLN100)      -
varSlopeSec49Fab*LOG(100)):NEXT(125$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 9 (Read Sec49 Order Schedule)
;
125$      READ,      MDaySchedule787,RECORDSET(RecordsetSec49LessMLGD):
                Sec49Time:NEXT(128$);


;
;
;   Model statements for module:  BasicProcess.Separate 9 (Separate Sec49)
;
128$      DUPLICATE,    100 - 50:
                1,1939$,50:NEXT(1938$);

1938$          ASSIGN:          Separate Sec49.NumberOut Orig=Separate Sec49.NumberOut Orig +
1:NEXT(131$);

1939$          ASSIGN:          Separate Sec49.NumberOut Dup=Separate Sec49.NumberOut Dup +
1:NEXT(125$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 9 (Delay Sec49 Order)
;
131$      DELAY:      Sec49Time,,Other:NEXT(74$);


;
;
;   Model statements for module:  BasicProcess.Assign 26 (Assign Sec49 Serial Number)
;
74$       ASSIGN:     varPartSec49=varPartSec49 + 1:
                attSN=varPartSec49:NEXT(134$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 9 (Route Sec49 Order)
;
134$      ROUTE:      0.,StnSec49Order;


;
;
;   Model statements for module:  BasicProcess.Create 12 (Sec47 Order)
;

1940$      CREATE,      1,DaysToBaseTime(0.0),Sec47:DaysToBaseTime(1),1:NEXT(1941$);
```

```
1941$       ASSIGN:       Sec47 Order.NumberOut=Sec47 Order.NumberOut + 1:NEXT(80$);


;
;
;   Model statements for module:  BasicProcess.Assign 27 (Assign CurveRate Sec47)
;
80$         ASSIGN:       varSlopeSec47AsmMajor=LOG(varCurveRate(81)/100)/LOG(2):
                          varSec47AsmMajorLN1=10        *             *        (LOG(varSec47AsmMajorLN100)        -
varSlopeSec47AsmMajor*LOG(100)):
                          varSlopeSec47Fab=LOG(varCurveRate(82)/100)/LOG(2):
                          varSec47FabLN1=10 *  * (LOG(varSec47FabLN100) - varSlopeSec47Fab*LOG(100)):
                          varSlopeSec4748Join=LOG(varCurveRate(87)/100)/LOG(2):
                          varSec4748JoinLN1=10          *             *        (LOG(varSec4748JoinLN100)          -
varSlopeSec4748Join*LOG(100)):NEXT(138$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 10 (Read Sec47 Order Schedule)
;
138$        READ,         MDaySchedule787,RECORDSET(RecordsetSec47):
                          Sec47Time:NEXT(140$);


;
;
;   Model statements for module:  BasicProcess.Separate 10 (Separate Sec47)
;
140$        DUPLICATE,    100 - 50:
                          1,1946$,50:NEXT(1945$);

1945$       ASSIGN:       Separate Sec47.NumberOut Orig=Separate Sec47.NumberOut Orig +
1:NEXT(142$);

1946$       ASSIGN:       Separate Sec47.NumberOut Dup=Separate Sec47.NumberOut Dup +
1:NEXT(138$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 10 (Delay Sec47 Order)
;
142$        DELAY:        Sec47Time,,Other:NEXT(81$);


;
;
;   Model statements for module:  BasicProcess.Assign 28 (Assign Sec47 Serial Number)
;
81$         ASSIGN:       varPartSec47=varPartSec47 + 1:
                          attSN=varPartSec47:NEXT(144$);


;
;
```

```
;   Model statements for module:  AdvancedTransfer.Route 10 (Route Sec47 Order)
;
144$        ROUTE:        0.,StnSec47Order;


;
;
;   Model statements for module:  BasicProcess.Create 13 (Sec48 Fwd Order)
;
1947$       CREATE,       1,DaysToBaseTime(0.0),Sec48Fwd:DaysToBaseTime(1),1:NEXT(1948$);

1948$       ASSIGN:       Sec48 Fwd Order.NumberOut=Sec48 Fwd Order.NumberOut + 1:NEXT(86$);


;
;
;   Model statements for module:  BasicProcess.Assign 29 (Assign CurveRate Sec48 Fwd)
;
86$         ASSIGN:       varSlopeSec48FwdAsmMajor=LOG(varCurveRate(80)/100)/LOG(2):
                          varSec48FwdAsmMajorLN1=10     *          *     (LOG(varSec48FwdAsmMajorLN100)     -
varSlopeSec48FwdAsmMajor*LOG(100)):
                          varSlopeSec48FwdFab=LOG(varCurveRate(84)/100)/LOG(2):
                          varSec48FwdFabLN1=10          *          *     (LOG(varSec48FwdFabLN100)          -
varSlopeSec48FwdFab*LOG(100)):NEXT(139$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 11 (Read Sec48 Order Schedule)
;
139$        READ,         MDaySchedule787,RECORDSET(RecordsetSec48FWD):
                          Sec48FwdTime:NEXT(141$);


;
;
;   Model statements for module:  BasicProcess.Separate 11 (Separate Sec48)
;
141$        DUPLICATE,    100 - 50:
                          1,1953$,50:NEXT(1952$);

1952$                     ASSIGN:             Separate Sec48.NumberOut Orig=Separate Sec48.NumberOut Orig +
1:NEXT(143$);

1953$                     ASSIGN:             Separate Sec48.NumberOut Dup=Separate Sec48.NumberOut Dup +
1:NEXT(139$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 11 (Delay Sec48Fwd Order)
;
143$        DELAY:        Sec48FwdTime,,Other:NEXT(87$);


;
```

```
;
;   Model statements for module:  BasicProcess.Assign 30 (Assign Sec48 Fwd Serial Number)
;
87$         ASSIGN:       varPartSec48Fwd=varPartSec48Fwd + 1:
                          attSN=varPartSec48Fwd:NEXT(145$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 11 (Route Sec48 Fwd Order)
;
145$        ROUTE:        0.,StnSec48FwdOrder;


;
;
;   Model statements for module:  BasicProcess.Create 14 (Horizontal Stabilizer Order)
;

1954$       CREATE,       1,DaysToBaseTime(0.0),HStabilizer:DaysToBaseTime(1),1:NEXT(1955$);

1955$       ASSIGN:       Horizontal Stabilizer Order.NumberOut=Horizontal Stabilizer Order.NumberOut +
1:NEXT(90$);


;
;
;   Model statements for module:  BasicProcess.Assign 31 (Assign CurveRate Horizontal Stabilizer)
;
90$         ASSIGN:       varSlopeHStbzFab=LOG(varCurveRate(86)/100)/LOG(2):
                          varHStbzFabLN1=10 * * (LOG(varHStbzFabLN100) - varSlopeHStbzFab*LOG(100)):
                          varSlopeHStbzAsmMinor=LOG(varCurveRate(85)/100)/LOG(2):
                          varHStbzAsmMinorLN1=10      *           *      (LOG(varHStbzAsmMinorLN100)      -
varSlopeHStbzAsmMinor*LOG(100)):
                          varSlopeHStbzAsmMajor=LOG(varCurveRate(87)/100)/LOG(2):
                          varHStbzAsmMajorLN1=10      *           *      (LOG(varHStbzAsmMajorLN100)      -
varSlopeHStbzAsmMajor*LOG(100)):
                          varSlopeHStbzJoin=LOG(varCurveRate(78)/100)/LOG(2):
                          varHStbzJoinLN1=10          *           *          (LOG(varHStbzJoinLN100)          -
varSlopeHStbzJoin*LOG(100)):NEXT(148$);


;
;
;       Model statements for module:  AdvancedProcess.ReadWrite 12 (Read Horizontal Stabilizer Order
Schedule)
;
148$        READ,         MDaySchedule787,RECORDSET(RecordsetHortzStab):
                          HortzStabTime:NEXT(149$);


;
;
;   Model statements for module:  BasicProcess.Separate 12 (Separate Hortz Stab)
;
149$        DUPLICATE,    100 - 50:
                          1,1960$,50:NEXT(1959$);
```

```
1959$        ASSIGN:        Separate Hortz Stab.NumberOut Orig=Separate Hortz Stab.NumberOut Orig +
1:NEXT(150$);

1960$        ASSIGN:        Separate Hortz Stab.NumberOut Dup=Separate Hortz Stab.NumberOut Dup +
1:NEXT(148$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 12 (Delay Hortz Stab Order)
;
150$        DELAY:        HortzStabTime,,Other:NEXT(91$);


;
;
;   Model statements for module:  BasicProcess.Assign 32 (Assign Horizontal Stabilizer Serial Number)
;
91$         ASSIGN:       varPartHStbz=varPartHStbz + 1:
                          attSN=varPartHStbz:NEXT(151$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 12 (Route Hortz Stab Order)
;
151$        ROUTE:        0.,StnHortzStabOrder;


;
;
;   Model statements for module:  BasicProcess.Create 15 (Sec48 Aft Order)
;

1961$       CREATE,       1,DaysToBaseTime(0.0),Sec48Aft:DaysToBaseTime(1),1:NEXT(1962$);

1962$       ASSIGN:       Sec48 Aft Order.NumberOut=Sec48 Aft Order.NumberOut + 1:NEXT(98$);


;
;
;   Model statements for module:  BasicProcess.Assign 33 (Assign CurveRate Sec48 Aft)
;
98$         ASSIGN:       varSlopeSec48AftAsmMajor=LOG(varCurveRate(83)/100)/LOG(2):
                          varSec48AftAsmMajorLN1=10       *       *       (LOG(varSec48AftAsmMajorLN100)       -
varSlopeSec48AftAsmMajor*LOG(100)):
                          varSlopeSec48AftFab=LOG(varCurveRate(85)/100)/LOG(2):
                          varSec48AftFabLN1=10       *       *       (LOG(varSec48AftFabLN100)       -
varSlopeSec48AftFab*LOG(100)):NEXT(153$);


;
;
;   Model statements for module:  AdvancedProcess.ReadWrite 13 (Read Sec48 Aft Order Schedule)
;
153$        READ,         MDaySchedule787,RECORDSET(RecordsetSec48Aft):
```

```
                Sec48AftTime:NEXT(154$);

;
;
;   Model statements for module:  BasicProcess.Separate 13 (Separate Sec48 Aft)
;
154$        DUPLICATE,    100 - 50:
                1,1967$,50:NEXT(1966$);

1966$           ASSIGN:         Separate Sec48 Aft.NumberOut Orig=Separate Sec48 Aft.NumberOut Orig +
1:NEXT(155$);

1967$           ASSIGN:         Separate Sec48 Aft.NumberOut Dup=Separate Sec48 Aft.NumberOut Dup +
1:NEXT(153$);


;
;
;   Model statements for module:  AdvancedProcess.Delay 13 (Delay Sec48 Aft Order)
;
155$        DELAY:      Sec48AftTime,,Other:NEXT(99$);


;
;
;   Model statements for module:  BasicProcess.Assign 34 (Assign Sec48 Aft Serial Number)
;
99$         ASSIGN:     varPartSec48Aft=varPartSec48Aft + 1:
                attSN=varPartSec48Aft:NEXT(156$);


;
;
;   Model statements for module:  AdvancedTransfer.Route 13 (Route Sec 48 Aft Order)
;
156$        ROUTE:      0.,StnSec48AftOrder;


;
;
;   Model statements for module:  AdvancedTransfer.Station 5 (Sec44 Order Arrival)
;

117$        STATION,    StnSec44Order;
1970$       DELAY:      0.0,,VA:NEXT(54$);


;
;
;   Model statements for module:  BasicProcess.Process 17 (Sec 44 Fab)
;
54$         ASSIGN:     Sec 44 Fab.NumberIn=Sec 44 Fab.NumberIn + 1:
                Sec 44 Fab.WIP=Sec 44 Fab.WIP+1;
2000$       STACK,      1:Save:NEXT(1974$);

1974$       QUEUE,      Sec 44 Fab.Queue;
```

```
1973$       SEIZE,       2,VA:
                ResourceSec44Fab,1:NEXT(1972$);

1972$       DELAY:
                MX( TRIA(0.85* (10 *  * ((varSlopeSec44Fab * LOG(attSN) )+log(varSec44FabLN1))), (10 *
* ((varSlopeSec44Fab  * LOG(attSN)  )+log(varSec44FabLN1))),1.15* (10 *    * ((varSlopeSec44Fab  *
LOG(attSN)  )+log(varSec44FabLN1))))  ,  TRIA(0.85*  varSec44FabLN100,varSec44FabLN100,1.15*
varSec44FabLN100)),,
                VA:NEXT(2015$);

2015$       ASSIGN:      Sec 44 Fab.WaitTime=Sec 44 Fab.WaitTime + Diff.WaitTime;
1979$       TALLY:       Sec 44 Fab.WaitTimePerEntity,Diff.WaitTime,1;
1981$       TALLY:       Sec 44 Fab.TotalTimePerEntity,Diff.StartTime,1;
2005$       ASSIGN:      Sec 44 Fab.VATime=Sec 44 Fab.VATime + Diff.VATime;
2006$       TALLY:       Sec 44 Fab.VATimePerEntity,Diff.VATime,1;
1971$       RELEASE:     ResourceSec44Fab,1;
2020$       STACK,       1:Destroy:NEXT(2019$);

2019$       ASSIGN:      Sec 44 Fab.NumberOut=Sec 44 Fab.NumberOut + 1:
                Sec 44 Fab.WIP=Sec 44 Fab.WIP-1:NEXT(55$);


;
;
;   Model statements for module:  BasicProcess.Process 18 (Sec 44 Asm Minor)
;
55$         ASSIGN:      Sec 44 Asm Minor.NumberIn=Sec 44 Asm Minor.NumberIn + 1:
                Sec 44 Asm Minor.WIP=Sec 44 Asm Minor.WIP+1;
2051$       STACK,       1:Save:NEXT(2025$);

2025$       QUEUE,       Sec 44 Asm Minor.Queue;
2024$       SEIZE,       2,VA:
                ResourceSec44AsmMinor,1:NEXT(2023$);

2023$       DELAY:
                MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec44AsmMinor     *     LOG(attSN)
)+log(varSec44AsmMinorLN1))),      (10      *          *      ((varSlopeSec44AsmMinor      *      LOG(attSN)
)+log(varSec44AsmMinorLN1))),1.15*     (10     *          *     ((varSlopeSec44AsmMinor     *     LOG(attSN)
)+log(varSec44AsmMinorLN1))))  ,  TRIA(0.85*  varSec44AsmMinorLN100,varSec44AsmMinorLN100,1.15*
varSec44AsmMinorLN100)),,
                VA:NEXT(2066$);

2066$       ASSIGN:      Sec 44 Asm Minor.WaitTime=Sec 44 Asm Minor.WaitTime + Diff.WaitTime;
2030$       TALLY:       Sec 44 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
2032$       TALLY:       Sec 44 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
2056$       ASSIGN:      Sec 44 Asm Minor.VATime=Sec 44 Asm Minor.VATime + Diff.VATime;
2057$       TALLY:       Sec 44 Asm Minor.VATimePerEntity,Diff.VATime,1;
2022$       RELEASE:     ResourceSec44AsmMinor,1;
2071$       STACK,       1:Destroy:NEXT(2070$);

2070$       ASSIGN:      Sec 44 Asm Minor.NumberOut=Sec 44 Asm Minor.NumberOut + 1:
                Sec 44 Asm Minor.WIP=Sec 44 Asm Minor.WIP-1:NEXT(56$);


;
;
;   Model statements for module:  BasicProcess.Process 19 (Sec 44 Asm Major)
```

```
;
56$         ASSIGN:      Sec 44 Asm Major.NumberIn=Sec 44 Asm Major.NumberIn + 1:
                         Sec 44 Asm Major.WIP=Sec 44 Asm Major.WIP+1;
2102$       STACK,       1:Save:NEXT(2076$);

2076$       QUEUE,       Sec 44 Asm Major.Queue;
2075$       SEIZE,       2,VA:
                         ResourceSec44AsmMajor,1:NEXT(2074$);

2074$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec44AsmMajor     *     LOG(attSN)
)+log(varSec44AsmMajorLN1))),      (10      *          *     ((varSlopeSec44AsmMajor      *     LOG(attSN)
)+log(varSec44AsmMajorLN1))),1.15*     (10     *          *     ((varSlopeSec44AsmMajor     *     LOG(attSN)
)+log(varSec44AsmMajorLN1))))  ,  TRIA(0.85*  varSec44AsmMajorLN100,varSec44AsmMajorLN100,1.15*
varSec44AsmMajorLN100)),,
                         VA:NEXT(2117$);

2117$       ASSIGN:      Sec 44 Asm Major.WaitTime=Sec 44 Asm Major.WaitTime + Diff.WaitTime;
2081$       TALLY:       Sec 44 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
2083$       TALLY:       Sec 44 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
2107$       ASSIGN:      Sec 44 Asm Major.VATime=Sec 44 Asm Major.VATime + Diff.VATime;
2108$       TALLY:       Sec 44 Asm Major.VATimePerEntity,Diff.VATime,1;
2073$       RELEASE:     ResourceSec44AsmMajor,1;
2122$       STACK,       1:Destroy:NEXT(2121$);

2121$       ASSIGN:      Sec 44 Asm Major.NumberOut=Sec 44 Asm Major.NumberOut + 1:
                         Sec 44 Asm Major.WIP=Sec 44 Asm Major.WIP-1:NEXT(103$);


;
;
;   Model statements for module:  BasicProcess.Process 102 (Ship Sec 44)
;
103$        ASSIGN:      Ship Sec 44.NumberIn=Ship Sec 44.NumberIn + 1:
                         Ship Sec 44.WIP=Ship Sec 44.WIP+1;
2153$       STACK,       1:Save:NEXT(2127$);

2127$       QUEUE,       Ship Sec 44.Queue;
2126$       SEIZE,       2,NVA:
                         ResourceSec44Shipper,1:NEXT(2125$);

2125$       DELAY:       3,,NVA:NEXT(2168$);

2168$       ASSIGN:      Ship Sec 44.WaitTime=Ship Sec 44.WaitTime + Diff.WaitTime;
2132$       TALLY:       Ship Sec 44.WaitTimePerEntity,Diff.WaitTime,1;
2134$       TALLY:       Ship Sec 44.TotalTimePerEntity,Diff.StartTime,1;
2158$       ASSIGN:      Ship Sec 44.NVATime=Ship Sec 44.NVATime + Diff.NVATime;
2159$       TALLY:       Ship Sec 44.NVATimePerEntity,Diff.NVATime,1;
2124$       RELEASE:     ResourceSec44Shipper,1;
2173$       STACK,       1:Destroy:NEXT(2172$);

2172$       ASSIGN:      Ship Sec 44.NumberOut=Ship Sec 44.NumberOut + 1:
                         Ship Sec 44.WIP=Ship Sec 44.WIP-1:NEXT(20$);


;
;
```

```
;    Model statements for module:  AdvancedTransfer.Station 6 (Sec46 Order Arrival)
;

122$        STATION,     StnSec46Order;
2177$       DELAY:       0.0,,VA:NEXT(59$);


;
;
;    Model statements for module:  BasicProcess.Process 20 (Sec 46 Fab)
;
59$         ASSIGN:      Sec 46 Fab.NumberIn=Sec 46 Fab.NumberIn + 1:
                         Sec 46 Fab.WIP=Sec 46 Fab.WIP+1;
2207$       STACK,       1:Save:NEXT(2181$);

2181$       QUEUE,       Sec 46 Fab.Queue;
2180$       SEIZE,       2,VA:
                         ResourceSec46Fab,1:NEXT(2179$);

2179$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec46Fab * LOG(attSN) )+log(varSec46FabLN1))), (10 *
*  ((varSlopeSec46Fab  *  LOG(attSN)  )+log(varSec46FabLN1))),1.15*  (10  *    *  ((varSlopeSec46Fab  *
LOG(attSN)     )+log(varSec46FabLN1))))),     TRIA(0.85*     varSec46FabLN100,varSec46FabLN100,1.15*
varSec46FabLN100)),,
                         VA:NEXT(2222$);

2222$       ASSIGN:      Sec 46 Fab.WaitTime=Sec 46 Fab.WaitTime + Diff.WaitTime;
2186$       TALLY:       Sec 46 Fab.WaitTimePerEntity,Diff.WaitTime,1;
2188$       TALLY:       Sec 46 Fab.TotalTimePerEntity,Diff.StartTime,1;
2212$       ASSIGN:      Sec 46 Fab.VATime=Sec 46 Fab.VATime + Diff.VATime;
2213$       TALLY:       Sec 46 Fab.VATimePerEntity,Diff.VATime,1;
2178$       RELEASE:     ResourceSec46Fab,1;
2227$       STACK,       1:Destroy:NEXT(2226$);

2226$       ASSIGN:      Sec 46 Fab.NumberOut=Sec 46 Fab.NumberOut + 1:
                         Sec 46 Fab.WIP=Sec 46 Fab.WIP-1:NEXT(60$);


;
;
;    Model statements for module:  BasicProcess.Process 21 (Sec 46 Asm Minor)
;
60$         ASSIGN:      Sec 46 Asm Minor.NumberIn=Sec 46 Asm Minor.NumberIn + 1:
                         Sec 46 Asm Minor.WIP=Sec 46 Asm Minor.WIP+1;
2258$       STACK,       1:Save:NEXT(2232$);

2232$       QUEUE,       Sec 46 Asm Minor.Queue;
2231$       SEIZE,       2,VA:
                         ResourceSec46AsmMinor,1:NEXT(2230$);

2230$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec46AsmMinor     *     LOG(attSN)
)+log(varSec46AsmMinorLN1))),     (10     *          *     ((varSlopeSec46AsmMinor     *     LOG(attSN)
)+log(varSec46AsmMinorLN1))),1.15*     (10     *          *     ((varSlopeSec46AsmMinor     *     LOG(attSN)
)+log(varSec46AsmMinorLN1)))),     TRIA(0.85*     varSec46AsmMinorLN100,varSec46AsmMinorLN100,1.15*
varSec46AsmMinorLN100)),,
                         VA:NEXT(2273$);
```

```
2273$       ASSIGN:      Sec 46 Asm Minor.WaitTime=Sec 46 Asm Minor.WaitTime + Diff.WaitTime;
2237$       TALLY:       Sec 46 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
2239$       TALLY:       Sec 46 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
2263$       ASSIGN:      Sec 46 Asm Minor.VATime=Sec 46 Asm Minor.VATime + Diff.VATime;
2264$       TALLY:       Sec 46 Asm Minor.VATimePerEntity,Diff.VATime,1;
2229$       RELEASE:     ResourceSec46AsmMinor,1;
2278$       STACK,       1:Destroy:NEXT(2277$);

2277$       ASSIGN:      Sec 46 Asm Minor.NumberOut=Sec 46 Asm Minor.NumberOut + 1:
                         Sec 46 Asm Minor.WIP=Sec 46 Asm Minor.WIP-1:NEXT(61$);


;
;
;   Model statements for module:  BasicProcess.Process 22 (Sec 46 Asm Major)
;
61$         ASSIGN:      Sec 46 Asm Major.NumberIn=Sec 46 Asm Major.NumberIn + 1:
                         Sec 46 Asm Major.WIP=Sec 46 Asm Major.WIP+1;
2309$       STACK,       1:Save:NEXT(2283$);

2283$       QUEUE,       Sec 46 Asm Major.Queue;
2282$       SEIZE,       2,VA:
                         ResourceSec46AsmMajor,1:NEXT(2281$);

2281$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec46AsmMajor     *     LOG(attSN)
)+log(varSec46AsmMajorLN1))),      (10      *          *     ((varSlopeSec46AsmMajor     *     LOG(attSN)
)+log(varSec46AsmMajorLN1))),1.15*     (10     *          *     ((varSlopeSec46AsmMajor     *     LOG(attSN)
)+log(varSec46AsmMajorLN1)))),     TRIA(0.85*     varSec46AsmMajorLN100,varSec46AsmMajorLN100,1.15*
varSec46AsmMajorLN100)),,
                         VA:NEXT(2324$);

2324$       ASSIGN:      Sec 46 Asm Major.WaitTime=Sec 46 Asm Major.WaitTime + Diff.WaitTime;
2288$       TALLY:       Sec 46 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
2290$       TALLY:       Sec 46 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
2314$       ASSIGN:      Sec 46 Asm Major.VATime=Sec 46 Asm Major.VATime + Diff.VATime;
2315$       TALLY:       Sec 46 Asm Major.VATimePerEntity,Diff.VATime,1;
2280$       RELEASE:     ResourceSec46AsmMajor,1;
2329$       STACK,       1:Destroy:NEXT(2328$);

2328$       ASSIGN:      Sec 46 Asm Major.NumberOut=Sec 46 Asm Major.NumberOut + 1:
                         Sec 46 Asm Major.WIP=Sec 46 Asm Major.WIP-1:NEXT(104$);


;
;
;   Model statements for module:  BasicProcess.Process 103 (Ship Sec 46)
;
104$        ASSIGN:      Ship Sec 46.NumberIn=Ship Sec 46.NumberIn + 1:
                         Ship Sec 46.WIP=Ship Sec 46.WIP+1;
2360$       STACK,       1:Save:NEXT(2334$);

2334$       QUEUE,       Ship Sec 46.Queue;
2333$       SEIZE,       2,NVA:
                         ResourceSec46Shipper,1:NEXT(2332$);
```

```
2332$       DELAY:       3,,NVA:NEXT(2375$);

2375$       ASSIGN:      Ship Sec 46.WaitTime=Ship Sec 46.WaitTime + Diff.WaitTime;
2339$       TALLY:       Ship Sec 46.WaitTimePerEntity,Diff.WaitTime,1;
2341$       TALLY:       Ship Sec 46.TotalTimePerEntity,Diff.StartTime,1;
2365$       ASSIGN:      Ship Sec 46.NVATime=Ship Sec 46.NVATime + Diff.NVATime;
2366$       TALLY:       Ship Sec 46.NVATimePerEntity,Diff.NVATime,1;
2331$       RELEASE:     ResourceSec46Shipper,1;
2380$       STACK,       1:Destroy:NEXT(2379$);

2379$       ASSIGN:      Ship Sec 46.NumberOut=Ship Sec 46.NumberOut + 1:
                         Ship Sec 46.WIP=Ship Sec 46.WIP-1:NEXT(20$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 7 (Sec11 Order Arrival)
;

135$        STATION,     StnSec11Order;
2384$       DELAY:       0.0,,VA:NEXT(64$);


;
;
;   Model statements for module:  BasicProcess.Process 23 (Sec 11 Fab)
;
64$         ASSIGN:      Sec 11 Fab.NumberIn=Sec 11 Fab.NumberIn + 1:
                         Sec 11 Fab.WIP=Sec 11 Fab.WIP+1;
2414$       STACK,       1:Save:NEXT(2388$);

2388$       QUEUE,       Sec 11 Fab.Queue;
2387$       SEIZE,       2,VA:
                         ResourceSec11Fab,1:NEXT(2386$);

2386$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec11Fab * LOG(attSN) )+log(varSec11FabLN1))), (10 *
* ((varSlopeSec11Fab  *  LOG(attSN)  )+log(varSec11FabLN1))),1.15*  (10  *    * ((varSlopeSec11Fab  *
LOG(attSN)     )+log(varSec11FabLN1))))),     TRIA(0.85*     varSec11FabLN100,varSec11FabLN100,1.15*
varSec11FabLN100)),,
                         VA:NEXT(2429$);

2429$       ASSIGN:      Sec 11 Fab.WaitTime=Sec 11 Fab.WaitTime + Diff.WaitTime;
2393$       TALLY:       Sec 11 Fab.WaitTimePerEntity,Diff.WaitTime,1;
2395$       TALLY:       Sec 11 Fab.TotalTimePerEntity,Diff.StartTime,1;
2419$       ASSIGN:      Sec 11 Fab.VATime=Sec 11 Fab.VATime + Diff.VATime;
2420$       TALLY:       Sec 11 Fab.VATimePerEntity,Diff.VATime,1;
2385$       RELEASE:     ResourceSec11Fab,1;
2434$       STACK,       1:Destroy:NEXT(2433$);

2433$       ASSIGN:      Sec 11 Fab.NumberOut=Sec 11 Fab.NumberOut + 1:
                         Sec 11 Fab.WIP=Sec 11 Fab.WIP-1:NEXT(65$);


;
;
;   Model statements for module:  BasicProcess.Process 24 (Sec 11 Asm Minor)
```

```
;
65$         ASSIGN:      Sec 11 Asm Minor.NumberIn=Sec 11 Asm Minor.NumberIn + 1:
                         Sec 11 Asm Minor.WIP=Sec 11 Asm Minor.WIP+1;
2465$       STACK,       1:Save:NEXT(2439$);

2439$       QUEUE,       Sec 11 Asm Minor.Queue;
2438$       SEIZE,       2,VA:
                         ResourceSec11AsmMinor,1:NEXT(2437$);

2437$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec11AsmMinor     *     LOG(attSN)
)+log(varSec11AsmMinorLN1))),     (10     *          *     ((varSlopeSec11AsmMinor     *     LOG(attSN)
)+log(varSec11AsmMinorLN1))),1.15*     (10     *          *     ((varSlopeSec11AsmMinor     *     LOG(attSN)
)+log(varSec11AsmMinorLN1)))),     TRIA(0.85*     varSec11AsmMinorLN100,varSec11AsmMinorLN100,1.15*
varSec11AsmMinorLN100)),,
                         VA:NEXT(2480$);

2480$       ASSIGN:      Sec 11 Asm Minor.WaitTime=Sec 11 Asm Minor.WaitTime + Diff.WaitTime;
2444$       TALLY:       Sec 11 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
2446$       TALLY:       Sec 11 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
2470$       ASSIGN:      Sec 11 Asm Minor.VATime=Sec 11 Asm Minor.VATime + Diff.VATime;
2471$       TALLY:       Sec 11 Asm Minor.VATimePerEntity,Diff.VATime,1;
2436$       RELEASE:     ResourceSec11AsmMinor,1;
2485$       STACK,       1:Destroy:NEXT(2484$);

2484$       ASSIGN:      Sec 11 Asm Minor.NumberOut=Sec 11 Asm Minor.NumberOut + 1:
                         Sec 11 Asm Minor.WIP=Sec 11 Asm Minor.WIP-1:NEXT(66$);


;
;
;   Model statements for module:  BasicProcess.Process 25 (Sec 11 Asm Major)
;
66$         ASSIGN:      Sec 11 Asm Major.NumberIn=Sec 11 Asm Major.NumberIn + 1:
                         Sec 11 Asm Major.WIP=Sec 11 Asm Major.WIP+1;
2516$       STACK,       1:Save:NEXT(2490$);

2490$       QUEUE,       Sec 11 Asm Major.Queue;
2489$       SEIZE,       2,VA:
                         ResourceSec11AsmMajor,1:NEXT(2488$);

2488$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec11AsmMajor     *     LOG(attSN)
)+log(varSec11AsmMajorLN1))),     (10     *          *     ((varSlopeSec11AsmMajor     *     LOG(attSN)
)+log(varSec11AsmMajorLN1))),1.15*     (10     *          *     ((varSlopeSec11AsmMajor     *     LOG(attSN)
)+log(varSec11AsmMajorLN1))))  ,  TRIA(0.85*  varSec11AsmMajorLN100,varSec11AsmMajorLN100,1.15*
varSec11AsmMajorLN100)),,
                         VA:NEXT(2531$);

2531$       ASSIGN:      Sec 11 Asm Major.WaitTime=Sec 11 Asm Major.WaitTime + Diff.WaitTime;
2495$       TALLY:       Sec 11 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
2497$       TALLY:       Sec 11 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
2521$       ASSIGN:      Sec 11 Asm Major.VATime=Sec 11 Asm Major.VATime + Diff.VATime;
2522$       TALLY:       Sec 11 Asm Major.VATimePerEntity,Diff.VATime,1;
2487$       RELEASE:     ResourceSec11AsmMajor,1;
2536$       STACK,       1:Destroy:NEXT(2535$);
```

```
2535$       ASSIGN:      Sec 11 Asm Major.NumberOut=Sec 11 Asm Major.NumberOut + 1:
                         Sec 11 Asm Major.WIP=Sec 11 Asm Major.WIP-1:NEXT(67$);


;
;
;   Model statements for module:  BasicProcess.Batch 5 (Batch Sec1145)
;
67$         QUEUE,       Batch Sec1145.Queue;
2538$       GROUP,       attSN,Permanent:2,Last:NEXT(2539$);

2539$       ASSIGN:      Batch Sec1145.NumberOut=Batch Sec1145.NumberOut + 1:NEXT(68$);


;
;
;   Model statements for module:  BasicProcess.Process 26 (Sec 1145 Join)
;
68$         ASSIGN:      Sec 1145 Join.NumberIn=Sec 1145 Join.NumberIn + 1:
                         Sec 1145 Join.WIP=Sec 1145 Join.WIP+1;
2569$       STACK,       1:Save:NEXT(2543$);

2543$       QUEUE,       Sec 1145 Join.Queue;
2542$       SEIZE,       2,VA:
                         ResourceSec1145Join,1:NEXT(2541$);

2541$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec1145Join * LOG(attSN) )+log(varSec1145JoinLN1))),
(10  *      *  ((varSlopeSec1145Join  *  LOG(attSN)  )+log(varSec1145JoinLN1))),1.15*  (10  *      *
((varSlopeSec1145Join                  *                  LOG(attSN)                  )+log(varSec1145JoinLN1)))),
TRIA(0.85*varSec1145JoinLN100,varSec1145JoinLN100,1.15*varSec1145JoinLN100)),,
                         VA:NEXT(2584$);

2584$       ASSIGN:      Sec 1145 Join.WaitTime=Sec 1145 Join.WaitTime + Diff.WaitTime;
2548$       TALLY:       Sec 1145 Join.WaitTimePerEntity,Diff.WaitTime,1;
2550$       TALLY:       Sec 1145 Join.TotalTimePerEntity,Diff.StartTime,1;
2574$       ASSIGN:      Sec 1145 Join.VATime=Sec 1145 Join.VATime + Diff.VATime;
2575$       TALLY:       Sec 1145 Join.VATimePerEntity,Diff.VATime,1;
2540$       RELEASE:     ResourceSec1145Join,1;
2589$       STACK,       1:Destroy:NEXT(2588$);

2588$       ASSIGN:      Sec 1145 Join.NumberOut=Sec 1145 Join.NumberOut + 1:
                         Sec 1145 Join.WIP=Sec 1145 Join.WIP-1:NEXT(106$);


;
;
;   Model statements for module:  BasicProcess.Process 105 (Ship Sec 11 45)
;
106$        ASSIGN:      Ship Sec 11 45.NumberIn=Ship Sec 11 45.NumberIn + 1:
                         Ship Sec 11 45.WIP=Ship Sec 11 45.WIP+1;
2620$       STACK,       1:Save:NEXT(2594$);

2594$       QUEUE,       Ship Sec 11 45.Queue;
2593$       SEIZE,       2,NVA:
                         ResourceSec1145Shipper,1:NEXT(2592$);
```

```
2592$       DELAY:      3,,NVA:NEXT(2635$);

2635$       ASSIGN:     Ship Sec 11 45.WaitTime=Ship Sec 11 45.WaitTime + Diff.WaitTime;
2599$       TALLY:      Ship Sec 11 45.WaitTimePerEntity,Diff.WaitTime,1;
2601$       TALLY:      Ship Sec 11 45.TotalTimePerEntity,Diff.StartTime,1;
2625$       ASSIGN:     Ship Sec 11 45.NVATime=Ship Sec 11 45.NVATime + Diff.NVATime;
2626$       TALLY:      Ship Sec 11 45.NVATimePerEntity,Diff.NVATime,1;
2591$       RELEASE:    ResourceSec1145Shipper,1;
2640$       STACK,      1:Destroy:NEXT(2639$);

2639$       ASSIGN:     Ship Sec 11 45.NumberOut=Ship Sec 11 45.NumberOut + 1:
                        Ship Sec 11 45.WIP=Ship Sec 11 45.WIP-1:NEXT(20$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 8 (Sec45 Order Arrival)
;

136$        STATION,    StnSec45Order;
2644$       DELAY:      0.0,,VA:NEXT(71$);


;
;
;   Model statements for module:  BasicProcess.Process 27 (Sec 45 Fab)
;
71$         ASSIGN:     Sec 45 Fab.NumberIn=Sec 45 Fab.NumberIn + 1:
                        Sec 45 Fab.WIP=Sec 45 Fab.WIP+1;
2674$       STACK,      1:Save:NEXT(2648$);

2648$       QUEUE,      Sec 45 Fab.Queue;
2647$       SEIZE,      2,VA:
                        ResourceSec45Fab,1:NEXT(2646$);

2646$       DELAY:
                        MX( TRIA(0.85* (10 *  * ((varSlopeSec45Fab * LOG(attSN) )+log(varSec45FabLN1))), (10 *
* ((varSlopeSec45Fab  *  LOG(attSN)  )+log(varSec45FabLN1))),1.15*  (10  *    *  ((varSlopeSec45Fab  *
LOG(attSN)                                  )+log(varSec45FabLN1)))),                                  TRIA(0.85*
varSec45FabLN100,varSec45FabLN100,1.15*varSec45FabLN100) ),,
                        VA:NEXT(2689$);

2689$       ASSIGN:     Sec 45 Fab.WaitTime=Sec 45 Fab.WaitTime + Diff.WaitTime;
2653$       TALLY:      Sec 45 Fab.WaitTimePerEntity,Diff.WaitTime,1;
2655$       TALLY:      Sec 45 Fab.TotalTimePerEntity,Diff.StartTime,1;
2679$       ASSIGN:     Sec 45 Fab.VATime=Sec 45 Fab.VATime + Diff.VATime;
2680$       TALLY:      Sec 45 Fab.VATimePerEntity,Diff.VATime,1;
2645$       RELEASE:    ResourceSec45Fab,1;
2694$       STACK,      1:Destroy:NEXT(2693$);

2693$       ASSIGN:     Sec 45 Fab.NumberOut=Sec 45 Fab.NumberOut + 1:
                        Sec 45 Fab.WIP=Sec 45 Fab.WIP-1:NEXT(72$);


;
;
;   Model statements for module:  BasicProcess.Process 28 (Sec 45 Asm Minor)
```

```
;
72$         ASSIGN:      Sec 45 Asm Minor.NumberIn=Sec 45 Asm Minor.NumberIn + 1:
                         Sec 45 Asm Minor.WIP=Sec 45 Asm Minor.WIP+1;
2725$       STACK,       1:Save:NEXT(2699$);

2699$       QUEUE,       Sec 45 Asm Minor.Queue;
2698$       SEIZE,       2,VA:
                         ResourceSec45AsmMinor,1:NEXT(2697$);

2697$       DELAY:
                         MX(TRIA(0.85*     (10     *          *     ((varSlopeSec45AsmMinor     *     LOG(attSN)
)+log(varSec45AsmMinorLN1))),     (10     *          *     ((varSlopeSec45AsmMinor     *     LOG(attSN)
)+log(varSec45AsmMinorLN1))),1.15*     (10     *          *     ((varSlopeSec45AsmMinor     *     LOG(attSN)
)+log(varSec45AsmMinorLN1))))                                                                    TRIA(0.85*
varSec45AsmMinorLN100,varSec45AsmMinorLN100,1.15*varSec45AsmMinorLN100)),,
                         VA:NEXT(2740$);

2740$       ASSIGN:      Sec 45 Asm Minor.WaitTime=Sec 45 Asm Minor.WaitTime + Diff.WaitTime;
2704$       TALLY:       Sec 45 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
2706$       TALLY:       Sec 45 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
2730$       ASSIGN:      Sec 45 Asm Minor.VATime=Sec 45 Asm Minor.VATime + Diff.VATime;
2731$       TALLY:       Sec 45 Asm Minor.VATimePerEntity,Diff.VATime,1;
2696$       RELEASE:     ResourceSec45AsmMinor,1;
2745$       STACK,       1:Destroy:NEXT(2744$);

2744$       ASSIGN:      Sec 45 Asm Minor.NumberOut=Sec 45 Asm Minor.NumberOut + 1:
                         Sec 45 Asm Minor.WIP=Sec 45 Asm Minor.WIP-1:NEXT(67$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 9 (Sec49 Order Arrival)
;

137$        STATION,     StnSec49Order;
2749$       DELAY:       0.0,,VA:NEXT(75$);


;
;
;   Model statements for module:  BasicProcess.Process 29 (Sec 49 Fab)
;
75$         ASSIGN:      Sec 49 Fab.NumberIn=Sec 49 Fab.NumberIn + 1:
                         Sec 49 Fab.WIP=Sec 49 Fab.WIP+1;
2779$       STACK,       1:Save:NEXT(2753$);

2753$       QUEUE,       Sec 49 Fab.Queue;
2752$       SEIZE,       2,VA:
                         ResourceSec49Fab,1:NEXT(2751$);

2751$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec49Fab * LOG(attSN) )+log(varSec49FabLN1))), (10 *
*  ((varSlopeSec49Fab  *  LOG(attSN)  )+log(varSec49FabLN1))),1.15*  (10  *     *  ((varSlopeSec49Fab  *
LOG(attSN)                                )+log(varSec49FabLN1)))),                                TRIA(0.85*
varSec49FabLN100,varSec49FabLN100,1.15*varSec49FabLN100)),,
                         VA:NEXT(2794$);
```

```
2794$       ASSIGN:      Sec 49 Fab.WaitTime=Sec 49 Fab.WaitTime + Diff.WaitTime;
2758$       TALLY:       Sec 49 Fab.WaitTimePerEntity,Diff.WaitTime,1;
2760$       TALLY:       Sec 49 Fab.TotalTimePerEntity,Diff.StartTime,1;
2784$       ASSIGN:      Sec 49 Fab.VATime=Sec 49 Fab.VATime + Diff.VATime;
2785$       TALLY:       Sec 49 Fab.VATimePerEntity,Diff.VATime,1;
2750$       RELEASE:     ResourceSec49Fab,1;
2799$       STACK,       1:Destroy:NEXT(2798$);

2798$       ASSIGN:      Sec 49 Fab.NumberOut=Sec 49 Fab.NumberOut + 1:
                         Sec 49 Fab.WIP=Sec 49 Fab.WIP-1:NEXT(76$);


;
;
;   Model statements for module:  BasicProcess.Process 30 (Sec 49 Asm Minor)
;
76$         ASSIGN:      Sec 49 Asm Minor.NumberIn=Sec 49 Asm Minor.NumberIn + 1:
                         Sec 49 Asm Minor.WIP=Sec 49 Asm Minor.WIP+1;
2830$       STACK,       1:Save:NEXT(2804$);

2804$       QUEUE,       Sec 49 Asm Minor.Queue;
2803$       SEIZE,       2,VA:
                         ResourceSec49AsmMinor,1:NEXT(2802$);

2802$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec49AsmMinor     *     LOG(attSN)
)+log(varSec49AsmMinorLN1))),     (10     *          *     ((varSlopeSec49AsmMinor     *     LOG(attSN)
)+log(varSec49AsmMinorLN1))),1.15*     (10     *          *     ((varSlopeSec49AsmMinor     *     LOG(attSN)
)+log(varSec49AsmMinorLN1)))),     TRIA(0.85*     varSec49AsmMinorLN100,varSec49AsmMinorLN100,1.15*
varSec49AsmMinorLN100)),,
                         VA:NEXT(2845$);

2845$       ASSIGN:      Sec 49 Asm Minor.WaitTime=Sec 49 Asm Minor.WaitTime + Diff.WaitTime;
2809$       TALLY:       Sec 49 Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
2811$       TALLY:       Sec 49 Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
2835$       ASSIGN:      Sec 49 Asm Minor.VATime=Sec 49 Asm Minor.VATime + Diff.VATime;
2836$       TALLY:       Sec 49 Asm Minor.VATimePerEntity,Diff.VATime,1;
2801$       RELEASE:     ResourceSec49AsmMinor,1;
2850$       STACK,       1:Destroy:NEXT(2849$);

2849$       ASSIGN:      Sec 49 Asm Minor.NumberOut=Sec 49 Asm Minor.NumberOut + 1:
                         Sec 49 Asm Minor.WIP=Sec 49 Asm Minor.WIP-1:NEXT(77$);


;
;
;   Model statements for module:  BasicProcess.Process 31 (Sec 49 Asm Major)
;
77$         ASSIGN:      Sec 49 Asm Major.NumberIn=Sec 49 Asm Major.NumberIn + 1:
                         Sec 49 Asm Major.WIP=Sec 49 Asm Major.WIP+1;
2881$       STACK,       1:Save:NEXT(2855$);

2855$       QUEUE,       Sec 49 Asm Major.Queue;
2854$       SEIZE,       2,VA:
                         ResourceSec49AsmMajor,1:NEXT(2853$);

2853$       DELAY:
```

```
                MX(    TRIA(0.85*    (10    *        *    ((varSlopeSec49AsmMajor    *    LOG(attSN)
)+log(varSec49AsmMajorLN1))),    (10    *        *    ((varSlopeSec49AsmMajor    *    LOG(attSN)
)+log(varSec49AsmMajorLN1))),1.15*    (10    *        *    ((varSlopeSec49AsmMajor    *    LOG(attSN)
)+log(varSec49AsmMajorLN1)))),    TRIA(0.85*    varSec49AsmMajorLN100,varSec49AsmMajorLN100,1.15*
varSec49AsmMajorLN100)),,
                VA:NEXT(2896$);

2896$      ASSIGN:     Sec 49 Asm Major.WaitTime=Sec 49 Asm Major.WaitTime + Diff.WaitTime;
2860$      TALLY:      Sec 49 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
2862$      TALLY:      Sec 49 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
2886$      ASSIGN:     Sec 49 Asm Major.VATime=Sec 49 Asm Major.VATime + Diff.VATime;
2887$      TALLY:      Sec 49 Asm Major.VATimePerEntity,Diff.VATime,1;
2852$      RELEASE:    ResourceSec49AsmMajor,1;
2901$      STACK,      1:Destroy:NEXT(2900$);

2900$      ASSIGN:     Sec 49 Asm Major.NumberOut=Sec 49 Asm Major.NumberOut + 1:
                       Sec 49 Asm Major.WIP=Sec 49 Asm Major.WIP-1:NEXT(107$);


;
;
;   Model statements for module:  BasicProcess.Process 106 (Ship Sec 49)
;
107$       ASSIGN:     Ship Sec 49.NumberIn=Ship Sec 49.NumberIn + 1:
                       Ship Sec 49.WIP=Ship Sec 49.WIP+1;
2932$      STACK,      1:Save:NEXT(2906$);

2906$      QUEUE,      Ship Sec 49.Queue;
2905$      SEIZE,      2,NVA:
                       ResourceSec49Shipper,1:NEXT(2904$);

2904$      DELAY:      4,,NVA:NEXT(2947$);

2947$      ASSIGN:     Ship Sec 49.WaitTime=Ship Sec 49.WaitTime + Diff.WaitTime;
2911$      TALLY:      Ship Sec 49.WaitTimePerEntity,Diff.WaitTime,1;
2913$      TALLY:      Ship Sec 49.TotalTimePerEntity,Diff.StartTime,1;
2937$      ASSIGN:     Ship Sec 49.NVATime=Ship Sec 49.NVATime + Diff.NVATime;
2938$      TALLY:      Ship Sec 49.NVATimePerEntity,Diff.NVATime,1;
2903$      RELEASE:    ResourceSec49Shipper,1;
2952$      STACK,      1:Destroy:NEXT(2951$);

2951$      ASSIGN:     Ship Sec 49.NumberOut=Ship Sec 49.NumberOut + 1:
                       Ship Sec 49.WIP=Ship Sec 49.WIP-1:NEXT(78$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 10 (Sec47 Order Arrival)
;
146$       STATION,    StnSec47Order;
2956$      DELAY:      0.0,,VA:NEXT(82$);


;
;
;   Model statements for module:  BasicProcess.Process 34 (Sec 47 Fab)
```

```
;
82$         ASSIGN:      Sec 47 Fab.NumberIn=Sec 47 Fab.NumberIn + 1:
                         Sec 47 Fab.WIP=Sec 47 Fab.WIP+1;
2986$       STACK,       1:Save:NEXT(2960$);

2960$       QUEUE,       Sec 47 Fab.Queue;
2959$       SEIZE,       2,VA:
                         ResourceSec47Fab,1:NEXT(2958$);

2958$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec47Fab * LOG(attSN) )+log(varSec47FabLN1))), (10 *
*  ((varSlopeSec47Fab  *  LOG(attSN)  )+log(varSec47FabLN1))),1.15*  (10  *     *  ((varSlopeSec47Fab  *
LOG(attSN)                                  )+log(varSec47FabLN1)))),                                  TRIA(0.85*
varSec47FabLN100,varSec47FabLN100,1.15*varSec47FabLN100)),,
                         VA:NEXT(3001$);

3001$       ASSIGN:      Sec 47 Fab.WaitTime=Sec 47 Fab.WaitTime + Diff.WaitTime;
2965$       TALLY:       Sec 47 Fab.WaitTimePerEntity,Diff.WaitTime,1;
2967$       TALLY:       Sec 47 Fab.TotalTimePerEntity,Diff.StartTime,1;
2991$       ASSIGN:      Sec 47 Fab.VATime=Sec 47 Fab.VATime + Diff.VATime;
2992$       TALLY:       Sec 47 Fab.VATimePerEntity,Diff.VATime,1;
2957$       RELEASE:     ResourceSec47Fab,1;
3006$       STACK,       1:Destroy:NEXT(3005$);

3005$       ASSIGN:      Sec 47 Fab.NumberOut=Sec 47 Fab.NumberOut + 1:
                         Sec 47 Fab.WIP=Sec 47 Fab.WIP-1:NEXT(83$);


;
;
;   Model statements for module:  BasicProcess.Process 35 (Sec 47 Asm Major)
;
83$         ASSIGN:      Sec 47 Asm Major.NumberIn=Sec 47 Asm Major.NumberIn + 1:
                         Sec 47 Asm Major.WIP=Sec 47 Asm Major.WIP+1;
3037$       STACK,       1:Save:NEXT(3011$);

3011$       QUEUE,       Sec 47 Asm Major.Queue;
3010$       SEIZE,       2,VA:
                         ResourceSec47AsmMajor,1:NEXT(3009$);

3009$       DELAY:
                         MX(     TRIA(0.85*     (10     *          *     ((varSlopeSec47AsmMajor     *     LOG(attSN)
)+log(varSec47AsmMajorLN1))),      (10      *          *     ((varSlopeSec47AsmMajor      *     LOG(attSN)
)+log(varSec47AsmMajorLN1))),1.15*     (10     *          *     ((varSlopeSec47AsmMajor     *     LOG(attSN)
)+log(varSec47AsmMajorLN1)))),                                                                  TRIA(0.85*
varSec47AsmMajorLN100,varSec47AsmMajorLN100,1.15*varSec47AsmMajorLN100)),,
                         VA:NEXT(3052$);

3052$       ASSIGN:      Sec 47 Asm Major.WaitTime=Sec 47 Asm Major.WaitTime + Diff.WaitTime;
3016$       TALLY:       Sec 47 Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
3018$       TALLY:       Sec 47 Asm Major.TotalTimePerEntity,Diff.StartTime,1;
3042$       ASSIGN:      Sec 47 Asm Major.VATime=Sec 47 Asm Major.VATime + Diff.VATime;
3043$       TALLY:       Sec 47 Asm Major.VATimePerEntity,Diff.VATime,1;
3008$       RELEASE:     ResourceSec47AsmMajor,1;
3057$       STACK,       1:Destroy:NEXT(3056$);

3056$       ASSIGN:      Sec 47 Asm Major.NumberOut=Sec 47 Asm Major.NumberOut + 1:
```

```
                Sec 47 Asm Major.WIP=Sec 47 Asm Major.WIP-1:NEXT(85$);


;
;
;   Model statements for module:  BasicProcess.Batch 7 (Batch Sec4748)
;
85$         QUEUE,      Batch Sec4748.Queue;
3059$       GROUP,      attSN,Permanent:2,Last:NEXT(3060$);

3060$       ASSIGN:     Batch Sec4748.NumberOut=Batch Sec4748.NumberOut + 1:NEXT(84$);


;
;
;   Model statements for module:  BasicProcess.Process 36 (Sec 4748 Join)
;
84$         ASSIGN:     Sec 4748 Join.NumberIn=Sec 4748 Join.NumberIn + 1:
                        Sec 4748 Join.WIP=Sec 4748 Join.WIP+1;
3090$       STACK,      1:Save:NEXT(3064$);

3064$       QUEUE,      Sec 4748 Join.Queue;
3063$       SEIZE,      2,VA:
                        ResourceSec4748Join,1:NEXT(3062$);

3062$       DELAY:
                        MX( TRIA(0.85* (10 *  * ((varSlopeSec4748Join * LOG(attSN) )+log(varSec4748JoinLN1))),
(10  *      *  ((varSlopeSec4748Join  *  LOG(attSN)  )+log(varSec4748JoinLN1))),1.15*  (10  *  *
((varSlopeSec4748Join  *  LOG(attSN)  )+log(varSec4748JoinLN1)))),  TRIA(0.85*
varSec4748JoinLN100,varSec4748JoinLN100,1.15*varSec4748JoinLN100)),,
                        VA:NEXT(3105$);

3105$       ASSIGN:     Sec 4748 Join.WaitTime=Sec 4748 Join.WaitTime + Diff.WaitTime;
3069$       TALLY:      Sec 4748 Join.WaitTimePerEntity,Diff.WaitTime,1;
3071$       TALLY:      Sec 4748 Join.TotalTimePerEntity,Diff.StartTime,1;
3095$       ASSIGN:     Sec 4748 Join.VATime=Sec 4748 Join.VATime + Diff.VATime;
3096$       TALLY:      Sec 4748 Join.VATimePerEntity,Diff.VATime,1;
3061$       RELEASE:    ResourceSec4748Join,1;
3110$       STACK,      1:Destroy:NEXT(3109$);

3109$       ASSIGN:     Sec 4748 Join.NumberOut=Sec 4748 Join.NumberOut + 1:
                        Sec 4748 Join.WIP=Sec 4748 Join.WIP-1:NEXT(109$);


;
;
;   Model statements for module:  BasicProcess.Process 108 (Ship Sec 47 48 joined)
;
109$        ASSIGN:     Ship Sec 47 48 joined.NumberIn=Ship Sec 47 48 joined.NumberIn + 1:
                        Ship Sec 47 48 joined.WIP=Ship Sec 47 48 joined.WIP+1;
3141$       STACK,      1:Save:NEXT(3115$);

3115$       QUEUE,      Ship Sec 47 48 joined.Queue;
3114$       SEIZE,      2,NVA:
                        ResourceSec4748Shipper,1:NEXT(3113$);

3113$       DELAY:      3,,NVA:NEXT(3156$);
```

```
3156$       ASSIGN:      Ship Sec 47 48 joined.WaitTime=Ship Sec 47 48 joined.WaitTime + Diff.WaitTime;
3120$       TALLY:       Ship Sec 47 48 joined.WaitTimePerEntity,Diff.WaitTime,1;
3122$       TALLY:       Ship Sec 47 48 joined.TotalTimePerEntity,Diff.StartTime,1;
3146$       ASSIGN:      Ship Sec 47 48 joined.NVATime=Ship Sec 47 48 joined.NVATime + Diff.NVATime;
3147$       TALLY:       Ship Sec 47 48 joined.NVATimePerEntity,Diff.NVATime,1;
3112$       RELEASE:     ResourceSec4748Shipper,1;
3161$       STACK,       1:Destroy:NEXT(3160$);

3160$       ASSIGN:      Ship Sec 47 48 joined.NumberOut=Ship Sec 47 48 joined.NumberOut + 1:
                         Ship Sec 47 48 joined.WIP=Ship Sec 47 48 joined.WIP-1:NEXT(16$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 11 (Sec48 Fwd Order Arrival)
;

147$        STATION,     StnSec48FwdOrder;
3165$       DELAY:       0.0,,VA:NEXT(88$);


;
;
;   Model statements for module:  BasicProcess.Process 37 (Sec 48 Fwd Fab)
;
88$         ASSIGN:      Sec 48 Fwd Fab.NumberIn=Sec 48 Fwd Fab.NumberIn + 1:
                         Sec 48 Fwd Fab.WIP=Sec 48 Fwd Fab.WIP+1;
3195$       STACK,       1:Save:NEXT(3169$);

3169$       QUEUE,       Sec 48 Fwd Fab.Queue;
3168$       SEIZE,       2,VA:
                         ResourceSec48FwdFab,1:NEXT(3167$);

3167$       DELAY:
                         MX(          TRIA(0.85*     (10     *          *     ((varSlopeSec48FwdFab     *     LOG(attSN)
)+log(varSec48FwdFabLN1))),       (10       *            *       ((varSlopeSec48FwdFab       *       LOG(attSN)
)+log(varSec48FwdFabLN1))),1.15*      (10      *           *      ((varSlopeSec48FwdFab      *      LOG(attSN)
)+log(varSec48FwdFabLN1)))),                                                                          TRIA(0.85*
varSec48FwdFabLN100,varSec48FwdFabLN100,1.15*varSec48FwdFabLN100)),,
                         VA:NEXT(3210$);

3210$       ASSIGN:      Sec 48 Fwd Fab.WaitTime=Sec 48 Fwd Fab.WaitTime + Diff.WaitTime;
3174$       TALLY:       Sec 48 Fwd Fab.WaitTimePerEntity,Diff.WaitTime,1;
3176$       TALLY:       Sec 48 Fwd Fab.TotalTimePerEntity,Diff.StartTime,1;
3200$       ASSIGN:      Sec 48 Fwd Fab.VATime=Sec 48 Fwd Fab.VATime + Diff.VATime;
3201$       TALLY:       Sec 48 Fwd Fab.VATimePerEntity,Diff.VATime,1;
3166$       RELEASE:     ResourceSec48FwdFab,1;
3215$       STACK,       1:Destroy:NEXT(3214$);

3214$       ASSIGN:      Sec 48 Fwd Fab.NumberOut=Sec 48 Fwd Fab.NumberOut + 1:
                         Sec 48 Fwd Fab.WIP=Sec 48 Fwd Fab.WIP-1:NEXT(89$);


;
;
;   Model statements for module:  BasicProcess.Process 38 (Sec 48 Fwd Asm Major)
```

```
;
89$         ASSIGN:      Sec 48 Fwd Asm Major.NumberIn=Sec 48 Fwd Asm Major.NumberIn + 1:
                         Sec 48 Fwd Asm Major.WIP=Sec 48 Fwd Asm Major.WIP+1;
3246$       STACK,       1:Save:NEXT(3220$);

3220$       QUEUE,       Sec 48 Fwd Asm Major.Queue;
3219$       SEIZE,       2,VA:
                         ResourceSec48FwdAsmMajor,1:NEXT(3218$);

3218$       DELAY:
                         MX(    TRIA(0.85*    (10    *        *    ((varSlopeSec48FwdAsmMajor    *    LOG(attSN)
)+log(varSec48FwdAsmMajorLN1))),    (10    *        *    ((varSlopeSec48FwdAsmMajor    *    LOG(attSN)
)+log(varSec48FwdAsmMajorLN1))),1.15*    (10    *        *    ((varSlopeSec48FwdAsmMajor    *    LOG(attSN)
)+log(varSec48FwdAsmMajorLN1)))),                                                                    TRIA(0.85*
varSec48FwdAsmMajorLN100,varSec48FwdAsmMajorLN100,1.15*varSec48FwdAsmMajorLN100)),,
                         VA:NEXT(3261$);

3261$            ASSIGN:          Sec 48 Fwd Asm Major.WaitTime=Sec 48 Fwd Asm Major.WaitTime +
Diff.WaitTime;
3225$       TALLY:       Sec 48 Fwd Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
3227$       TALLY:       Sec 48 Fwd Asm Major.TotalTimePerEntity,Diff.StartTime,1;
3251$       ASSIGN:      Sec 48 Fwd Asm Major.VATime=Sec 48 Fwd Asm Major.VATime + Diff.VATime;
3252$       TALLY:       Sec 48 Fwd Asm Major.VATimePerEntity,Diff.VATime,1;
3217$       RELEASE:     ResourceSec48FwdAsmMajor,1;
3266$       STACK,       1:Destroy:NEXT(3265$);

3265$       ASSIGN:      Sec 48 Fwd Asm Major.NumberOut=Sec 48 Fwd Asm Major.NumberOut + 1:
                         Sec 48 Fwd Asm Major.WIP=Sec 48 Fwd Asm Major.WIP-1:NEXT(85$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 12 (Hortz Stab Order Arrival)
;

152$        STATION,     StnHortzStabOrder;
3270$       DELAY:       0.0,,VA:NEXT(92$);


;
;
;   Model statements for module:  BasicProcess.Process 39 (Horizontal Stabilizer Fab)
;
92$         ASSIGN:      Horizontal Stabilizer Fab.NumberIn=Horizontal Stabilizer Fab.NumberIn + 1:
                         Horizontal Stabilizer Fab.WIP=Horizontal Stabilizer Fab.WIP+1;
3300$       STACK,       1:Save:NEXT(3274$);

3274$       QUEUE,       Horizontal Stabilizer Fab.Queue;
3273$       SEIZE,       2,VA:
                         ResourceHStbzFab,1:NEXT(3272$);

3272$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeHStbzFab * LOG(attSN) )+log(varHStbzFabLN1))), (10 *  *
((varSlopeHStbzFab * LOG(attSN) )+log(varHStbzFabLN1))),1.15* (10 *  * ((varSlopeHStbzFab * LOG(attSN)
)+log(varHStbzFabLN1)))), TRIA(0.85* varHStbzFabLN100,varHStbzFabLN100,1.15*varHStbzFabLN100)),,
                         VA:NEXT(3315$);
```

```
3315$          ASSIGN:          Horizontal Stabilizer Fab.WaitTime=Horizontal Stabilizer Fab.WaitTime +
Diff.WaitTime;
3279$      TALLY:      Horizontal Stabilizer Fab.WaitTimePerEntity,Diff.WaitTime,1;
3281$      TALLY:      Horizontal Stabilizer Fab.TotalTimePerEntity,Diff.StartTime,1;
3305$      ASSIGN:      Horizontal Stabilizer Fab.VATime=Horizontal Stabilizer Fab.VATime + Diff.VATime;
3306$      TALLY:      Horizontal Stabilizer Fab.VATimePerEntity,Diff.VATime,1;
3271$      RELEASE:      ResourceHStbzFab,1;
3320$      STACK,      1:Destroy:NEXT(3319$);

3319$      ASSIGN:      Horizontal Stabilizer Fab.NumberOut=Horizontal Stabilizer Fab.NumberOut + 1:
                Horizontal Stabilizer Fab.WIP=Horizontal Stabilizer Fab.WIP-1:NEXT(93$);


;
;
;   Model statements for module:  BasicProcess.Process 40 (Horizontal Stabilizer Asm Minor)
;
93$        ASSIGN:      Horizontal Stabilizer Asm Minor.NumberIn=Horizontal Stabilizer Asm Minor.NumberIn
+ 1:
                Horizontal Stabilizer Asm Minor.WIP=Horizontal Stabilizer Asm Minor.WIP+1;
3351$      STACK,      1:Save:NEXT(3325$);

3325$      QUEUE,      Horizontal Stabilizer Asm Minor.Queue;
3324$      SEIZE,      2,VA:
                ResourceHStbzAsmMinor,1:NEXT(3323$);

3323$      DELAY:
                MX(     TRIA(0.85*     (10     *          *     ((varSlopeHStbzAsmMinor     *     LOG(attSN)
)+log(varHStbzAsmMinorLN1))),     (10     *          *     ((varSlopeHStbzAsmMinor     *     LOG(attSN)
)+log(varHStbzAsmMinorLN1))),1.15*     (10     *          *     ((varSlopeHStbzAsmMinor     *     LOG(attSN)
)+log(varHStbzAsmMinorLN1)))),                                                                    TRIA(0.85*
varHStbzAsmMinorLN100,varHStbzAsmMinorLN100,1.15*varHStbzAsmMinorLN100)),,
                VA:NEXT(3366$);

3366$      ASSIGN:      Horizontal Stabilizer Asm Minor.WaitTime=Horizontal Stabilizer Asm Minor.WaitTime
+ Diff.WaitTime;
3330$      TALLY:      Horizontal Stabilizer Asm Minor.WaitTimePerEntity,Diff.WaitTime,1;
3332$      TALLY:      Horizontal Stabilizer Asm Minor.TotalTimePerEntity,Diff.StartTime,1;
3356$      ASSIGN:      Horizontal Stabilizer Asm Minor.VATime=Horizontal Stabilizer Asm Minor.VATime +
Diff.VATime;
3357$      TALLY:      Horizontal Stabilizer Asm Minor.VATimePerEntity,Diff.VATime,1;
3322$      RELEASE:      ResourceHStbzAsmMinor,1;
3371$      STACK,      1:Destroy:NEXT(3370$);

3370$          ASSIGN:          Horizontal Stabilizer Asm Minor.NumberOut=Horizontal Stabilizer Asm
Minor.NumberOut + 1:
                Horizontal Stabilizer Asm Minor.WIP=Horizontal Stabilizer Asm Minor.WIP-1:NEXT(94$);


;
;
;   Model statements for module:  BasicProcess.Process 41 (Horizontal Stabilizer Asm Major)
;
94$        ASSIGN:      Horizontal Stabilizer Asm Major.NumberIn=Horizontal Stabilizer Asm Major.NumberIn
+ 1:
                Horizontal Stabilizer Asm Major.WIP=Horizontal Stabilizer Asm Major.WIP+1;
3402$      STACK,      1:Save:NEXT(3376$);
```

```
3376$       QUEUE,      Horizontal Stabilizer Asm Major.Queue;
3375$       SEIZE,      2,VA:
                ResourceHStbzAsmMajor,1:NEXT(3374$);

3374$       DELAY:
                MX(     TRIA(0.85*     (10     *       *     ((varSlopeHStbzAsmMajor     *     LOG(attSN)
)+log(varHStbzAsmMajorLN1))),     (10     *       *     ((varSlopeHStbzAsmMajor     *     LOG(attSN)
)+log(varHStbzAsmMajorLN1))),1.15*     (10     *       *     ((varSlopeHStbzAsmMajor     *     LOG(attSN)
)+log(varHStbzAsmMajorLN1)))),                                                          TRIA(0.85*
varHStbzAsmMajorLN100,varHStbzAsmMajorLN100,1.15*varHStbzAsmMajorLN100)),,
                VA:NEXT(3417$);

3417$       ASSIGN:     Horizontal Stabilizer Asm Major.WaitTime=Horizontal Stabilizer Asm Major.WaitTime
+ Diff.WaitTime;
3381$       TALLY:      Horizontal Stabilizer Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
3383$       TALLY:      Horizontal Stabilizer Asm Major.TotalTimePerEntity,Diff.StartTime,1;
3407$       ASSIGN:     Horizontal Stabilizer Asm Major.VATime=Horizontal Stabilizer Asm Major.VATime +
Diff.VATime;
3408$       TALLY:      Horizontal Stabilizer Asm Major.VATimePerEntity,Diff.VATime,1;
3373$       RELEASE:    ResourceHStbzAsmMajor,1;
3422$       STACK,      1:Destroy:NEXT(3421$);

3421$       ASSIGN:     Horizontal Stabilizer Asm Major.NumberOut=Horizontal Stabilizer Asm
Major.NumberOut + 1:
                Horizontal Stabilizer Asm Major.WIP=Horizontal Stabilizer Asm Major.WIP-1:NEXT(97$);


;
;
;   Model statements for module:  BasicProcess.Process 43 (Ship HStabilizer)
;
97$         ASSIGN:     Ship HStabilizer.NumberIn=Ship HStabilizer.NumberIn + 1:
                Ship HStabilizer.WIP=Ship HStabilizer.WIP+1;
3453$       STACK,      1:Save:NEXT(3427$);

3427$       QUEUE,      Ship HStabilizer.Queue;
3426$       SEIZE,      2,NVA:
                ResourceHStbzShipper,1:NEXT(3425$);

3425$       DELAY:      28,,NVA:NEXT(3468$);

3468$       ASSIGN:     Ship HStabilizer.WaitTime=Ship HStabilizer.WaitTime + Diff.WaitTime;
3432$       TALLY:      Ship HStabilizer.WaitTimePerEntity,Diff.WaitTime,1;
3434$       TALLY:      Ship HStabilizer.TotalTimePerEntity,Diff.StartTime,1;
3458$       ASSIGN:     Ship HStabilizer.NVATime=Ship HStabilizer.NVATime + Diff.NVATime;
3459$       TALLY:      Ship HStabilizer.NVATimePerEntity,Diff.NVATime,1;
3424$       RELEASE:    ResourceHStbzShipper,1;
3473$       STACK,      1:Destroy:NEXT(3472$);

3472$       ASSIGN:     Ship HStabilizer.NumberOut=Ship HStabilizer.NumberOut + 1:
                Ship HStabilizer.WIP=Ship HStabilizer.WIP-1:NEXT(95$);


;
;
;   Model statements for module:  BasicProcess.Batch 8 (Batch Horz Stabilizer and Sec 48 Aft)
```

```
;
95$         QUEUE,       Batch Horz Stabilizer and Sec 48 Aft.Queue;
3475$       GROUP,       attSN,Permanent:2,Last:NEXT(3476$);

3476$       ASSIGN:      Batch Horz Stabilizer and Sec 48 Aft.NumberOut=Batch Horz Stabilizer and Sec 48
Aft.NumberOut + 1
                         :NEXT(96$);


;
;
;   Model statements for module:  BasicProcess.Process 42 (Horz Stabilizer Sec 48 Aft Join)
;
96$         ASSIGN:      Horz Stabilizer Sec 48 Aft Join.NumberIn=Horz Stabilizer Sec 48 Aft Join.NumberIn +
1:
                         Horz Stabilizer Sec 48 Aft Join.WIP=Horz Stabilizer Sec 48 Aft Join.WIP+1;
3506$       STACK,       1:Save:NEXT(3480$);

3480$       QUEUE,       Horz Stabilizer Sec 48 Aft Join.Queue;
3479$       SEIZE,       2,VA:
                         ResourceHStbzJoin,1:NEXT(3478$);

3478$       DELAY:
                         MX( TRIA(0.85* (10 * * ((varSlopeHStbzJoin * LOG(attSN) )+log(varHStbzJoinLN1))), (10 *
* ((varSlopeHStbzJoin * LOG(attSN) )+log(varHStbzJoinLN1))),1.15* (10 * * ((varSlopeHStbzJoin *
LOG(attSN) )+log(varHStbzJoinLN1)))),  TRIA(0.85* varHStbzJoinLN100,varHStbzJoinLN100,1.15*
varHStbzJoinLN100)),,
                         VA:NEXT(3521$);

3521$       ASSIGN:      Horz Stabilizer Sec 48 Aft Join.WaitTime=Horz Stabilizer Sec 48 Aft Join.WaitTime +
Diff.WaitTime;
3485$       TALLY:       Horz Stabilizer Sec 48 Aft Join.WaitTimePerEntity,Diff.WaitTime,1;
3487$       TALLY:       Horz Stabilizer Sec 48 Aft Join.TotalTimePerEntity,Diff.StartTime,1;
3511$       ASSIGN:      Horz Stabilizer Sec 48 Aft Join.VATime=Horz Stabilizer Sec 48 Aft Join.VATime +
Diff.VATime;
3512$       TALLY:       Horz Stabilizer Sec 48 Aft Join.VATimePerEntity,Diff.VATime,1;
3477$       RELEASE:     ResourceHStbzJoin,1;
3526$       STACK,       1:Destroy:NEXT(3525$);

3525$       ASSIGN:      Horz Stabilizer Sec 48 Aft Join.NumberOut=Horz Stabilizer Sec 48 Aft
Join.NumberOut + 1:
                         Horz Stabilizer Sec 48 Aft Join.WIP=Horz Stabilizer Sec 48 Aft Join.WIP-1:NEXT(16$);


;
;
;   Model statements for module:  AdvancedTransfer.Station 13 (Sec48 Aft Order Arrival)
;

157$        STATION,     StnSec48AftOrder;
3530$       DELAY:       0.0,,VA:NEXT(100$);


;
;
;   Model statements for module:  BasicProcess.Process 44 (Sec 48 Aft Fab)
;
```

```
100$        ASSIGN:      Sec 48 Aft Fab.NumberIn=Sec 48 Aft Fab.NumberIn + 1:
                         Sec 48 Aft Fab.WIP=Sec 48 Aft Fab.WIP+1;
3560$       STACK,       1:Save:NEXT(3534$);

3534$       QUEUE,       Sec 48 Aft Fab.Queue;
3533$       SEIZE,       2,VA:
                         ResourceSec48AftFab,1:NEXT(3532$);

3532$       DELAY:
                         MX( TRIA(0.85* (10 *  * ((varSlopeSec48AftFab * LOG(attSN) )+log(varSec48AftFabLN1))),
(10  *      *  ((varSlopeSec48AftFab  *  LOG(attSN)  )+log(varSec48AftFabLN1))),1.15*  (10  *  *
((varSlopeSec48AftFab  *  LOG(attSN)  )+log(varSec48AftFabLN1)))),  TRIA(0.85*
varSec48AftFabLN100,varSec48AftFabLN100,1.15* varSec48AftFabLN100)),,
                         VA:NEXT(3575$);

3575$       ASSIGN:      Sec 48 Aft Fab.WaitTime=Sec 48 Aft Fab.WaitTime + Diff.WaitTime;
3539$       TALLY:       Sec 48 Aft Fab.WaitTimePerEntity,Diff.WaitTime,1;
3541$       TALLY:       Sec 48 Aft Fab.TotalTimePerEntity,Diff.StartTime,1;
3565$       ASSIGN:      Sec 48 Aft Fab.VATime=Sec 48 Aft Fab.VATime + Diff.VATime;
3566$       TALLY:       Sec 48 Aft Fab.VATimePerEntity,Diff.VATime,1;
3531$       RELEASE:     ResourceSec48AftFab,1;
3580$       STACK,       1:Destroy:NEXT(3579$);

3579$       ASSIGN:      Sec 48 Aft Fab.NumberOut=Sec 48 Aft Fab.NumberOut + 1:
                         Sec 48 Aft Fab.WIP=Sec 48 Aft Fab.WIP-1:NEXT(101$);


;
;
;   Model statements for module:  BasicProcess.Process 45 (Sec 48 Aft Asm Major)
;
101$        ASSIGN:      Sec 48 Aft Asm Major.NumberIn=Sec 48 Aft Asm Major.NumberIn + 1:
                         Sec 48 Aft Asm Major.WIP=Sec 48 Aft Asm Major.WIP+1;
3611$       STACK,       1:Save:NEXT(3585$);
3585$       QUEUE,       Sec 48 Aft Asm Major.Queue;
3584$       SEIZE,       2,VA:
                         ResourceSec48AftAsmMajor,1:NEXT(3583$);

3583$       DELAY:
                         MX(  TRIA(0.85*  (10  *  *  ((varSlopeSec48AftAsmMajor  *  LOG(attSN)
)+log(varSec48AftAsmMajorLN1))),  (10  *  *  ((varSlopeSec48AftAsmMajor  *  LOG(attSN)
)+log(varSec48AftAsmMajorLN1))),1.15*  (10  *  *  ((varSlopeSec48AftAsmMajor  *  LOG(attSN)
)+log(varSec48AftAsmMajorLN1)))),                                                    TRIA(0.85*
varSec48AftAsmMajorLN100,varSec48AftAsmMajorLN100,1.15*varSec48AftAsmMajorLN100)),,
                         VA:NEXT(3626$);

3626$       ASSIGN:      Sec 48 Aft Asm Major.WaitTime=Sec 48 Aft Asm Major.WaitTime + Diff.WaitTime;
3590$       TALLY:       Sec 48 Aft Asm Major.WaitTimePerEntity,Diff.WaitTime,1;
3592$       TALLY:       Sec 48 Aft Asm Major.TotalTimePerEntity,Diff.StartTime,1;
3616$       ASSIGN:      Sec 48 Aft Asm Major.VATime=Sec 48 Aft Asm Major.VATime + Diff.VATime;
3617$       TALLY:       Sec 48 Aft Asm Major.VATimePerEntity,Diff.VATime,1;
3582$       RELEASE:     ResourceSec48AftAsmMajor,1;
3631$       STACK,       1:Destroy:NEXT(3630$);
3630$       ASSIGN:      Sec 48 Aft Asm Major.NumberOut=Sec 48 Aft Asm Major.NumberOut + 1:
                         Sec 48 Aft Asm Major.WIP=Sec 48 Aft Asm Major.WIP-1:NEXT(102$);
```

```
;
;
;   Model statements for module:  BasicProcess.Process 46 (Ship Sec48 Aft)
;
102$        ASSIGN:      Ship Sec48 Aft.NumberIn=Ship Sec48 Aft.NumberIn + 1:
                         Ship Sec48 Aft.WIP=Ship Sec48 Aft.WIP+1;
3662$       STACK,       1:Save:NEXT(3636$);

3636$       QUEUE,       Ship Sec48 Aft.Queue;
3635$       SEIZE,       2,NVA:
                         ResourceSec48AftShipper,1:NEXT(3634$);

3634$       DELAY:       21,,NVA:NEXT(3677$);

3677$       ASSIGN:      Ship Sec48 Aft.WaitTime=Ship Sec48 Aft.WaitTime + Diff.WaitTime;
3641$       TALLY:       Ship Sec48 Aft.WaitTimePerEntity,Diff.WaitTime,1;
3643$       TALLY:       Ship Sec48 Aft.TotalTimePerEntity,Diff.StartTime,1;
3667$       ASSIGN:      Ship Sec48 Aft.NVATime=Ship Sec48 Aft.NVATime + Diff.NVATime;
3668$       TALLY:       Ship Sec48 Aft.NVATimePerEntity,Diff.NVATime,1;
3633$       RELEASE:     ResourceSec48AftShipper,1;
3682$       STACK,       1:Destroy:NEXT(3681$);

3681$       ASSIGN:      Ship Sec48 Aft.NumberOut=Ship Sec48 Aft.NumberOut + 1:
                         Ship Sec48 Aft.WIP=Ship Sec48 Aft.WIP-1:NEXT(95$);


;
;
;   Model statements for module:  AdvancedProcess.StateSet 1 (StateSetDisruption)
;
```

VITA

ROBERTO F. LU

EDUCATION

Doctor of Philosophy in Industrial Engineering, 2008.
University of Washington, Seattle, Washington, USA.

Master of Science in Industrial Engineering, 2004.
University of Washington, Seattle, Washington, USA.

Master of Engineering in Industrial and Systems Engineering, 1994.
Virginia Tech, Blacksburg, Virginia, USA.

Master of Science in Mechanical Engineering, 1989.
Marquette University, Milwaukee, Wisconsin, USA.

Bachelor of Science in Materials Science, 1985.
Feng Chia University, Taichung, Taiwan.

PROFESSIONAL REGISTRATION

Registered and licensed Professional Engineer (PE), State of Ohio, since 4/1996.

PROFESSIONAL EXPERIENCE

The Boeing Company, Seattle, Washington, 10/1997 – present

Associate Technical Fellow,
Boeing Commercial Airplanes Material & Process Technology, 10/2002 – present

- Applied discrete event simulation and optimization models to broad range of operations and major component fabrications in Boeing's commercial airplanes 737, 747, 767, 777, and 787 programs.
- Led, developed, planned, prioritized, organized, executed, and monitored simulation modeling related strategies and techniques systematically with team approach.
- Managed technical projects with close interactions with customers, team members, and finance personnel.
- Shortened project iterations and cycle times from concept through deployment stages by simulation and optimization.
- Invented new methodologies to model the 787 global production system.
- Invented new algorithms in simulation models that can analyze and dynamically create 787 DreamLifter demand driven optimal schedules.

- Tested and successfully modeled the use of an XML specification (from the NIST) on a wing fabrication line.
- Established and taught discrete event simulation and risk analysis optimization classes within Boeing to fellow engineers and financial analysts.
- Modeled multi-server multi-queue job shop operations for batched traceable group of components at the Boeing Portland fabrication plant and presented solutions to reduce the overall batching process times by 300+%.
- Innovated transporter constrained logistics modeling for the 787 global logistics that justified the use of Large Cargo Freighters (DreamLifter), instead of ocean vessels, to transport major 787 large components globally.
- Invited to present and to publish book chapters, journal papers, and conference proceedings internationally.
- Mentored and trained team members and fellow Boeing employees.
- Facilitated technology classes that benefited more than 1200 attendees from Boeing organizations across the North America.

Sr. Principal Simulation Engineer,
BCA Manufacturing Research & Development (MR&D), 6/1999 – 10/2002

- Constructed 3D discrete event simulation models to solve and analyze production logistics problems, operation throughputs, asset utilization, and process validations for several Boeing Commercial Airplane manufacturing operations in 737, 747, 757, 767, and Wing Processes.
- Identified potential numbers of transportation vessels, inventories, and transit impacts for the Sonic Cruiser global component logistics plan using simulation modeling and optimization.
- Provided solutions and contributed in reducing the floor space by half for the 737 and 757 empennage assemblies through advanced discrete event simulation modeling.
- Presented MR&D and acted as a company consultant with vendors in the company-wide technical evaluation of Catia V5 in the Early Adopter Program.
- Led MR&D to integrate the V5 suite (CATIA, Delmia, & Enovia) into the way of doing business in design, process modeling, PDM, and information collection for projects. Played a central role in the evolving collaborative process information integration environment.
- Planed and managed simulation works for major airplane program projects focused on solving major problems and evaluating new business opportunities.
- Helped reduce the final assembly flow time from 24 to the capability of 20 days for the 747 final assembly moving line by applying simulation and analysis techniques. Successfully mentored an entry-level simulation engineer on this project.
- Provided mentorship and leadership to simulation engineers in Boeing to help them accumulate advanced simulation knowledge, skills and expertise throughout the company.
- Supported facility planning by means of discrete event simulation including transportation logistics.

- President of the Boeing Visual Studio.Net User Group. Hosted company-wide conferences across North America in 17 states plus Canada.
- Published technical papers and presentations at international conferences.
- Filed and earned a US patent.

Sr. Specialist Manufacturing Process Engineer,
Boeing Commercial Airplanes Wing Responsibility Center, 10/1997 – 6/1999

- Conducted and managed a process change initiative that involved the supplier (ALCOA), and in-house engineers, purchasing, and business management for an estimated net present value return of $3.8 millions in 1.2 years.
- Wrote T-Chord Cell Artifact inspection program, Kanban scheme, and led an effort in reducing T-Chord processing steps. Close to 1 million savings per year.
- Led a T-Cell BPSP/HVC Reliability Team in pursuing requirements of the Boeing HVC standards.
- Facilitated a cross functional team in configuring, developing, coding, and launching an artifact probing data collection system and developed the user interface software in VB5.
- Contributed technically (3D GD&T) during 737 & 777 T-Chord tolerance modifications via a team effort.
- Successfully configured an inspection data management system that changed manual data operations to automatic electronic data handling with supports from multi-departmental team members. This effort and the T-Chord tolerance change saved more than 12 person-hour per day in the inspection area.
- Developed software (in VB5) to assist other MBU in collecting tool numbers faster, easier, and more accurately as part of a team effort in cutter database development.
- "Lead" of six NC programmers and three Manufacturing Process Engineers.

Pilkington - Libbey Owens Ford Company, Toledo, Ohio, 7/1994 – 10/1997

Advanced Engineer II
North America Technical Center, 4/1996 – 10/1997

- Coordinated cellular manufacturing equipment development, design, purchase, installation and start-up.
- Innovated, programmed (in C++), and installed automatic robot cells with 12% throughput increase.
- Led robotic automation cell implementations, developed, programmed (CVIM II), and installed a new generation machine vision system that ran 18 times faster and within budget that yielded a more than $12 million worth of capacity gain per year.
- Invented a new lean automatic manufacturing system by using reliable PLCs (AB5) without dependency on non-reliable components and eliminated signal-transferring logistic bottlenecks.

- Wrote codes (in CAD and BASIC) that successfully contributed the first automatic robotic (ABB S4) laser inspection on automotive windshield glass contour through team works.
- Constructed simulation models in AutoMod for process constraint identifications.
- Provided technical support to plants; training classes for engineers, technicians, and union operators.
- Supported company-wide Coordinate Measuring Machine (CMM) and robotic applications.
- Wrote QS9000/ISO9000 instructions.

Manufacturing Specialist
OE Tooling and Development Center, 7/1994 – 4/1996

- Managed the manufacturing group of four engineers, two technicians, and six union personnel on a rotating basis in a self-directed team.
- Contributed in product development iteration time reductions from 4 - 6 weeks to 1 - 3 days. This yielded about $2 million dollars in savings.
- Participated in tool/fixture design, reverse engineering and rapid prototyping for new OE product development.
- Provided CMM technical supports and services to production plants.
- Installed, programmed, operated, and maintained a CMM for OE glass size checking and tool development.
- Supplied SPC information for OE glass GP-11 and PPAP for General Motors.
- Wrote QS9000/ISO9000 instructions.
- Wrote and managed financially justified capital projects.
- Trained engineers, contractors, and union operators.

Intermet - New River Castings Company, Radford, Virginia, 2/1990 – 6/1994
Quality/NDT Engineer

- Improved process capabilities from 3 to 4 Sigma by using statistical process control and design of experiments.
- Designed, integrated, purchased, installed, calibrated, and maintained lab instruments and cellular Non-Destructive Testing (NDT) manufacturing systems.
- Developed and implemented process control plans, FMEAs, and operating instructions through team effort (led a team of 25 members) and plant wide training (3800+ person-hour of operator training).
- Established and measured critical 3D dimensional characteristics for new parts and tooling per ANSI Y14.5M.
- Established and implemented alloy treatment tables for casting processes.
- Performed spectrographic metal microstructure analysis, NDT, and mechanical tests for raw materials and production parts per NIST, ASTM, and ASNT standards.
- Participated and contributed in winning the Ford Q1 award.
- Managed, evaluated, trained technicians and union operators in A2LA certified metrology, metallurgical, and NDT laboratories.

- Certified level 2 in both UT and RT by ASNT guidelines.

Marquette University, Milwaukee, Wisconsin, 1/1988 – 5/1989
Teaching Assistant

- For courses: Engineering Economy, Design on Machine Elements, and Energy Conversion Process.

Chinese Military Police Training Headquarters, Taiwan, 7/1985 – 5/1987
Military Police

- Managed five cooks, ran the best military kitchen in Taiwan, and won a Citation.
- Maintained food inventory precisely, purchased food, issued non-repetitive weekly menu, controlled budget, and always served food on time to a group of 180 persons.
- Directed military related traffic on public roads (only occasionally).

Kennex Co., Ltd., Taiwan, 7/1984 – 9/1984
Intern Quality Control Engineer

- Monitored the quality of aluminum tennis rackets from raw material to final products in every step of the manufacturing process.

PUBLICATIONS and PRESENTATIONS

[1] Lu, R.F., Petersen, T.D., and Storch, R.L., (2008) "Modeling Customized Product Configuration in Large Assembly Manufacturing with Supply Chain Considerations," Internal Journal of Flexible Manufacturing System, submitted and accepted for publication.

[2] Lu, R.F., Storch, R.L., Goto, J., and Kluczny, B., (2008) "Modeling a Large Scale Production System with Converging Production Lines and System Learning," Proceedings of the IIE Conference, Vancouver, BC, Canada, submitted and accepted for publication.

[3] Lu, R.F. and Storch, R.L., (2008) "Simulation Approaches to Practice Lean in Education and Training," Proceedings of the IERC Conference, Vancouver, BC, Canada, submitted and accepted for publication.

[4] Storch, R.L., Lu, R.F., and Petersen, T.D., (2007) "Optimizing Product Configuration Decision Times In Shipbuilding," Proceedings of the 13th International Conference on Computer Applications in Shipbuilding, Portsmouth, UK.

[5] Lu, R.F., Petersen, T.D., and Storch, R.L., (2007) "Asynchronous Stochastic Learning Curve Effects in Engineering-To-Order Customization Processes," International Journal of Production Research, submitted and accepted for publication.

[6] Lu, R.F. and Storch, R.L., (2007) "A Statistical Verification Method in Modeling Mass Customization in a Production System of Asynchronous Stochastic Learning Curves," Proceedings of the IERC Conference, Nashville, TN, USA.

[7] Petersen, T.D., Lu, R.F., and Storch, R.L., (2007) "Postponement of Product Configuration Decisions in ETO Industries," Proceedings of the 6th International Conference on Computer Applications and Information Technology in the Maritime Industries, Cortona, Italy.

[8] Lu, R.F., Storch, R.L., Goto, J., and Kluczny, B., (2007) "Modeling of Scheduling Condition Changes in a Customized Large Scale Production System," Proceedings of the 4th Interdisciplinary World Congress on Mass Customization and Personalization, Cambridge, MA, USA. Submitted for MCPC2008 Book publication.

[9] Zeng, Q.L., Tseng, M.M., and Lu, R.F., (2006) "Staged Postponement of Order Specification Commitment for Supply Chain Management," Proceedings of the 56th CIRP General Assembly, Kobe, Japan.

[10] Lu, R.F. and Storch, R.L., (2006) "Simulation Modeling of Customized Product Family Decision Points and Logistics," Proceedings of the IERC Conference, Orlando, FL, USA.

[11] Lu, R.F., (2006) Presentation: "Mass Customization in Large Scale Production System: Case Studies," Taipei Mass Customization Conference, Taipei, Taiwan.

[12] Lu, R.F., (2006) Presentation: "A Modeling Method to Evaluate Resource Levels for Product Mass Customization Decisions," Proceedings of the INFORMS International Conference, Hong Kong, China.

[13] Lu, R.F. and Storch, R.L, (2006) "Modeling of Mass Customization Deployment to Long Life-Cycle Products in a Large Scale Manufacturing System," Proceedings of the 17th Annual Conference of POMS – OM in the New World Uncertainties, Boston, MA, USA.

[14] Lu, R.F., (2006) Presentation: "Simulation Modeling of a Worldwide Production System Logistics," Proceedings of the SAE – AMAF Conference, Toulouse, France.

[15] Qiao, G., Lu, R.F., and McLean, C., (2006) "Flexible Manufacturing Systems for Mass Customisation Manufacturing," International Journal of Mass Customisation, Vol. 1, Issue 2/3, pp. 374 – 393.

[16] Lu, R.F. and Storch, R.L., (2005) "Modeling of Supplier Event Management Influences in Large Manufacturing System Integration," Proceedings of APMS 2005 - IFIP 5.7 Conference on "Advances in Production Mgt Systems", NIST publishing, Washington DC, USA.

[17] Sundaram, S., and Lu, R.F., (2005) Presentation: "Modeling Practices of a New Product Development at an Integrated Large Manufacturing System – A Case Demonstration," Proceedings of IIE – Simulation Solutions Conference, Atlanta, GA, USA.

[18] Lu, R.F. (2005) Presentation: "Supply Chain Simulation Applications: A Transporter Driven Logistics Study", International Federation of Operational Research Societies (INFORMS/IFORS), Honolulu, Hawaii, USA.

[19] Lu, R.F. and Storch, R.L., (2005) "Modeling of Customer Decision Point and Design Change Impact in Customized Large Manufacturing System Integration," Proceedings of the 3rd Interdisciplinary World Congress on Mass Customization and Personalization, Hong Kong, China.

[20] Lu, R.F. (2005) Presentation: "Simulation Modeling of Customer Selectable Catalogue Items in an Integrated Production System," Proceedings of Aerospace Manufacturing Technology Conference (AMTC) by SAE, Dallas / Ft. Worth, TX, USA

[21] Lu, R.F. and Li, L., (2004) Presentation: "Large Scale Integrated Manufacturing System: Supply Chain Modeling Using Discrete Event Simulation", Proceedings of the 2nd International Seminar on Digital Enterprise Technology, Seattle, Washington, USA.

[22] Qiao, G., Lu, R.F., and McLean, C., (2004) Presentation: "Enterprise Adaptive Information Integration of Manufacturing Abstract Data Type and Simulation Flows", Proceedings of the 2nd International Seminar on Digital Enterprise Technology, Seattle, Washington, USA.

[23] Lu, R.F., Li, L., and Kim, K.C., (2004) "Traceable Part Batching Performance Modeling: A Simulation Case Study", Proceedings of Aerospace Manufacturing Technology Conference (AMTC) by SAE, St. Louis, MO, USA.

[24] Lu, R.F. and Storch, R.L., (2004) "Large Scale Manufacturing System Integration: Modeling of a Global Component Transportation Logistics Case", Proceedings of the 4th IASTED International Conference on Modeling, Simulation, and Optimization, Kauai, Hawaii, USA.

[25] Lu, R.F., (2004) "Design and Configuration of Machine Vision Robotic Cells in a Manufacturing System", Proceedings of the 2004 DETC: 28th Biennial Mechanisms and Robotics Conference by ASME, Salt Lake City, UT, USA.

[26] Qiao, G. and Lu, R.F., (2004) "Process control and logistics management for Mass Customization Manufacturing", Proceedings of the IERC, Houston, TX, USA.

[27] Qiao, G. and Lu, R.F., (2003) "Flexible Manufacturing System for Mass Customization Manufacturing", Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization, Munich, Germany.

[28] Lu, R.F. and Qiao, G., (2003) "Simulation Modeling to Optimize Stochastic Manufacturing Processes and Resources by a Dynamic Monte Carlo Method", Proceedings of the IERC conference, New Orleans, LA, USA.

[29] Qiao, G., Lu, R.F., and Riddick, F., (2003) "Flexible Modeling and Simulation for Mass Customization Manufacturing", Proceedings of the IERC, New Orleans, LA, USA.

[30] Lu, R.F., Qiao, G., and McLean, C., (2003) "NIST XML Simulation Interface Specification at Boeing: A Case Study", Winter Simulation Conference Proceedings, Vol. 2, pp. 1230 – 1237.

[31] McLean, C., Leong, S., Zimmerman, P., Harreil, C., and Lu, R.F., (2003) "Simulation standards: Current Status, Needs, and Future Directions", Winter Simulation Conference Proceedings, Vol. 2, pp. 2019 – 2026.

[32] Lu, R.F., Qiao, G., and Snyders, G., (2003) Presentation: "An XML Enabled Simulation Modeling Case Study", Delmia Worldwide User Conference 2003 – North America, Auburn Hills, MI, USA.

[33] Lu, R.F. and Sundaram, S., (2002) "Manufacturing process modeling of Boeing 747 moving line concepts", Winter Simulation Conference Proceedings, Vol. 1, 2002, pp. 1041 – 1045.

[34] Lu, R.F., (2002) Presentation: "Integrated Discrete Event Simulation Presentation Media", Delmia International Digital Manufacturing Conference 2002, Troy, MI, USA.

[35] Lu, R.F., (2001) Presentation: "Lean Manufacturing Practices in the T-Chord Cell", Aerospace Congress & Exhibition by Aerospace North America & SAE, Seattle, WA, USA.

[36] Lu, R.F. and Gross, L., (2001) "Simulation Modeling of a Pull and Push Assemble-To-Order System", Proceedings of the European Operational Research Conference EURO 2001, Rotterdam, The Netherlands.

[37] Lu, R.F., (2000) Presentation: "Simulation in Wing Manufacturing Process Management", Delmia International Digital Manufacturing Conference 2000, Troy, MI, USA.

INVENTIONS

[38] Lu, R.F., Sundaram, S., and Wei, T.F., (2008) "Statistically Prioritized Discrete Modeling Methodology which Produces Optimized Production Forecasts Depicting Dynamic Critical Path Re-configuration Based on Process Re-sequencing of a Stochastic Event Driven Pull-based Process in a Large-Scale Manufacturing Push-based Production System." Invention Disclosure filed, February, 2008.

[39] Lu, R.F., (2007) "Asynchronous Stochastic Learning Curve Effects in a Large Scale Production System," Invention Disclosure filed, January, 2007; provision patent filed to the US Patent office, May, 2007.

[40] Lu, R.F., (2006) "Systems and Methods for Manufacturing a Product in a Pull and Push Manufacturing System and Associated Methods and Computer Program Products for Modeling the Same," application Number: 10/172,709, filed: June 14, 2002, awarded: January 3, 2006, US PATENT (6,983,189)

[41] Lu, R.F. and Thim, R. (2005) "Modeling a Large Scale Production System," Invention Disclosure filed, May, 2005.

PROFESSIONAL AFFLIATIONS

Professional Societies:

Institute of Industrial Engineers (IIE) – Senior Member
International Institute of Mass Customization and Personalization (IIMCP) – Founding Member
Institute for Operations Research and the Management Science (INFORMS)
Society of Automotive Engineers (SAE – International)
American Society of Mechanical Engineers (ASME)
Simulation Interoperability Standards Organization (SISO)
American Society for Metals (ASM - International)
American Society for Quality Control (ASQC)
American Society for Nondestructive Testing (ASNT)
American Foundrymen's Society (AFS)

Professional Services:

Conference/Seminar Chairs
Boeing/Microsoft Web Application Developers' Conferences 2000 – 2005

Invited Program Committee / Speaker
Keynote speaker:
The 2nd International Conference on Mass Customization, Grand Rapids, Michigan, 2006.
Taipei Mass Customization Conference, 2006.
The Inauguration Conference of The Smart Customization Group, Massachusetts Institute of Technology, MA, USA, 2007.

Program Committee:
The 2nd International Seminar on Digital Enterprise Technology, Seattle, Washington, 2004
The INFORMS Simulation Society, 2006
The INFORMS Practitioner and Practice Society, 2006
The World Conference on Mass Customization and Personalization, 2007
Boeing Technical Excellence Conference 2008

Session chairs:
The European Operational Research Conference 2001
Winter Simulation Conference 2002
Winter Simulation Conference 2003
IASTAD – Modeling Simulation and Optimization 2004
INFORMS International Conference, Hawaii, 2005
INFORMS International Conference, Hong Kong, 2006
POMS, 2006
SAE, Toulouse, France, 2006
The World Conference on Mass Customization and Personalization, 2007
SAE, Los Angles, California, 2007

Reviewer:
IIE Annual Research Conference 2003
IIE Annual Research Conference 2004
IIE Annual Research Conference 2005
IIE Annual Research Conference 2006
IIE Annual Research Conference 2007
ASME – DETC 2004
IEEE Transactions of Engineering Management 2005
Boeing Technical Excellence Conference 2005
SAE International 2006
SAE International 2007

Judge:
National Council of Examiners for Engineering and Surveying (NCEES):
- PE exams 2003, 2004, and 2005.
- FE exams 2004, 2005, 2006, and 2007.
Boeing Technical Excellence Conference 2005

TECHNICAL SKILLS

Domain Knowledge:
Asynchronous stochastic learning curves in large scale production system.
Discrete event, Monte Carlo, and spatial simulation modeling and optimization.
Large scale, continuous flow, and job shop manufacturing systems.
Mass customized manufacturing and personalization.

Non-linear integer programming.
Supply chain management logistics.
Computer integrated manufacturing system of machine vision automations.
Flexible manufacturing system of robotic equipment.
Manufacturing quality control and statistical process control.
Lean and agile manufacturing implementations with JIT.
Metrology automation programming and implementation.
Table layout - a trade of manual dimensional inspections.
Metallurgy in precision ductile iron castings.
Thermodynamics and spectrographic analysis of metals.
Tooling and molding pattern design on ductile iron castings.
Non-Destructive testing in UT and RT.
Computer architecture configuration and program coding.
Implementation of research projects.

Operating Systems:

MS-Windows Vista/XP/2000/98/NT4.0, Active Directory, LAN, UNIX, and DOS.

Languages:

C, C++, FORTRAN, Pascal, Q-BASIC, HTML, BCL/SCL, UML, XML, SQL, and Visual Basic.

Software:

Delmia Quest, Arena, OptQuest, Crystal Ball, R, S-Plus, Vensim, Maple, MiniTab, Matlab, Logware, ProModel, 3-D CAD solid & freeform (CATIA V5, Intergraph EMS, AutoSurf of AutoCAD, MicroStation, IronCAD), 6-ax robot (ABB S4), 3-D CMM (LK, Brown & Sharpe, Sheffield, CheckMate) w/GD&T, CAM on 21/2-ax and 5 ax NC machines, Visual Studio dot Net, and MS-Office suites.

HONORS and ACTIVITIES

- Participant of multiple Boeing Leadership Programs (2000 – 2007).
- Invited Subject Matter Experts on the nation-wide Professional Engineer (PE) and Fundamentals of Engineering (FE) exam board that approves scope, contents, questions, and passing criteria of annual PE and FE exams in Industrial Engineering (2003 – 2007).
- President of the Boeing Visual Studio dot Net Users Group (Since 4/2002).
- Numerous Boeing stock, cash, and stock option awards (1999 – 2007).
  - More than 10 recognition awards from Boeing SSG, IDS, and BCA in 2006.
  - Special Cash Award, for simulation modeling of the 7E7 global transportation logistics (4/2004).
  - Award of Boeing Stocks, for the success in the fellowship program (1/2003).
  - Achievement Award, for organizing a W2K conference (6/2002).
  - Achievement Award, for hosting a company-wide Windows 2000 application conference (8/2001).

  - Achievement Award, for supporting Windows 2000 application migrations in Boeing (8/2001).
  - Special Incentive Award of Boeing Stock options, for simulation support to the 747 (7/2001).
  - Outstanding Performance Award, for hosting a company-wide WEB Application Development conference (12/2000).
  - Special Incentive Award of Boeing Stocks, for supporting Win2K and the 737/757 wing panel shop simulation (9/2000).
  - Certificate of Appreciation, for the success in the wing stringer tool code extraction program (7/1999).
  - Certificate of Appreciation, for efforts on the first side-of-body chords in the 767-400 program (3/1999).
- Secretary of the Boeing Visual Basic Users Group (4/1999 – 4/2002).
- Teaching Assistant scholarship. Marquette University (1988 – 1989).
- Citation for Loyal and Patriotic with Excellent Performance, for operating the best military kitchen, from the Chief of the General Staff in Taiwan (1987).
- Certificate of Good Conduct at the Feng Chia University (1984).
- Leader of the Chinese Classical Music Orchestra at the Feng Chia University (1982 – 1984).
- President of the Society of Material Science Department at the Feng Chia University (1981 – 1982).